36.6.4 Synchronization

Not applicable.

36.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	CFBS[2:0]			AESMODE[2:0]			ENABLE	SWRST
		15:8		XORKEY	KEYGEN	LOD	STARTMODE	CIPHER	KEYSIZE[1:0]	
		23:16					CTYPE[3:0]			
		31:24								
0x04	CTRLB	7:0					GFMUL	EOM	NEWMSG	START
0x05	INTENCLR	7:0							GFMCMP	ENCCMP
0x06	INTENSET	7:0							GFMCMP	ENCCMP
0x07	INTFLAG	7:0							GFMCMP	ENCCMP
0x08	DATABUFPTR	7:0							INDATAPTR[1:0]	
0x09	DBGCTRL	7:0								DBGRUN
0x0A ... 0x0B	Reserved									
0C	KEYWORD0	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
10	KEYWORD1	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
14	KEYWORD2	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
18	KEYWORD3	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
1C	KEYWORD4	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
20	KEYWORD5	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
24	KEYWORD6	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
28	KEYWORD7	7:0	KEYWORD[7:0]							
		15:8	KEYWORD[15:8]							
		23:16	KEYWORD[23:16]							
		31:24	KEYWORD[31:24]							
0x2C ... 0x37	Reserved									
0x38	INDATA	7:0	INDATA[7:0]							
		15:8	INDATA[15:8]							
		23:16	INDATA[23:16]							
		31:24	INDATA[31:24]							
3C	INTVECTV0	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
40	INTVECTV1	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							
44	INTVECTV2	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							
48	INTVECTV3	7:0	INTVECTV[7:0]							
		15:8	INTVECTV[15:8]							
		23:16	INTVECTV[23:16]							
		31:24	INTVECTV[31:24]							
0x4C ... 0x5B	Reserved									
0x5C	HASHKEY0	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x60	HASHKEY1	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x64	HASHKEY2	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x68	HASHKEY3	7:0	HASHKEY[7:0]							
		15:8	HASHKEY[15:8]							
		23:16	HASHKEY[23:16]							
		31:24	HASHKEY[31:24]							
0x6C	GHASH0	7:0	GHASH[7:0]							
		15:8	GHASH[15:8]							
		23:16	GHASH[23:16]							
		31:24	GHASH[31:24]							
0x70	GHASH1	7:0	GHASH[7:0]							
		15:8	GHASH[15:8]							
		23:16	GHASH[23:16]							
		31:24	GHASH[31:24]							
0x74	GHASH2	7:0	GHASH[7:0]							
		15:8	GHASH[15:8]							
		23:16	GHASH[23:16]							
		31:24	GHASH[31:24]							
0x78	GHASH3	7:0	GHASH[7:0]							
		15:8	GHASH[15:8]							
		23:16	GHASH[23:16]							
		31:24	GHASH[31:24]							
0x7C ... 0x7F	Reserved									
0x80	CIPLEN	7:0	CIPLEN[7:0]							
		15:8	CIPLEN[15:8]							
		23:16	CIPLEN[23:16]							
		31:24	CIPLEN[31:24]							
0x84	RANDSEED	7:0	RANDSEED[7:0]							
		15:8	RANDSEED[15:8]							
		23:16	RANDSEED[23:16]							
		31:24	RANDSEED[31:24]							

36.8 Register Description

Registers can be 8, 16, or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description. See *Register Access Protection* from Related Links.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Related Links

36.5.8. Register Access Protection

36.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Enable-protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					CTYPE[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
		XORKEY	KEYGEN	LOD	STARTMODE	CIPHER	KEYSIZE[1:0]	
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CFBS[2:0]			AESMODE[2:0]			ENABLE	SWRST
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:16 – CTYPE[3:0] Counter Measure Type

Value	Name	Description
XXX0	CTYPE1 disabled	Countermeasure1 disabled
XXX1	CTYPE1 enabled	Countermeasure1 enabled
XX0X	CTYPE2 disabled	Countermeasure2 disabled
XX1X	CTYPE2 enabled	Countermeasure2 enabled
X0XX	CTYPE3 disabled	Countermeasure3 disabled
X1XX	CTYPE3 enabled	Countermeasure3 enabled
0XXX	CTYPE4 disabled	Countermeasure4 disabled
1XXX	CTYPE4 enabled	Countermeasure4 enabled

Bit 14 – XORKEY XOR Key Operation

Value	Description
0	No effect
1	The user keyword gets XORed with the previous keyword register content.

Bit 13 – KEYGEN Last Key Generation

Value	Description
0	No effect
1	Start Computation of the last NK words of the expanded key

Bit 12 – LOD Last Output Data Mode

Value	Description
0	No effect
1	Start encryption in Last Output Data mode

Bit 11 – STARTMODE Start Mode Select

Value	Name	Description
0	Manual Mode	Start Encryption / Decryption in Manual mode
1	Auto Mode	Start Encryption / Decryption in Auto mode

Bit 10 – CIPHER Cipher Mode Select

Value	Description
0	Decryption
1	Encryption

Bits 9:8 – KEYSIZE[1:0] Encryption Key Size

Value	Name	Description
0	128-bit Key	128-bit Key for Encryption / Decryption
1	192-bit Key	192-bit Key for Encryption / Decryption
2	256-bit Key	256-bit Key for Encryption / Decryption
3	Reserved	Reserved

Bits 7:5 – CFBS[2:0] Cipher Feedback Block Size

Value	Name	Description
0	128-bit data block	128-bit Input data block for Encryption/Decryption in Cipher Feedback mode
1	64-bit data block	64-bit Input data block for Encryption/Decryption in Cipher Feedback mode
2	32-bit data block	32-bit Input data block for Encryption/Decryption in Cipher Feedback mode
3	16-bit data block	16-bit Input data block for Encryption/Decryption in Cipher Feedback mode
4	8-bit data block	8-bit Input data block for Encryption/Decryption in Cipher Feedback mode
5-7	Reserved	Reserved

Bits 4:2 – AESMODE[2:0] AES Modes of Operation

Value	Name	Description
0	ECB	Electronic code book mode
1	CBC	Cipher block chaining mode
2	OFB	Output feedback mode
3	CFB	Cipher feedback mode
4	Counter	Counter mode
5	CCM	CCM mode
6	GCM	Galois counter mode
7	Reserved	Reserved

Bit 1 – ENABLE Enable

Value	Description
0	The peripheral is disabled
1	The peripheral is enabled

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the AES module to their initial state, and the module will be disabled.
Writing a '1' to SWRST will always take precedence, meaning that all other writes in the same write operation will be discarded.

Value	Description
0	There is no reset operation ongoing
1	The reset operation is ongoing

36.8.2 Control B

Name: CTRLB
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
					GFMUL	EOM	NEWMSG	START
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – GFMUL GF Multiplication
This bit is applicable only to GCM mode.

Value	Description
0	No action
1	Setting this bit calculates GF multiplication with data buffer content and hashkey register content.

Bit 2 – EOM End of Message
This bit is applicable only to GCM mode.

Value	Description
0	No action
1	Setting this bit generates final GHASH value for the message.

Bit 1 – NEWMSG New Message
This bit is used in cipher block chaining (CBC), cipher feedback (CFB) and output feedback (OFB), counter (CTR) modes to indicate the hardware to use Initialization vector for encrypting the first block of message.

Value	Description
0	No action
1	Setting this bit indicates start of new message to the module.

Bit 0 – START Start Encryption/Decryption

Value	Description
0	No action
1	Start encryption / decryption in manual mode.

36.8.3 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x05
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (`INTENSET`) register.

Bit	7	6	5	4	3	2	1	0
							GFMCMP	ENCCMP
Access							R/W	R/W
Reset							0	0

Bit 1 – GFMCMP GF Multiplication Complete Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the GF Multiplication Complete Interrupt Enable bit, which disables the GF Multiplication Complete interrupt.

Value	Description
0	The GF Multiplication Complete interrupt is disabled.
1	The GF Multiplication Complete interrupt is enabled.

Bit 0 – ENCCMP Encryption Complete Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Encryption Complete Interrupt Enable bit, which disables the Encryption Complete interrupt.

Value	Description
0	The Encryption Complete interrupt is disabled.
1	The Encryption Complete interrupt is enabled.

36.8.4 Interrupt Enable Set

Name: INTENSET
Offset: 0x06
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (`INTENCLR`) register.

Bit	7	6	5	4	3	2	1	0
							GFMCMP	ENCCMP
Access							R/W	R/W
Reset							0	0

Bit 1 – GFMCMP GF Multiplication Complete Interrupt Enable
Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the GF Multiplication Complete Interrupt Enable bit, which enables the GF Multiplication Complete interrupt.

Value	Description
0	The GF Multiplication Complete interrupt is disabled.
1	The GF Multiplication Complete interrupt is enabled.

Bit 0 – ENCCMP Encryption Complete Interrupt Enable
Writing a '0' to this bit has no effect. Writing a '1' to this bit will clear the Encryption Complete Interrupt Enable bit, which enables the Encryption Complete interrupt.

Value	Description
0	The Encryption Complete interrupt is disabled.
1	The Encryption Complete interrupt is enabled.

36.8.5 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x07
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
							GFMCMP	ENCCMP
Access							R/W	R/W
Reset							0	0

Bit 1 – GFMCMP GF Multiplication Complete
This flag is cleared by writing a '1' to it.
This flag is set when GHASH value is available on the Galois Hash Registers (GHASHx) in GCM mode.
Writing a '0' to this bit has no effect.
This flag is also automatically cleared in the following cases.

1. Manual encryption/decryption occurs (START in CTRLB register).
2. Reading from the GHASHx register.

Bit 0 – ENCCMP Encryption Complete
This flag is cleared by writing a '1' to it.
This flag is set when encryption/decryption is complete and valid data is available on the Data Register.
Writing a '0' to this bit has no effect.
This flag is also automatically cleared in the following cases:

1. Manual encryption/decryption occurs (START in CTRLA register). (This feature is needed only if we do not support double buffering of INDATA registers).
2. Reading from the data register (INDATAx) when LOD = 0.
3. Writing into the data register (INDATAx) when LOD = 1.
4. Reading from the Hash Key register (HASHKEYx).

36.8.6 Data Buffer Pointer

Name: DATABUFPTR
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
							INDATAPTR[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 1:0 – INDATAPTR[1:0] Input Data Pointer
Writing to this field changes the value of the input data pointer, which determines which of the four data registers is written to/read from when the next write/read to the `INDATA` register address is performed.

36.8.7 Debug

Name: DBGCTRL
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								W
Reset								0

Bit 0 – DBGRUN Debug Run

Writing a '0' to this bit causes the AES to halt during debug mode.
Writing a '1' to this bit allows the AES to continue normal operation during debug mode. This bit can only be changed while the AES is disabled.

36.8.8 Keyword

Name: KEYWORD
Offset: 0x0C + n*0x04 [n=0..7]
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	KEYWORD[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	KEYWORD[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	KEYWORD[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	KEYWORD[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – KEYWORD[31:0] Key Word Value

The four/six/eight 32-bit Key Word registers set the 128-bit/192-bit/256-bit cryptographic key used for encryption/decryption. `KEYWORD0.KEYWORD` corresponds to the first word of the key and `KEYWORD3/KEYWORD5/KEYWORD7.KEYWORD` to the last one.

Note: By setting the XORKEY bit of CTRLA register, keyword will update with the resulting XOR value of user keyword and previous keyword content.

36.8.9 Data

Name: INDATA
Offset: 0x38
Reset: 0x00000000

Bit	31	30	29	28	27	26	25	24
	INDATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INDATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INDATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INDATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INDATA[31:0] Data Value

A write to or read from this register corresponds to a write to or read from one of the four data registers. The four 32-bit Data registers set the 128-bit data block used for encryption/decryption. The data register that is written to or read from is given by the `DATABUFPTR.INDATPTR` field.

Note: Both input and output shares the same data buffer. Reading INDATA register will return 0's when AES is performing encryption or decryption operation.

36.8.10 Initialization Vector Register

Name: INTVECTV
Offset: 0x3C + n*0x04 [n=0..3]
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	INTVECTV[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTVECTV[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INTVECTV[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INTVECTV[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INTVECTV[31:0] Initialization Vector Value

The four 32-bit Initialization Vector registers `INTVECTVn` set the 128-bit Initialization Vector data block that is used by some modes of operation as an additional initial input. `INTVECTV0.INTVECTV` corresponds to the first word of the Initialization Vector, `INTVECTV3.INTVECTV` to the last one. These registers are write-only to prevent the Initialization Vector from being read by another application. For CBC, OFB, and CFB modes, the Initialization Vector corresponds to the initialization vector. For CTR mode, it corresponds to the counter value.

36.8.11 Hash Key (GCM mode only)

Name: HASHKEY
Offset: 0x5C + n*0x04 [n=0..3]
Reset: 0x00000000
Property: PAC Write-protection

Bit	31	30	29	28	27	26	25	24
	HASHKEY[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HASHKEY[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HASHKEY[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HASHKEY[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HASHKEY[31:0] Hash Key Value

The four 32-bit `HASHKEY` registers contain the 128-bit Hash Key value computed from the AES KEY. The Hash Key value can also be programmed offering single GF128 multiplication possibilities.

36.8.12 Galois Hash (GCM mode only)

Name: GHASH
Offset: 0x6C + n*0x04 [n=0..3]
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	GHASH[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	GHASH[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	GHASH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	GHASH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – GHASH[31:0] Galois Hash Value

The four 32-bit Hash Word registers GHASHcontain the GHASH value after GF128 multiplication in GCM mode. Writing a new key to KEYWORD registers causes GHASH to be initialized with zeroes. These registers can also be programmed.

36.8.13 Galois Hash x (GCM mode only)

Name: CIPLEN
Offset: 0x80
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	CIPLEN[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CIPLEN[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CIPLEN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CIPLEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CIPLEN[31:0] Cipher Length
This register contains the length in bytes of the Cipher text that is to be processed. This is programmed by the user in GCM mode for Tag generation.

36.8.14 Random Seed

Name: RANDSEED
Offset: 0x84
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
	RANDSEED[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RANDSEED[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RANDSEED[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RANDSEED[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – RANDSEED[31:0] Random Seed
A write to this register corresponds to loading a new seed into the Random number generator.

37. Public Key Cryptography Controller (PUKCC)

37.1 Overview

The Public Key Cryptography Controller (PUKCC) processes public key cryptography algorithm calculus in both GF(p) and GF(2n) fields.

The Public Key Cryptography Library (PUKCL) is stored in ROM inside the device. The library can be used in applications to access features of PUKCC, and includes the complete implementation of the following public key cryptography algorithms:

- RSA (Rivest-Shamir-Adleman public key cryptosystem), DSA (Digital Signature Algorithm):
  - Modular Exponentiation with CRT up to 7168 bits
  - Modular Exponentiation without CRT up to 5376 bits
  - Prime generation
  - Utilities: GCD/modular Inverse, Divide, Modular reduction, Multiply, ...
- Elliptic Curves:
  - ECDSA GF(p) up to 521 bits for common curves (up to 1120 bits for future use)
  - ECDSA GF(2n) up to 571 bits for common curves (up to 1440 bits for future use)
  - Choice of the curve parameters for compatibility with NIST Curves or other curves in Weierstrass equation
  - Point Multiply
  - Point Add/Doubling
  - Other high level elliptic curve algorithms (ECDH, ...) can be implemented by user using library functions
- Deterministic Random Number Generation (DRNG ANSI X9.31) for DSA

37.2 Product Dependencies

37.2.1 I/O Lines

Not applicable.

37.2.2 Power Management

The PUKCC will continue to operate in any sleep mode, as long as its source clock is running.

37.2.3 Clocks

The bus clock (PB2_CLK) can be enabled and disabled by the CRU.

37.2.4 DMA

Not applicable.

37.2.5 Interrupts

Not applicable.

37.2.6 Events

Not applicable.

37.3 Functional Description

37.3.1 Public Key Cryptography Library (PUKCL) Application Programming Interface (API)

The Public Key Cryptography Controller (PUKCC) is a peripheral that can be used to accelerate public key cryptography, and processes public key cryptography algorithm calculus in both Prime field (GF(p)) and Binary field

(GF(2^n)). Different functionalities of the PUKCC are accessed with the help of the Public Key Cryptography Library (PUKCL), which is embedded into a dedicated ROM inside the microcontroller.

The PUKCL provides access to many algorithms and functions. The features provided, start from basic addition or comparison, up to the RSA or ECDSA complete computation. The library can be utilized by including the PUKCL Driver in the application and passing parameters through a common Application Programming Interface (API). The PUKCC Driver is available in Harmony 3. This library can be used in conjunction with a SSL software stack to improve performance and helps to reduce the RAM usage and time taken to perform different cryptographic functions.

37.3.2 PUKCL Features

PUKCL features include:

- 37.3.4. Basic Arithmetic and Cryptographic Services - PUKCL self-test, GCD, integral division, etc.
- 37.3.5. Modular Arithmetic Services - Modular reduction, modular exponentiation, probable prime generation and modular exponentiation
- 37.3.6. Elliptic Curves Over GF(p) Services - Point addition and doubling on an elliptic curve in a prime field, ECDSA signature generation and verification on an elliptic curve over GF(p)
- 37.3.7. Elliptic Curves Over GF(2n) Services - Point addition and doubling on an elliptic curve in a prime field, ECDSA signature generation and verification on an elliptic curve over GF(2^n)

37.3.3 PUKCL Usage

The following sections provide details on accessing the PUKCL and its features.

37.3.3.1 Initializing the PUKCC and PUKCL

For a project created with Harmony 3, the clock initialization is handled by the initialization function CLK_Initialize(). After a power-on reset, and when the PUKCC Clock is enabled, a Crypto RAM clear process is launched. It is mandatory to wait until the end of this process before using the Crypto Library.

The following code shows how to wait for the Crypto RAM clear process.

```
while ((PUKCCSR & BIT_PUKCCSR_CLRRAM_BUSY) != 0);
```

The next task to be done is self-test. From the generated project in Harmony 3, copy the example for the PUKCC Driver SelfTest and add it to the main source file. This is a mandatory step before using the library. The return values from the SelfTest service must be compared against known values mentioned in the service description (see the **Description** section in 37.3.4.1. SelfTest).

Example 37-1. PUKCC Initialization

```
void PUKCC_self_test(void)
{
    // Clear contents of PUKCLParam
    memset(&PUKCLParam, 0, sizeof(PUKCL_PARAM));

    pvPUKCLParam = &PUKCLParam;
    vPUKCL_Process(SelfTest, pvPUKCLParam);

    // In case of error, loop here
    while (PUKCL(u2Status) != PUKCL_OK) {
    ;
    }
    while (pvPUKCLParam->P.PUKCL_SelfTest.u4Version != PUKCL_VERSION) {
    ;
    }
    while (pvPUKCLParam->P.PUKCL_SelfTest.u4CheckNum1 != 0x6E70DDD2) {
    ;
    }
    while (pvPUKCLParam->P.PUKCL_SelfTest.u4CheckNum2 != 0x25C8D64F) {
    ;
    }
}

int main(void)
{
```

```
    /* Initializes MCU, drivers and middleware */
    SYS_Initialize();

    // Wait for Crypto RAM clear process
    while ((PUKCCSR & BIT_PUKCCSR_CLRRAM_BUSY) != 0);

    // Initialize PUKCC and perform self test
    PUKCC_self_test();
    while(1)
    {
    }
}
```

Note: It may also be necessary to initialize the Random Number Generator (RNG) on the microcontroller, as some services in the library use the peripheral. Before calling such services, be sure to follow the directives given for random number generation on the selected microcontroller (particularly initialization and seeding) and compulsorily start the RNG. For details refer to each service.

37.3.3.2 Accessing Different Library Services

All cryptographic services in the library are accessed by the macro vPUKCL_Process. All of these services use the same process for receiving and returning parameters. PUKCL receives two arguments: the requested service and a pointer to a structure called the parameter block. The parameter block contains two structures, a common parameter structure for all commands and specific parameter structure for each service. A specific service is accessed with vPUKCL_Process by passing the service name as the first argument. For example, to perform SelfTest, use vPUKCL_Process(SelfTest, pvPUKCLParam).

Example 37-2. PUKCL Parameter Block

```
typedef struct _PUKCL_param {
    PUKCL_HEADER PUKCL_Header;
    union {
    _PUKCL_CLEARFLAGS PUKCL_ClearFlags;
    _PUKCL_COMP       PUKCL_Comp;
    _PUKCL_CONDCOPY   PUKCL_CondCopy;
    _PUKCL_CRT        PUKCL_CRT;
    _PUKCL_DIV        PUKCL_Div;
    _PUKCL_EXPMOD     PUKCL_ExpMod;
    _PUKCL_FASTCOPY   PUKCL_FastCopy;
    _PUKCL_FILL       PUKCL_Fill;
    _PUKCL_FMULT      PUKCL_Fmult;
    _PUKCL_GCD        PUKCL_GCD;
    _PUKCL_PRIMEGEN   PUKCL_PrimeGen;
    _PUKCL_REDMOD     PUKCL_RedMod;
    _PUKCL_RNG        PUKCL_Rng;
    _PUKCL_SELFTEST   PUKCL_SelfTest;
    _PUKCL_SMULT      PUKCL_Smult;
    _PUKCL_SQUARE     PUKCL_Square;
    _PUKCL_SWAP       PUKCL_Swap;

    // ECC
    _PUKCL_ZPECCADD                   PUKCL_ZpEccAdd;
    _PUKCL_ZPECCDBL                   PUKCL_ZpEccDbl;
    _PUKCL_ZPECCADDSUB                PUKCL_ZpEccAddSub;
    _PUKCL_ZPECCMUL                   PUKCL_ZpEccMul;
    _PUKCL_ZPECDSAGENERATE            PUKCL_ZpEcDsaGenerate;
    _PUKCL_ZPECDSAVERIFY              PUKCL_ZpEcDsaVerify;
    _PUKCL_ZPECDSAQUICKVERIFY         PUKCL_ZpEcDsaQuickVerify;
    _PUKCL_ZPECCQUICKDUALMUL          PUKCL_ZpEccQuickDualMul;
    _PUKCL_ZPECCONVPROJTOAFFINE       PUKCL_ZpEcConvProjToAffine;
    _PUKCL_ZPECCONVAFFINETOPROJECTIVE PUKCL_ZpEcConvAffineToProjective;
    _PUKCL_ZPECRANDOMIZECOORDINATE    PUKCL_ZpEcRandomiseCoordinate;
    _PUKCL_ZPECPOINTISONCURVE         PUKCL_ZpEcPointIsOnCurve;

    // ECC
    _PUKCL_GF2NECCADD                   PUKCL_GF2NEccAdd;
    _PUKCL_GF2NECCDBL                   PUKCL_GF2NEccDbl;
    _PUKCL_GF2NECCMUL                   PUKCL_GF2NEccMul;
    _PUKCL_GF2NECDSAGENERATE            PUKCL_GF2NEcDsaGenerate;
    _PUKCL_GF2NECDSAVERIFY              PUKCL_GF2NEcDsaVerify;
    _PUKCL_GF2NECCONVPROJTOAFFINE       PUKCL_GF2NEcConvProjToAffine;
    _PUKCL_GF2NECCONVAFFINETOPROJECTIVE PUKCL_GF2NEcConvAffineToProjective;
```

```
    _PUKCL_GF2NECRANDOMIZECOORDINATE    PUKCL_GF2NEcRandomiseCoordinate;
    _PUKCL_GF2NECPOINTISONCURVE         PUKCL_GF2NEcPointIsOnCurve;
    } P;
} PUKCL_PARAM,
```

37.3.3.2.1 PUKCL_HEADER Structure

The PUKCL_HEADER is common for all services of the library. This header includes standard fields to indicate the requested service, sub-service, options, return status, and so on, as shown in the following tables.

Different terms used in the below description to be understood, are as follows:

- Parameter – Represents a variable used by the PUKCL. Every parameter belongs to either PUKCL_HEADER or PUKCL Service Specific Header
- Type – Indicates the data type. For details on data type, please refer to `CryptoLib_typedef_pb.h` file in the library
- Dir – Direction. Indicates whether PUKCL considers the variable as input or output. Input means that the application passes data to the PUKCL using the variable. Output means that the PUKCL uses the variable to pass data to the application.
- Location – Suggests whether the parameter need to be stored in Crypto RAM or device SRAM. The PUKCL driver has macros for placing parameters into Crypto RAM, so that the user does not have to worry about the addresses
- Data Length – If a parameter is a pointer variable, the Data Length column shows the size of the data pointed by the pointer

Table 37-1. PUKCL_HEADER Structure

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u1Service	u1	I	–	–	Required service	Executed service
u1SubService	u1	I	–	–	Required sub-service	Executed sub-service
u2Option	u2	I	–	–	Required option	Executed option
Specific	PUKCL_STATUS	I/O	–	–	See the following table PUKCL_STATUS Structure	See the following table PUKCL_STATUS Structure
u2Status	u2	I/O	–	–	–	Output Status
Reserved	u2	–	–	–	–	–
Reserved	u4	–	–	–	–	–

The Specific field in the PUKCL_HEADER structure is another structure named PUKCL_STATUS. The following table describes this structure. The details of the use of these bits are provided in the individual service descriptions.

37.3.3.2.2 PUKCL_STATUS Structure

Members of the PUKCL_STATUS structure are shown in the following table.

Table 37-2. PUKCL_STATUS Structure

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
CarryIn (see **Note 1**)	bit	I	–	–	CarryIn	–
CarryOut	bit	O	–	–	–	CarryOut
Zero	bit	O	–	–	–	1: Result is zero 0: Result is not zero
Gf2n (see **Note 1**)	bit	I	–	–	Mathematical field 0: Integers (Z_p) 1: Field GF(2^n)	–
Violation	bit	O	–	–	–	Indicates a violation

Note:

1. Two of these fields must be filled in to avoid problems during computations. If the Gf2n and CarryIn fields are not reset or initialized properly, problems may be encountered during computations. For instance, not initializing the Gf2n field may result in getting a correct mathematical result, but computed over GF(2^n) instead of Z_p.

37.3.3.2.3 PUKCL Service Specific Header

Details about each service specific header are provided with service descriptions in a subsequent section. Such structures may contain input or output parameters. A parameter is considered as an input parameter when it used for passing information to the PUKCL, and it is considered as an output parameter when the PUKCL uses it to pass a result back to the application code.

The following code provides the service specific header example for the SelfTest service.

```
typedef struct _PUKCL_selftest {
    u4 u4Version;
    u4 u4PUKCCVersion;
    u4 u4CheckNum1;
    u4 u4CheckNum2;
    u1 u1Step;
} _PUKCL_SELFTEST;
```

After the SelfTest service is invoked (with vPUKCL_Process(SelfTest, pvPUKCLParam)), the service specific return values can be checked using pvPUKCLParam.

To check whether the version returned by the PUKCL is correct, the following code can be used.

```
while (pvPUKCLParam->P.PUKCL_SelfTest.u4Version != PUKCL_VERSION);
```

In a similar way, other returns can also be accessed.

37.3.3.3 Parameter Passing (Special Considerations)

Most of the PUKCL services work with memory area and accept pointers and lengths as parameters to define input and output areas. Most of the time, the pointers and lengths are untouched by the services, while the defined areas are read, filled, or overwritten. These memory areas are defined with an initial pointer and a byte length. For most of the commands, the memory area location must be in the PUKCC Cryptographic RAM. The Cryptographic RAM is the memory area for parameter exchange with the PUKCL and is 4 Kbytes large. Sometimes memory areas can be located in Embedded SRAM, which is detailed in the Location column of the parameters description tables.

When working with binary fields, polynomials in GF(2^n) need no transformation to be written in an area:

- Each bit represents a polynomial coefficient 0 or 1
- The polynomials must be written Low Significant Byte First
- A zero padding on the Most Significant Bytes may be added if the area is larger than the real size of the polynomial

Important: The Cryptographic RAM is 4 Kbytes in size and is dedicated to PUKCC. However, to ensure correct library operation, the two last 32-bit words must not be used. Unless otherwise specified, these memory areas contain integers in GF(p) or polynomials in GF(2^n) with the Less Significant Byte first.

Unless otherwise specified, the length must be a multiple of four and the pointers must be four bytes aligned. This is because most of the services work with 32-bit words.

37.3.3.4 Aligned Significant Length

Parameters in memory areas can have any Significant Length in bytes. As the lengths in PUKCL must be a multiple of four, a padding is processed on the Most Significant Side with zero to three bytes cleared to zero. Now the parameter can be considered to meet the Aligned Significant Length requirement for PUKCL.

37.3.3.5 Processing Field GF(p) and GF(2^n)

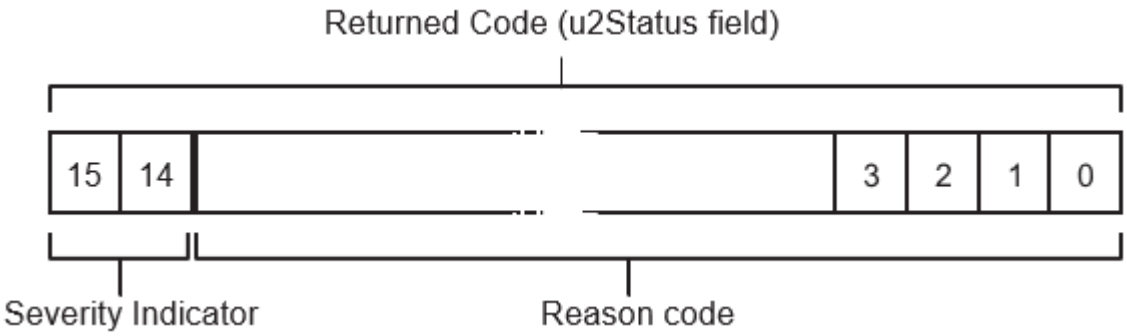
The library can process arithmetic functions over GF(p) (or Zp integers) and GF(2^n), when applicable. The choice of these processing fields is made using the following rules:

- If a processing field is not applicable to the function, it is not mentioned and the Specific.GF2n bit has no effect.
- If the function can support both processing fields, the choice is mentioned and the Specific.GF2n bit must be filled according to the choice.
- If the function supports only one of the processing fields, the processing field is mentioned and the Specific.GF2n bit has no effect.

37.3.3.6 Return Codes

Each call to one of the PUKCL services returns a status code indicating whether or not the execution is correct, which can be decoded, as shown in the following figure.

Figure 37-1. Return Code Status Decoding



The following table shows how the severity indicators must be decoded.

Table 37-3. Severity Indicators

Value for Bits 14–15	Severity	Comment
0xC000	Severe	Indicates a blocking error condition
0x8000	Warning	Indicates a cautionary use of the return values
0x4000	Information	Indicates the result is correct and gives information
0x0000	–	No error or no severity given

The following table contains the exhaustive list of all reason codes.

Table 37-4. Return Codes

Value for Bits 00–13	Severity Code	Reason Code
0x0000	—	PUKCL_OK
0x4001	Informative	PUKCL_NUMBER_IS_NOT_PRIME
0x4002	Informative	PUKCL_NUMBER_IS_PRIME
0xC001	Severe	PUKCL_COMPUTATION_NOT_STARTED
0xC002	Severe	PUKCL_UNKNOWN_SERVICE
0xC003	Severe	PUKCL_UNEXPLOITABLE_OPTIONS
0xC004	Severe	PUKCL_HARDWARE_ISSUE
0xC005	Severe	PUKCL_WRONG_HARDWARE
0xC006	Severe	PUKCL_LIBRARY_MALFORMED
0xC007	Severe	PUKCL_ERROR
0xC008	Severe	PUKCL_UNKNOWN_SUBSERVICE
0xC101	Severe	PUKCL_DIVISION_BY_ZERO

.....continued		
Value for Bits 00–13	**Severity Code**	**Reason Code**
0xC102	Severe	PUKCL_MALFORMED_MODULUS
0xC103	Severe	PUKCL_FAULT_DETECTED
0xC104	Severe	PUKCL_MALFORMED_KEY

Please note the following rules about return codes:

- A status value indicating a severe error, means that an expected operation has not been executed or has been corrupted. Therefore, the result of such an operation must not be used.
- A status value indicating a warning must be looked at precisely, as the expected correctness of the result cannot be guaranteed.
- A status value indicating an information always means that the result is correct with no possible misinterpretation of the values.
- A status value zero indicates that there is no error or no severity.

In the following sections, for each service, the constraints on the parameters placement are detailed. For reduced code size and higher execution speed, tests are processed on these constraints. It is important that PUKCL users take these placement constraints into consideration at the development and test stages to ensure the correct functioning of the library.

37.3.4 Basic Arithmetic and Cryptographic Services

37.3.4.1 SelfTest

37.3.4.1.1 Purpose

This service is used to initialize the PUKCL. It resets the PUKCC, clears the Crypto RAM, and returns the library and PUKCC version numbers.

It must be called before using any other services in the library and the user must verify the return status at the end of the service execution.

37.3.4.1.2 How to Use the Service

37.3.4.1.3 Description

This service processes internal tests and returns information and status codes as described in 37.3.4.1.7. Status Returned Values. The service name for this operation is `SelfTest`.

37.3.4.1.4 Parameters Definition

It is possible to directly address this service through the `PUKCL_SelfTest()` macro.

Table 37-5. SelfTest Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
u4Version	u4	O	–	–	–	PUKCL version
u4PUKCCVersion	u4	O	–	–	–	PUKCC Version
u4CheckNum1	u4	O	–	–	–	Test result value 1
u4CheckNum2	u4	O	–	–	–	Test result value 2
u1Step	u1	O	–	–	–	Latest correctly executed step

37.3.4.1.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library
```

```
vPUKCL_Process(SelfTest,pvPUKCLParam);

if (PUKCL(u2Status) == PUKCL_OK)
            {
            // The Library version is available
            // in PUKCL_SelfTest(u4Version)
            // The PUKCL version is available
            // in PUKCL_SelfTest(u4PUKCCVersion)
            }
```

37.3.4.1.6 Returned Values

The expected u4Version value depends on the version of PUKCL being used, and the u4PUKCCVersion value depends on the version of PUKCC being used.

The expected u4CheckNum1 value is 0x6e70ddd2 and the expected one for u4CheckNum2 is 0x25c8d64f. The expected final u1Step value is 3.

37.3.4.1.7 Status Returned Values

Table 37-6. SelfTest Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly.
PUKCL_ERROR	Severe	An issue has been encountered.

37.3.4.2 Clear Flags

37.3.4.2.1 Purpose

This service can be used to clear parameter structure flags.

37.3.4.2.2 How to Use the Service

37.3.4.2.3 Description

This service clears CarryOut, CarryIn, Zero and Violation flags in the Specific bit field. The Gf2n flag is untouched.

The service name for this operation is `ClearFlags`.

37.3.4.2.4 Parameters Definition

It is possible to directly address this service through the `PUKCL_ClearFlags()` macro.

Table 37-7. Clear Flags Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/CarryOut	Bit	O	–	–	–	Cleared
Specific/CarryIn	Bit	O	–	–	–	Cleared
Specific/Zero	Bit	O	–	–	–	Cleared
Specific/Violation	Bit	O	–	–	–	Cleared

37.3.4.2.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ClearFlags,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            // Success
            }
else // Manage the error
```

37.3.4.2.6 Status Returned Values

Table 37-8. ClearFlags Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly.

37.3.4.3 Swap

37.3.4.3.1 Purpose

This service performs swapping of two buffers.

37.3.4.3.2 How to Use the Service

37.3.4.3.3 Description

This service swaps two buffers, X and Y, of the same size in memory.

The service name for this operation is `Swap`.

37.3.4.3.4 Parameters Definition

This service can easily be accessed through the use of the `PUKCL_Swap()` macro.

Table 37-9. Swap Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1XBase	nu1	I	Crypto RAM	u2Length	Base of the number X	Base of X filled with Y
nu1YBase	nu1	I	Crypto RAM	u2Length	Base of the number Y	Base of Y filled with X
u2XLength	u2	I	–	–	Length of X and Y	Length of X and Y

37.3.4.3.5 Code Example

```
_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Initialize parameters
PUKCL_Swap(nu1XBase) = <Base of the X number>;
PUKCL_Swap(nu1YBase) = <Base of the Y number>;
PUKCL_Swap(u2XLength) = <Length of the numbers>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Swap,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.4.3.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1YBase are not aligned on 32-bit boundaries
- u2XLength is either <4, > 0xffc, or not a 32-bit length
- {nu1XBase, u2XLength} or {nu1YBase, u2XLength} do not entirely lie in PUKCCRAM
- {nu1XBase, u2XLength} overlaps {nu1YBase,u2YLength}

37.3.4.3.7 Status Returned Values

Table 37-10. Swap Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

37.3.4.4 Fill

37.3.4.4.1 Purpose

This service performs a memory fill operation, with a given 32-bit constant.

37.3.4.4.2 How to Use the Service

37.3.4.4.3 Description

This service fills a Crypto RAM space with a provided 32-bit constant: Fill (R, FillValue)

The service name for this operation is `Fill`.

37.3.4.4.4 Parameters Definition

This service can easily be accessed through the use of the `PUKCL_Fill()` macro.

Table 37-11. Fill Service Parameters

Parameter	Type	Direction.	Location	Data Length	Before Executing the Service	After Executing the Service
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R value filled repetitively with u4FillValue
u2RLength	u2	I	Crypto RAM	–	Length of R	Length of R
u4FillValue	u4	I	–	–	Filling value	Filling value

37.3.4.4.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Initialize parameters
PUKCL_Fill(nu1RBase) = <Base of the R number>;
PUKCL_Fill(u2RLength) = <Length of the R number>;
PUKCL_Fill(u4FillValue) = <32-bits value to fill with>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Fill,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.4.4.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1RBase are not aligned on 32-bit boundaries
- u2RLength is either: <4, >0xffc or not a 32-bit length
- {nu1RBase, u2RLength} do not entirely lie in Crypto RAM

37.3.4.4.7 Status Returned Values

Table 37-12. Fill Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly.

37.3.4.5 Fast Copy/Clear

37.3.4.5.1 Purpose

This service performs a copy from a memory area to another or a memory area clear.

37.3.4.5.2 How to Use the Service

37.3.4.5.3 Description

This service copies a number X into another number R, padding with zero on the MSB side up to the length specified for R.

R = X

If the lengths of R and X are equal, a complete fast copy is processed.

If the length of R is strictly greater than the length of X, X is first copied in the Low Significant Bytes side of R, and R is padded with zeros on the Most Significant Bytes side.

If the pointer on the X area equals zero, R is filled with zeros. This operation can also be made by using the Fill service (see 37.3.4.4. Fill).

The service name for this operation is `FastCopy`.

Important: The length of R must be greater or equal to the length of X.

37.3.4.5.4 Parameters Definition

This service can easily be accessed through the use of the `PUKCL_FastCopy()` macro.

Table 37-13. FastCopy Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1XBase	nu1	I	Crypto RAM	u2XLength	Base of X	Base of X number untouched
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R filled with X
u2RLength	u2	I	–	–	Length of R	Length of R
u2XLength	u2	I	–	–	Length of X	Length of X

37.3.4.5.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Initialize parameters
PUKCL_FastCopy(nu1XBase) = <Base of the X number>;
PUKCL_FastCopy(nu1RBase) = <Base of the R number>;
PUKCL_FastCopy(u2XLength) = <Length of the X number>;
PUKCL_FastCopy(u2RLength) = <Length of the R number>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(FastCopy,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.4.5.6 Constraints

The parameter placements that are not allowed are are as follows.

If nu1XBase equals zero, no checks are made on nu1XBase (fixed) and u2XLength (unused).

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1RBase are not aligned on 32-bit boundaries

- u2XLength or u2RLength is either: <4, >0xffc or not a 32-bit length or u2XLength >u2RLength
- {nu1XBase, u2XLength} or {nu1RBase, u2RLength} do not entirely lie in Crypto RAM
- {nu1XBase, u2XLength} overlaps {nu1RBase,u2RLength}

37.3.4.5.7 Status Returned Values

Table 37-14. FastCopy Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

37.3.4.6 Conditional Copy/Clear

37.3.4.6.1 Purpose

This service conditionally performs a copy from a memory area to another or a memory area clear.

37.3.4.6.2 How to Use the Service

37.3.4.6.3 Description

This service copies a number X into another number R, padding with zero on the MSB side up to the length specified for R. This copy operation is performed under the conditions specified in the options.

If the condition is verified, R = X.

The copy or clear action is made under condition.

The four possible options for the condition are described in the following table. Two of the conditions check the Specific.CarryIn bit.

The processing is done as follows:

- If the condition is not verified, nothing is processed.
- If the condition is verified the copy or clear follows the rules:
  - If the lengths of R and X are equal, a complete fast copy is processed
  - If the length of R is strictly greater than the length of X, X is first copied in the Low Significant Bytes side of R, and R is padded with zeros on the Most Significant Bytes side.
  - If the pointer on the X area equals zero, R is filled with zeros.

The service name for this operation is `CondCopy`.

Important: If the condition is verified, the length of R must be greater or equal to the length of X.

37.3.4.6.4 Parameters Definition

This service can easily be accessed through the use of the `PUKCL_CondCopy()` and `PUKCL()` macros.

Table 37-15. CondCopy Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Option for condition (see the following table)	Option for condition (see the following table)
Specific/CarryIn	Bit	I	–	–	Bit CarryIn	Bit CarryIn
nu1XBase	nu1	I	Crypto RAM	u2XLength	Base of X	Base of X number untouched

.....continued						
Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R filled with X if condition holds
u2RLength	u2	I	–	–	Length of R	Length of R
u2XLength	u2	I	–	–	Length of X	Length of X

37.3.4.6.5 Available Options

The option for the condition is set by the u2Options input parameter that must take one of the values listed in the following table.

Table 37-16. CondCopy Service Options

Option	Purpose	Needed parameters
PUKCL_CONDCOPY_ALWAYS	Always perform the copy	nu1XBase,u2XLength,nu1RBase, u2RLength
PUKCL_CONDCOPY_NEVER	Never perform the copy	None
PUKCL_CONDCOPY_IF_CARRY	Perform the copy if CarryIn is 1	Specific/CarryIn nu1XBase,u2XLength,nu1RBase, u2RLength
PUKCL_CONDCOPY_IF_NOT_CARRY	Perform the copy if CarryIn is zero	Specific/CarryIn nu1XBase,u2XLength,nu1RBase, u2RLength

37.3.4.6.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// CarryIn shall be beforehand filled (with zero or one) PUKCL(Specific).CarryIn = ...;

// Condition Option PUKCL(u2Options) = ...;

// Initialize parameters
PUKCL_CondCopy(nu1XBase) = <Base of the X number>;
PUKCL_CondCopy(nu1RBase) = <Base of the R number>;
PUKCL_CondCopy(u2XLength) = <Length of the X number>;
PUKCL_CondCopy(u2RLength) = <Length of the R number>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(CondCopy,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.4.6.7 Constraints

The parameters placement that are not allowed are listed below.

If the conditional option and the CarryIn do not lead to execute the copy, no checks are made on the constraints to be respected.

If nu1XBase equals zero, no checks are made on nu1XBase (fixed) and u2XLength (unused).

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1RBase are not aligned on 32-bit boundaries
- u2XLength or u2RLength is either: <4, >0xffc or not a 32-bit length or u2XLength >u2RLength

- {nu1XBase, u2XLength} or {nu1RBase, u2RLength} do not entirely lie in Crypto RAM
- {nu1XBase, u2XLength} overlaps {nu1RBase,u2RLength}

37.3.4.6.8 Status Returned Values

Table 37-17. CondCopy Service Return Codes

Returned status	Importance	Meaning
PUKCL_WRONG_SERVICE	Severe	An inconsistency has been detected between the called service and the provided service number.
PUKCL_OK	–	Service functioned correctly

37.3.4.7 Small Multiply, Add, Subtract, Exclusive OR

Related Links

37.3.4.5. Fast Copy/Clear

37.3.5.1. Modular Reduction

37.3.4.7.1 Purpose

This purpose of this service is to multiply a large number X by a single-word number, MulValue, and perform an optional accumulation/subtract with a large number Z, returning the result R.

The following options are available:

- Work in the GF(2^n) or in the standard GF(p) arithmetic integer field
- Add of a supplemental CarryOperand
- Overlap of the operands is possible, taking into account some constraints
- Modulo-reduction of the computation result (see *Modular Reduction* from Related Links)

In addition to a multiply, possible uses of this service can include:

- Copy a block of data from one place to another (if u4MulValue is 1). This operation can alternatively be made by using the Fast Copy service (see *Fast Copy/Clear* from Related Links)
- Adding/Subtracting two numbers (if u4MulValue is1)
- Xoring two blocks of data (if u4MulValue is 1 and the selected mathematical field is GF(2^n))

37.3.4.7.2 How to Use the Service

37.3.4.7.3 Description

This service processes the following operation (if not computing a modular reduction of the result):

$R = [Z] \pm (MulValue \times X + CarryOperand)$

Or (if computing a modular reduction of the result):

$R = ([Z] \pm (MulValue \times X + CarryOperand)) mod\ N$

The service name for this operation is `Smult`.

The result of the Small Multiply Operation is stored on u2RLength bytes, so the choice of this length compared to u2XLength may lead to:

- A truncation if the result is too big to be stored on u2RLengthbytes.
- A padding on the MSB side if the result does not take all the u2RLengthbytes.
  However, in all cases this rule must be followed:

Important: The length of R must be greater than or equal to the length of X.

In these computations, the following parameters need to be provided:

- R the result (pointed by{nu1RBase,u2Rlength})
- X one input number or GF(2^n) polynomial (pointed by{nu1XBase,u2XLength})
- Z one optional input number or GF(2^n) polynomial (pointed by{nu1ZBase,u2Rlength}).
- MulValue one input number or GF(2^n)polynomial on one word (provided in u4MulValue)
- CarryOperand (provided through the CarryOptions and Carry values).

Important: Even if neither accumulation nor subtraction is specified, the nu1ZBase must always be filled and point to a Crypto RAM space. It this case, nu1ZBase can point to the same space as the nu1RBase.

If using the modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links) and the following parameters must be additionally provided:

- N—the modulus (pointed by {nu1ModBase,u2Modlength +4})
- Cns—the reduction constant
  - In case of Big reduction, Cns is pointed by {nu1CnsBase,64bytes}.
  - In case of Fast or Normalized reduction, Cns is pointed by {nu1CnsBase,u2ModLength +8}

Important:
The result buffer R must first be padded with zero bytes until its length is sufficient to perform the reduction (2*u2ModLength + 8) to be used by the Modular Reduction service as an input parameter.

The result of the reduction is written in the area X pointed by {nu1XBase, u2ModLength + 4}.

- For example, if relevant u2ModLength is 0x80 bytes and u2XLength is 0x80 too, the length of the Rspace may be 2*(u2ModLength + 4) = 0x108 bytes.
  In case of fast or normalized reduction, the length of the result may be u2ModLength + 4 so 0x84 bytes.
  Therefore, the zone X may lengths 0x84 bytes (at least). The multiplication of X by 1 word provide a result in the zone R which MSB bytes will be padded with zero bytes.

  In that example, the length of the zone R will be 2*u2ModLength + 8 = 0x108 bytes.

37.3.4.7.4 Parameters Definition

Table 37-18. Smult Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	—	—	Options (see below)	Options (see below)
Specific/Gf2n CarryIn	Bits	I	—	—	GF(2^n) Bit and Carry In	—
Specific/CarryOut Zero Violation	Bits	I	—	—	—	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	—	—	Length of N	Length of N

.....continued						
Parameter	**Type**	**Direction**	**Location**	**Data Length**	**Before Executing the Service**	**After Executing the Service**
nu1XBase	nu1	I	Crypto RAM	u2XLength or u2ModLength + 4(1)	Base of X	Base of X(2)
u2XLength	u2	I	—	—	Length of X	Length of X
nu1ZBase	nu1	I	Crypto RAM	u2RLength	Base of Z	Base of Z untouched
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R (see **Note 3**)
u2RLength	u2	I	—	—	Length of R	Length of R
u4MulValue	u4	I	—	—	Value of MulValue	Value of MulValue untouched

Notes:

1. If a reduction option is specified, the area X will be, if necessary, extended to u2ModLength + 4 bytes.
2. If Smult is without reduction, X is untouched. If Smult is with reduction, X is filled with the final result.
3. If Smult is without reduction, R is filled with the final result. If Smult is with reduction, R is corrupted.

37.3.4.7.5 Available Options

The options are set by the u2Options input parameter, which is composed of:

- The mandatory Small Multiplication operation option described in the following table.
- The mandatory CarryOperand option described in Smult Service (with Accumulate/Subtract From) Carry Settings and Smult Service Carry Settings tables.
- The facultative Modular Reduction option (see Modular Reduction). If the Modular Reduction is not requested, this option is absent.

The u2Options number is calculated by an “Inclusive OR” of the options. Some examples in C language are:

- Operation: Small Multiply only without carry and without Modular Reduction
```
PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_SMULT_ONLY) |
SET_CARRYOPTION(CARRY_NONE);
```
- Operation: Small Multiply with addition with Specific/CarryIn addition and with Fast Modular Reduction
```
PUKCL(u2Options) =SET_MULTIPLIEROPTION(PUKCL_SMULT_ADD) |
SET_CARRYOPTION(ADD_CARRY) | PUKCL_REDMOD_REDUCTION | PUKCL_REDMOD_USING_FASTRED;
```

The following table lists all of the necessary parameters for the Small Multiply option. When the Addition or Subtraction option is not chosen, it is not necessary to fill in the nu1ZBase parameter.

Table 37-19. Smult Service Operation Options

Option	Purpose	Required Parameters
SET_MULTIPLIEROPTION(PUKCL_SMULT_ONLY)	Perform R = MulValue*X + CarryOperand	nu1RBase, u2RLength, nu1XBase, u2XLength, u4MulValue
SET_MULTIPLIEROPTION(PUKCL_SMULT_ADD)	Perform R = Z + MulValue*X + CarryOperand	nu1RBase, u2RLength, nu1ZBase, nu1XBase, u2XLength,u4MulValue
SET_MULTIPLIEROPTION(PUKCL_SMULT_SUB)	Perform R = Z - (MulValue*X + CarryOperand)	nu1RBase, u2RLength, nu1ZBase, nu1XBase, u2XLength,u4MulValue

37.3.4.7.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Gf2n and CarryIn shall be beforehand filled (with zero or one)
PUKCL(Specific).Gf2n = ...;
PUKCL(Specific).CarryIn = ...; PUKCL(u2Options) =...;

// Depending on the option specified, all fields must not be filled
PUKCL_Smult(nu1XBase) = <Base of the X number>;
PUKCL_Smult(u2XLength) = <Length of the X number>;
PUKCL_Smult(nu1RBase) = <Base of the R number>;
PUKCL_Smult(u2RLength) = <Length of the R number>;
PUKCL_Smult(nu1ZBase) = <Base of the Z number>;
PUKCL_Smult(u4MulValue) = <Value to be multiplied with>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Smult,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            // The Small multiplication has been executed correctly
            ...
            }
else // Manage the error
```

Note:
The length of R must be greater or equal to the length of X. Additional options are available through the use of a modular reduction to be executed at the end of this operation. Some important considerations have to be taken into account concerning the length of resulting operands to get a mathematically correct result.

The output of this operation is not obviously compatible with the modular reduction, as it may be either smaller or bigger. In the case (most of the time) where the result (pointed by nu1RBase) is smaller in size than twice the modulus plus one word, it is mandatory to add padding bytes to zero. Otherwise, the reduced value will be taken considering the high order words (potentially uninitialized) as part of the number, thus resulting in a mathematically correct but unexpected result.

In the case that the result is bigger than twice the modulus plus one word, the modular reduction feature has to be executed as a separate operation, using an Euclidean division.

37.3.4.7.7 Constraints

For the case of a small multiplication with an option indicating either subtraction or accumulation, the following conditions must be avoided to ensure the service works correctly:

- nu1XBase, nu1RBase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength}, {nu1ZLength, u2RLength} or {nu1RBase, u2RLength} do not entirely lie in Crypto RAM
- u2XLength or u2RLength is either: < 4, > 0xffc or not a 32-bit length or u2XLength >u2RLength
- {nu1RBase, u2RLength} overlaps {nu1XBase, u2XLength} or nu1R < nu1Z and {nu1RBase,u2RLength} overlaps {nu1ZBase, u2RLength}

If the nu1R value is greater or equals to the nu1Z one, the overlapping between R and Z is allowed.

If a modular reduction is specified, the relevant parameters must be defined according to the chosen reduction and follow the description in Modular Reduction. Additional constraints to be respected and error codes are described in this section and in Smult Service Return Codes.

Multiplication with Accumulation or Subtraction
When the options bits specify that either an Accumulation or a Subtraction must be performed, this service performs the following operation:

$R = (Z \pm (MulValue \times X + CarryOperand)) mod\ B^{RLength}$

Table 37-20. Smult Service (with Accumulate/Subtract From) Carry Settings

Carry Options	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	R = Z ± (MulValue*X + CarryIn)

.....continued

Carry Options	CarryOperand	Resulting Operation
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	R = Z ± (MulValue*X - CarryIn)
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	R = Z ± (MulValue*X + 1 + CarryIn)
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	R = Z ± (MulValue*X + 1 - CarryIn)
SET_CARRYOPTION(CARRY_NONE)	0	R = Z ± (MulValue*X)
SET_CARRYOPTION(ADD_1)	1	R = Z ± (MulValue*X + 1)
SET_CARRYOPTION(SUB_1)	- 1	R = Z ± (MulValue*X - 1)
SET_CARRYOPTION(ADD_2)	2	R = Z ± (MulValue*X + 2)

Multiplication without Accumulation or Subtraction

When the case the options bits specify that neither an Accumulation nor a Subtraction must be performed, this service performs the following operation:

$R = (MulValue \times X + CarryOperand) mod\ B^{RLength}$

Table 37-21. Smult Service Carry Settings

Carry Options	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	R = MulValue*X + CarryIn
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	R = MulValue*X - CarryIn
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	R = MulValue*X + 1 + CarryIn
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	R = MulValue*X + 1 - CarryIn
SET_CARRYOPTION(CARRY_NONE)	0	R = MulValue*X
SET_CARRYOPTION(ADD_1)	1	R = MulValue*X + 1
SET_CARRYOPTION(SUB_1)	-1	R = MulValue*X - 1
SET_CARRYOPTION(ADD_2)	2	R = MulValue*X + 2

37.3.4.7.8 Status Returned Values

Table 37-22. Smult Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	—	Service functioned correctly

37.3.4.8 Compare

37.3.4.8.1 Purpose

The purpose of this service is to compare two numbers in classical arithmetic GF(p).

Important: This service works only with integers.

37.3.4.8.2 How to Use the Service

37.3.4.8.3 Description

This service accepts two numbers in classical arithmetic in input and performs a comparison, virtually subtracting (X + CarryIn) from Y:

CompareGetFlags (Y - (X + CarryIn))

The numbers X and Y are untouched but the resulting flags CarryOut and the Zero Bit are filled. If the lengths of Y and X are equal, a comparison is processed.

If the length of Y is strictly greater than the length of X, X is first virtually padded with zeros on the Most Significant Bytes side, then a comparison is processed.

Note: The length of Y must be greater or equal to the length of X.

In this computation, the following data need to be provided:

- X (pointed by{nu1XBase,u2XLength})
- Y (pointed by{nu1YBase,u2YLength})

The service name for this operation is `Comp`.

37.3.4.8.4 Parameters Definition

Table 37-23. Comp Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/Gf2n CarryIn	Bits	I	–	–	GF(2n) Bit and Carry In	–
Specific/CarryOut Zero Violation	Bits	I	–	–	–	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1XBase	nu1	I	Crypto RAM	u2XLength	Base of X	Base of X
u2XLength	u2	I	–	–	Length of X	Length of X
nu1YBase	nu1	I	Crypto RAM	u2YLength	Base of Y	Base of Y
u2YLength	u2	I	–	–	Length of Y	Length of Y

37.3.4.8.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// CarryIn shall be beforehand filled (with zero or one) PUKCL(Specific).CarryIn = ...;

// Initializing parameters
PUKCL_Comp(nu1XBase) = <Base of the ram location of X>;
PUKCL_Comp(u2XLength) = <Length of X>;
PUKCL_Comp(nu1YBase) = <Base of the ram location of Y>;
PUKCL_Comp(u2YLength) = <Length of Y>;

// vPUKCL_Process() is a macro command,
// and then calls the library...
vPUKCL_Process(Comp,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            // The COMPARE has been executed correctly
            // CarryOut, Zero ... are available
            ... = PUKCL(Specific).CarryOut;
            ... = PUKCL(Specific).Zero;
            }
else // Manage the error
```

37.3.4.8.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase or nu1YBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength} or {nu1YLength, u2YLength} are not in Crypto RAM
- u2XLength or u2YLength is either: < 4, > 0xffc or not a 32-bit length or u2XLength >u2YLength

37.3.4.8.7 Status Returned Values

Table 37-24. Comp Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

37.3.4.9 Full Multiply

Related Links

37.3.5.1. Modular Reduction

37.3.4.9.1 Purpose

The purpose of this service is to multiply two large numbers, X and Y, and optionally accumulate/subtract from a third large number, Z, returning the result, R.

The available options are as follows:

- Work in the GF(2^n) field or in the standard arithmetic field
- Add of a supplemental CarryOperand
- Overlap of the operands is possible, taking into account some constraints
- Modular Reduction of the computation result (see *Modular Reduction* from Related Links)

37.3.4.9.2 How to Use the Service

37.3.4.9.3 Description

This service provides the following (if not computing a modular reduction of the result):

$R = [Z] \pm (X \times Y + CarryOperand)$

Or (if computing a modular reduction of the result):

$R = ([Z] \pm (X \times Y + CarryOperand)) mod\ N$

The service name for this operation is `Fmult`.

In these computations, the following data has to be provided:

- R the result (pointed by {nu1RBase,u2Xlength +u2YLength})
- X one input number or GF(2^n) polynomial (pointed by{nu1XBase,u2XLength})
- Y one input number or GF(2^n) polynomial (pointed by{nu1YBase,u2YLength})
- Z one optional input number or GF(2n) polynomial (pointed by {nu1ZBase,u2Xlength +u2YLength})
- CarryOperand (provided through the Carry Options and Carry values)

Important: Even if neither accumulation nor subtraction is specified, the nu1ZBase must always be filled and point to a Crypto RAM space. It this case, nu1ZBase can point to the same space as the nu1RBase.

If using the big modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case, the length of Cns is 64 bytes.

If using the modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case the following parameters must be additionally provided:

- N—the modulus (pointed by {nu1ModBase,u2Modlength +4})

- Cns—the reduction constant
  - In case of Big reduction, Cns is pointed by {nu1CnsBase,64bytes}.
  - In case of Fast or Normalized reduction, Cns is pointed by (pointed by {nu1CnsBase,u2ModLength+ 8})

Note:
The result buffer R must first be padded with zero bytes until its length is sufficient to perform the reduction (2*u2ModLength + 8) to be used by the Modular Reduction service as an input parameter.

The result of the reduction is written in the area X pointed by {nu1XBase, u2ModLength + 4}.

For example, if u2ModLength, u2XLength and u2YLength are 0x80 bytes, the length of the R space is 2*(u2ModLength + 4) = 0x108 bytes because of the constraints of modular reduction.

In case of Fast or Normalized Reduction, the length of the result is u2ModLength + 4 so 0x84 bytes. Thus, the zone X has a length of 0x84 bytes (at least). The multiplication of X by Y provides a result of length 0x100 bytes in the zone R so the 8 MSB bytes must be previously padded with zero bytes (in offsets 0x100 to 0x107).

37.3.4.9.4 Parameters Definition

Table 37-25. Fmult Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Options (see below)	Options (see below)
Specific/Gf2n CarryIn	Bits	I	–	–	GF(2n) Bit and Carry In	–
Specific/CarryOut Zero Violation	Bits	I	–	–	–	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8 or 64 bytes	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	–	–	Length of N	Length of N
nu1XBase	nu1	I	Crypto RAM	u2XLength or u2ModLength + 4(1)	Base of X	Base of X(2)
u2XLength	u2	I	–	–	Length of X	Length of X
nu1YBase	nu1	I	Crypto RAM	u2YLength	Base of Y	Base of Y
u2YLength	u2	I	–	–	Length of Y	Length of Y
nu1ZBase	nu1	I	Crypto RAM	u2XLength + u2YLength	Base of Z	Base of Z untouched
nu1RBase	nu1	I	Crypto RAM	u2XLength + u2YLength	Base of R	Base of R(3)

Notes:
1. In case of a reduction option is specified, if necessary, the area X will be extended to u2ModLength + 4 bytes.
2. If FMult is without reduction, X is untouched. If FMult is with reduction, X is filled with the final result.
3. If FMult is without reduction, R is filled with the final result. If FMult is with reduction, R is corrupted.

37.3.4.9.5 Available Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Full Multiplication operation option described in Table 37-26
- the mandatory CarryOperand option described in Table 37-27 and Table 37-28
- the facultative Modular Reduction option(see *Modular Reduction* from Related Links). If the Modular Reduction is not requested, this option is absent.

The u2Options number is calculated by an Inclusive OR of the options.

Some Examples in C language are:

- Operation: Full Multiply only without carry and without Modular Reduction
  ```
  PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_FMULT_ONLY) |
  SET_CARRYOPTION(CARRY_NONE);
  ```
- Operation: Full Multiply with addition with Specific/CarryIn addition and with Fast Modular Reduction
  ```
  PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_FMULT_ADD) |

  SET_CARRYOPTION(ADD_CARRY) |

  PUKCL_REDMOD_REDUCTION |

  PUKCL_REDMOD_USING_FASTRED;
  ```

The following table shows all of the necessary parameters for the Full Multiply option. When the Addition or Subtraction option is not chosen, it is not necessary to fill in the nu1ZBase parameter.

Table 37-26. Fmult Service Options

Option	Purpose	Required Parameters
SET_MULTIPLIEROPTION(PUKCL_FMUL_ONLY)	Perform R = X*Y + CarryOperand	nu1RBase, nu1YBase, u2YLength, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_FMUL_ADD)	Perform R = Z + X*Y + CarryOperand	nu1RBase, nu1ZBase, nu1YBase, u2YLength, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_FMUL_SUB)	Perform R = Z - (X*Y + CarryOperand)	nu1RBase, nu1ZBase, nu1YBase, u2YLength, nu1Xlength, u2XLength

37.3.4.9.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


// Gf2n and CarryIn shall be beforehand filled (with zero or one)
PUKCL(Specific).Gf2n = ...;
PUKCL(Specific).CarryIn = ...;

PUKCL(u2Option) =...;
// Depending on the option specified, not all fields must be filled
PUKCL_Fmult(nu1XBase) = <Base of the ram location of X>;
PUKCL_Fmult(u2XLength) = <Length of X>;
PUKCL_Fmult(nu1YBase) = <Base of the ram location of Y>;
PUKCL_Fmult(u2YLength) = <Length of Y>;
PUKCL_Fmult(nu1ZBase) = <Base of the ram location of Z>;
PUKCL_Fmult(nu1RBase) = <Base of the ram location of R>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Fmult,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            // The Full multiply has been executed correctly
            ...
            }
else // Manage the error
```

37.3.4.9.7 Important Considerations for Modular Reduction of a Fmult Computation Result

Note:
Additional options are available through the use of a modular reduction to be executed at the end of this operation. Some important considerations have to be taken into account concerning the length of resulting operands to get a mathematically correct result.

The output of this operation is not always compatible with the modular reduction as it may be either smaller or bigger. In the case (most of the time) the result (pointed by nu1RBase) is smaller in size than “twice the modulus plus one word” by one word, a padding word must be added to zero. Otherwise, the reduced value will be taken considering the high order words (potentially uninitialized) as part of the number, thus resulting in getting a mathematically correct but unexpected result.

In the case that the result is bigger than twice the modulus plus one word, the modular reduction feature has to be executed as a separate operation, using an Euclidean division.

37.3.4.9.8 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase, nu1YBase, nu1RBase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength}, {nu1YLength, u2YLength}, {nu1ZBase, u2XLength+u2YLength} or{nu1RBase, u2XLength+u2YLength} are not in Crypto RAM
- u2XLength, u2YLength is either: < 4, > 0xffc or not a 32-bit length
- {nu1RBase, u2XLength+u2YLength} overlaps {nu1YBase, u2YLength} or{nu1RBase, u2XLength+u2YLength} overlaps {nu1XBase, u2XLength}
- {nu1RBase, u2XLength+u2YLength} overlaps {nu1ZBase, u2XLength+u2YLength} and nu1RBase> nu1ZBase

If a modular reduction is specified, the relevant parameters must be defined according to the chosen reduction and follow the description in Modular Reduction (see *Modular Reduction* from Related Links). Additional constraints to be respected and error codes are described in this section and in Table 37-49.

Multiplication with Accumulation or Subtraction

In the case where the options bits specify that either an Accumulation or a subtraction must be performed, this service performs the following operation:

$R = (Z \pm (X \times Y + CarryOperand)) mod\ B^{XLength + YLength}$

Table 37-27. Fmult Service (with Accumulate/Subtract From) Carry Settings

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	R = Z ± (X*Y + CarryIn)
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	R = Z ± (X*Y - CarryIn)
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	R = Z ± (X*Y + 1 + CarryIn)
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	R = Z ± (X*Y + 1 - CarryIn)
SET_CARRYOPTION(CARRY_NONE)	0	R = Z ± (X*Y)
SET_CARRYOPTION(ADD_1)	1	R = Z ± (X*Y + 1)
SET_CARRYOPTION(SUB_1)	- 1	R = Z ± (X*Y - 1)
SET_CARRYOPTION(ADD_2)	2	R = Z ± (X*Y + 2)

Multiplication without Accumulation or Subtraction

In the case the options bits specify that either an Accumulation or a subtraction must be performed, this service performs the following operation:

$R = (X \times Y + CarryOperand) mod\ B^{XLength + YLength}$

Table 37-28. Fmult Service Carry Settings

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	R = X*Y + CarryIn
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	R = X*Y - CarryIn
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	R = X*Y + 1 + CarryIn
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	R = X*Y + 1 - CarryIn
SET_CARRYOPTION(CARRY_NONE)	0	R = X*Y
SET_CARRYOPTION(ADD_1)	1	R = X*Y + 1
SET_CARRYOPTION(SUB_1)	- 1	R = X*Y - 1
SET_CARRYOPTION(ADD_2)	2	R = X*Y + 2

37.3.4.9.9 Status Returned Values

Table 37-29. Fmult Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

37.3.4.10 Square

Related Links

37.3.5.1. Modular Reduction

37.3.4.10.1 Purpose

The purpose of this service is to compute the square of a big number and optionally accumulate/subtract from a second big number.

Please note that this service uses an optimized implementation of the squaring. It also means that when the GF(2^n) flag is set, the execution time will be smaller than when not set (in that case, the squaring execution time will still be smaller than for a standard multiplication).

The available options are as follows:

- Work in the GF(2^n) or in the standard integer arithmetic field
- Add of a supplemental CarryOperand
- Overlapping of the operands is possible, taking into account some constraints
- Modular Reduction of the computation result

37.3.4.10.2 How to Use the Service

37.3.4.10.3 Description

This service provides the following (if not computing a modular reduction of the result):

$R = [Z] \pm (X^2 + CarryOperand)$

Or (if computing a modular reduction of the result):

$R = ([Z] \pm (X^2 + CarryOperand))mod\ N$

The service name for this operation is `Square`.

In these computations, the following data has to be provided:

- R the result (pointed by {nu1RBase,2 *u2Xlength})
- X one input number or GF(2n) polynomial (pointed by{nu1XBase,u2XLength})
- Z one optional input number or GF(2n) polynomial (pointed by {nu1ZBase,2 *u2Xlength})
- CarryOperand (provided through the CarryOptions and Carry values)

Important: Even if neither accumulation nor subtraction is specified, the nu1ZBase must always be filled and point to a Crypto RAM space. It this case, nu1ZBase can point to the same space as the nu1RBase.

If using the big modular reduction option, the Multiply operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case, the length of Cns is 64 bytes.

If using the modular reduction option the Square operation is followed by a reduction (see *Modular Reduction* from Related Links). In this case the following parameters must be additionally provided:

- N—the modulus (pointed by {nu1ModBase,u2Modlength +4}).
- Cns—the reduction constant (pointed by {nu1CnsBase,u2Modlength +8})
  - In case of big reduction option, the length of Cns is 64bytes.

Note:
The result buffer R must first be padded with zero bytes until its length is sufficient to perform the reduction (2*u2ModLength + 8) to be used by the Modular Reduction service as an input parameter.

The result of the reduction is written in the area X pointed by {nu1XBase, u2ModLength + 4}.

For example, if u2ModLength, u2XLength is 0x80 bytes, the length of the R space is 2*(u2ModLength + 4) = 0x108 bytes because of the constraints of modular reduction.

In case of Fast or Normalized Reduction, the length of the result is u2ModLength + 4 so 0x84 bytes. Thus, the zoneX has a length of 0x84 bytes (at least). The square of X provides a result of length 0x100 bytes in the zone R so the 8 MSB bytes previously must be previously padded with zero bytes (in offsets 0x100 to 0x107).

37.3.4.10.4 Parameters Definition

Table 37-30. Square Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Options (see below)	Options (see below)
Specific/Gf2n CarryIn	Bits	I	–	–	GF(2n) Bit and Carry In	–
Specific/CarryOut Zero Violation	Bits	I	–	–	–	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8 or 64 bytes	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	–	–	Length of N	Length of N
nu1XBase	nu1	I	Crypto RAM	u2XLength or u2ModLength + 4(1)	Base of X	Base of X(2)
u2XLength	u2	I	–	–	Length of X	Length of X
nu1ZBase	nu1	I	Crypto RAM	2 * u2XLength	Base of Z	Base of Z
nu1RBase	nu1	I	Crypto RAM	2 * u2XLength	Base of R	Base of R(3)

Notes:

1. In case of a reduction option is specified, the area X will be, if necessary, extended to u2ModLength + 4 bytes.
2. If Square is without reduction, X is untouched. If Square is with reduction, X is filled with the final result.
3. If Square is without reduction, R is filled with the final result. If Square is with reduction, R is corrupted.

37.3.4.10.5 Available Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Square operation option described in Table 37-31
- the mandatory CarryOperand option described in Table 37-32 and Table 37-33
- the facultative Modular Reduction option (see *Modular Reduction* from Related Links). If the Modular Reduction is not requested, this option is absent.

The u2Options number is calculated by an Inclusive OR of the options. Some Examples in C language are:

- Operation: Square only without carry and without Modular Reduction
  ```
  PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_SQUARE_ONLY) |
  SET_CARRYOPTION(CARRY_NONE);
  ```
- Operation: Square with addition with Specific/CarryIn addition and with Fast Modular Reduction
  ```
  PUKCL(u2Options) = SET_MULTIPLIEROPTION(PUKCL_SQUARE_ADD) |
  SET_CARRYOPTION(ADD_CARRY) | PUKCL_REDMOD_REDUCTION | PUKCL_REDMOD_USING_FASTRED;
  ```

The following table lists all of the necessary parameters for the Square option. When the Addition or Subtraction option is not chosen it is not necessary to fill in the nu1ZBase parameter.

Table 37-31. Square Service Options

Option	Purpose	Required Parameters
SET_MULTIPLIEROPTION(PUKCL_SQUARE_ONLY)	Perform R = X^2 + CarryOperand	nu1RBase, nu1ZBase, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_SQUARE_ADD)	Perform R = Z + X^2 + CarryOperand	nu1RBase, nu1ZBase, nu1XBase, u2XLength
SET_MULTIPLIEROPTION(PUKCL_SQUARE_SUB)	Perform R = Z - (X^2 + CarryOperand)	nu1RBase, nu1ZBase, nu1Xlength, u2XLength

37.3.4.10.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


// Gf2n and CarryIn shall be beforehand filled (with zero or one)
PUKCL(Specific).Gf2n = ...;
PUKCL(Specific).CarryIn = ...;

PUKCL(u2Option) =...;
// Depending on the option specified, not all fields must be filled
PUKCL_Fmult(nu1XBase) = <Base of the ram location of X>;
PUKCL_Fmult(u2XLength) = <Length of X>;
PUKCL_Fmult(nu1ZBase) = <Base of the ram location of Z>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Square,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            // The Squaring has been executed correctly
            ...
            }
else // Manage the error
```

37.3.4.10.7 Important Considerations for Modular Reduction of a Square Computation

Note:
Additional options are available through the use of a modular reduction to be executed at the end of this operation. Some important considerations have to be taken into account concerning the length of resulting operands to get a mathematically correct result.

The output of this operation is not obviously compatible with the modular reduction as it may be either smaller or bigger. In the case (most of the time) the result (pointed by nu1RBase) is smaller in size than “twice the modulus plus one word” by one word, a padding word must be added to zero. Otherwise, the reduced value will be taken considering the high order words (potentially uninitialized) as part of the number, thus resulting in getting a mathematically correct but unexpected result.

In the case that the result is greater than twice the modulus plus one word, the modular reduction feature has to be executed as a separate operation, using an Euclidean division.

37.3.4.10.8 Constraints

When the options only indicate a square, the constraints involving nu1ZBase are not checked. The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase, nu1RBase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2XLength}, {nu1ZBase, 2*u2XLength} or {nu1RBase, 2*u2XLength} are not in Crypto RAM
- u2XLength is either: < 4, > 0xffc or not a 32-bit length
- {nu1RBase, 2*u2XLength} overlaps {nu1XBase,u2XLength}
- {nu1RBase, 2*u2XLength} overlaps {nu1ZBase, 2*u2XLength} and nu1RBase >nu1ZBase

If a modular reduction is specified, the relevant parameters must be defined according to the chosen reduction and follow the description in Modular Reduction (see *Modular Reduction* from Related Links). Additional constraints to be respected and error codes are described in this section and in Table 37-49.

Multiplication with Accumulation or Subtraction

Where the options bits specify that either an Accumulation or a subtraction must be performed, this command performs the following operation:

$R = (Z \pm (X^2 + CarryOperand)) mod\ B^{2 \times XLength}$

Table 37-32. Multiplication with Accumulation or Subtraction

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	$R = Z \pm (X^2 + CarryIn)$
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	$R = Z \pm (X^2 - CarryIn)$
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	$R = Z \pm (X^2 + 1 + CarryIn)$
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	$R = Z \pm (X^2 + 1 - CarryIn)$
SET_CARRYOPTION(CARRY_NONE)	0	$R = Z \pm (X^2)$
SET_CARRYOPTION(ADD_1)	1	$R = Z \pm (X^2 + 1)$
SET_CARRYOPTION(SUB_1)	- 1	$R = Z \pm (X^2 - 1)$
SET_CARRYOPTION(ADD_2)	2	$R = Z \pm (X^2 + 2)$

37.3.4.10.9 Multiplication without Accumulation or Subtraction

Where the options bits specify that either an accumulation or a subtraction must be performed, this command performs the following operation:

$R = (X^2 + CarryOperand) mod\ B^{2 \times XLength}$

Table 37-33. Square Service Carry Settings

Option AND CARRYOPTIONS	CarryOperand	Resulting Operation
SET_CARRYOPTION(ADD_CARRY)	CarryIn	$R = X^2 + CarryIn$

.....continued		
Option AND CARRYOPTIONS	**CarryOperand**	**Resulting Operation**
SET_CARRYOPTION(SUB_CARRY)	- CarryIn	R = X^2 - CarryIn
SET_CARRYOPTION(ADD_1_PLUS_CARRY)	1 + CarryIn	R = X^2 + 1 + CarryIn
SET_CARRYOPTION(ADD_1_MINUS_CARRY)	1 - CarryIn	R = X^2 + 1 - CarryIn
SET_CARRYOPTION(CARRY_NONE)	0	R = X^2
SET_CARRYOPTION(ADD_1)	1	R = X^2 + 1
SET_CARRYOPTION(SUB_1)	- 1	R = X^2 - 1
SET_CARRYOPTION(ADD_2)	2	R = X^2 + 2

37.3.4.10.10 Status Returned Values

Table 37-34. Square Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

37.3.4.11 Integral (Euclidean) Division

37.3.4.11.1 Purpose

The purpose of this service is to compute the Euclidean Division of two multiple precision numbers in GF(p) or polynomial in GF(2^n). The Numerator is divided by the Denominator giving the Quotient “Quo” and the Remainder “R”.

The following options are available:

- Work in the GF(2^n) field or in the standard integer arithmetic field GF(p)

37.3.4.11.2 How to Use the Service

37.3.4.11.3 Description

This service processes the calculus corresponding to:

$$Num = Mod \times Quo + R \;\; with \;\; 0 \leq R < Mod \;\; and \;\; Quo = \left[\frac{Num}{Mod}\right]$$

The Numerator is Num.

The Divisior (Modulus) is Mod.

The Quotient is Quo.

The Remainder is R.

The Inputs are, the Numerator Num, and the Denominator Mod. The service calculates the Quotient and the Remainder. The Remainder overwrites the Numerator and is copied to the R area.

If the parameter nu1QuoBase equals zero, the Quotient is not stored in memory.

If nu1QuoBase is different from zero, the Quotient length is (<Numerator Length> - <Denominator Length>) + 4 bytes.

In this computation, the following areas need to be provided:

- Num (pointed by {nu1NumBase,u2NumLength}) filled with the Numerator (with MSB word to zero).
- Mod (pointed by {nu1ModBase,u2ModLength}) filled with the Denominator.
- Workspace (pointed by {nu1CnsBase,64 or68}).
- Quo (pointed by {nu1QuoBase,u2NumLength - u2ModLength + 4}) to contain calculated Quotient.
  - When the quotient is not needed, the nu1QuoBase pointer can be provided as NULL. In that case, only the remainder will be provided as a result.

- R (pointed by {nu1RBase,u2ModLength}) to contain the calculated Remainder.

The service name for this operation is `Div`.

37.3.4.11.4 Parameters Definition

Table 37-35. Div Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/Gf2n	Bit	I	–	–	GF(2n) Bit	–
nu1NumBase	nu1	I	Crypto RAM	u2NumLength	Base of Num Filled with the Numerator	Base of Num Filled with the Remainder
u2NumLength	u2	I	–	–	Length of the Numerator	Length of the Numerator
nu1ModBase	nu1	I	Crypto RAM	u2ModLengt	Base of the Divisor	Base of the Divisor untouched
u2ModLength	u2	I	–	–	Length of the Divisor	Length of the Divisor
nu1QuoBase (see **Note 1**)	nu1	I	Crypto RAM	u2NumLength - u2ModLength + 4	Base of the Quotient	Base of the Quotient
nu1WorkSpace	nu1	I	Crypto RAM	GF(p): 64 GF(2n): 68	Base of the WorkSpace	Base of the WorkSpace corrupted
nu1RBase (see **Note 2**)	nu1	I	Crypto RAM	u2ModLength	Base of the Remainder	Base of the Remainder

Notes:

1. If the quotient is not needed, set nu1QuoBase to zero and the quotient will not be written to memory. If the quotient is needed, set the nu1QuoBase to the beginning of an area of size (u2NumLength - u2ModLength + 4) to write the whole quotient.
2. The Remainder is present in the area {nu1NumBase, u2NumLength} at the end of the calculus. The nu1RBase parameter makes it possible to copy this result in the other area {nu1RBase, u2ModLength}, if this copy is not needed, set nu1RBase to the same value as nu1NumBase and the copy will not be done.

Note: The parameter Num must have its most significant 32-bit word cleared to zero. The length u2NumLength is the length of Num including this zero word.
One additional word is used on the LSB side of the Num parameter, this word is restored at the end of the calculus. As a consequence the parameter nu1NumBase must never been at the beginning of the Crypto RAM, i.e., ensure that nu1NumBase ≥ <Crypto RAM Base> + 4 bytes.

One additional word is used on the MSB side of the Num parameter, this word is not corrupted. As a consequence the Area {nu1NumBase, u2NumLength} must not be at the end of the Crypto RAM, i.e., en sure that nu1NumBase+u2NumLength ≤ <Crypto RAM End> - 4.

u2ModLength must be the true length of the Modulus, i.e., the MSB word of the area {nu1ModBase, u2ModLength} must be different from zero.

The minimum value for u2ModLength is 8 bytes, so the significant length of Num must be at least 8 bytes. To divide by a 32-bit value, the divider and numerator shall be multiplied by 232. The resulting remainder will have to be divided by 2^{32}, the quotient will be exact.

37.3.4.11.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// Fill all the fields
// In that case, the quotient will be computed
```

```
// If it was not needed, set nu1QuoBase to NULL
PUKCL_Div(nu1NumBase) = <Base of the ram location of Num>;
PUKCL_Div(nu1ModBase) = <Base of the ram location of Mod>;
PUKCL_Div(nu1QuoBase) = <Base of the ram location of Quo>;
PUKCL_Div(nu1WorkSpace) = <Base of the workspace>;
PUKCL_Div(nu1RBase) = <Base of the ram location of R>;
PUKCL_Div(u2NumLength) = <Length of Num>;
PUKCL_Div(u2ModLength) = <Length of Mod>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Div,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            // The Division has been executed correctly
            ...
            }
else // Manage the error
```

37.3.4.11.6 Constraints

The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1RBase, nu1QuoBase, nu1WorkSpace or nu1NumBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength}, {nu1RBase, u2ModLength}, {nu1WorkSpace, 64} or{nu1NumBase, u2NumLength} are not in Crypto RAM
- u2ModLength, u2NumLength is either: < 4, > 0xffc or not a 32-bit length
- One or more overlaps exist between two of the areas: {nu1ModBase,u2ModLength},{nu1RBase, u2ModLength} {nu1NumBase, u2NumLength}(1) or {nu1WorkSpace,64}
- If nu1QuoBase is different from zero and: {nu1QuoBase, u2NumLength - u2ModLength + 4} are not in Crypto RAM
- If nu1QuoBase is different from zero and one or more overlaps exist between two of the areas: {nu1QuoBase, u2NumLength - u2ModLength + 4}, {nu1ModBase, u2ModLength}, {nu1RBase, u2ModLength}, {nu1NumBase, u2NumLength} or {nu1WorkSpace, 64}

Overlaps between {nu1RBase, u2ModLength} and {nu1NumBase, u2NumLength} are forbidden, but the equality between nu1RBase and nu1NumBase is authorized

37.3.4.11.7 Status Returned Values

Table 37-36. Div Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly.
PUKCL_DIVISION_BY_ZERO	Severe	The operation was not performed because the Denominator value is zero.

37.3.4.12 GCD, Modular Inverse

37.3.4.12.1 Purpose

The purpose of this command is to compute the Greatest Common Divisor (GCD) and the Modular Inverse. The algorithm used is the Extended Euclidean Algorithm for the GCD.

This command accepts as input two multiple precision numbers in GF(p) or two polynomials in GF(2^n) X and Y and computes their GCD (D), if D equals one, the command also supplies the inverse of X modulo Y.

The available options are as follows:

- Work in the GF(2^n) field or in the standard integer arithmetic field GF(p)

37.3.4.12.2 How to Use the Service

37.3.4.12.3 Description

This command calculates:

D = GCD(X,Y).

and parameter A in the Bezout equation:

$A \times X + B \times Y = D$.

The first input, or input to inverse is X.

The second input, or modulus is Y.

The GCD is output in D.

The modular inverse if X and Y are co-primes is output A:

$A = X^{-1} mod(Y)$

The command calculates the GCD and the value A. The value A is the multiplicative inverse of X, only if X and Y are co-prime. As a supplemental result, Z is given back, being the quotient of Y divided by D only if D is different from zero:

$Z = \left[\frac{Y}{D}\right]$

At the end of the command: X is overwritten by D.

Y is cleared.

The value of A is calculated and stored.

The value of Z is calculated and stored if D is different from zero.

The service name for this operation is `GCD`.

In this computation, the following areas have to be provided:

- X (pointed by {nu1XBase,u2Length}) filled with X (with MSB word to zero)
- Y (pointed by {nu1YBase,u2Length}) filled with Y (with MSB word to zero)
- A (pointed by {nu1ABase,u2Length}) to contain calculated A
- Z (pointed by {nu1ZBase,u2Length}) to contain calculated Z
- The workspace (pointed by {nu1WorkSpace,32})

37.3.4.12.4 Parameters Definition

Table 37-37. GCD Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/Gf2n	Bit	I	–	–	GF(2n) Bit	–
nu1XBase	nu1	I	Crypto RAM	u2Length	Base of X Number X	Base of X Filled with the GCD D
u2Length	u2	I	–	–	Length of the Areas X, Y, A, Z	Length of the Areas X, Y, A, Z
nu1YBase	nu1	I	Crypto RAM	u2Length	Base of Y Number Y	Base of Y Cleared area
nu1ABase	nu1	I	Crypto RAM	u2Length	Base of A	Base of A Filled with the result
nu1ZBase	nu1	I	Crypto RAM	u2Length + 4 (see **Note 1**)	Base of Z	Base of Z Filled with the result
nu1WorkSpace	nu1	I	Crypto RAM	32 bytes	Base of the workspace	Base of the workspace corrupted

Note:

1. The additional word is 4 zero bytes.

The parameters X and Y must have their most significant 32-bit word cleared to zero. The length u2Length is the length of the longer of the parameters X and Y including this zero word.

To clarify here is an example:

- X is an 8 bytes number.
- Y is a 12 bytes number.

This example is processed this way before the use of the GCD service:

- The longer number is Y so its length is taken and increased by 4 bytes for the 32-bit word cleared to zero, this gives u2Length = 16 bytes. Therefore, X, Y, A and Z areas have a length of 16 bytes.
- Y is padded with 4 bytes cleared to zero on the MSB side and the u2Length = 16 bytes are written in memory (LSB first).
- X is padded with 8 bytes cleared to zero on the MSB side and the u2Length = 16 bytes are written in memory (LSB first).
- The areas A and Z are mapped in memory with a size of u2Length = 16 bytes.
- The workspace is mapped in memory with its constant size of 32 bytes

37.3.4.12.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
// Fill all the fields
PUKCL(u2Option) = 0;
PUKCL_GCD(nu1XBase) = <Base of the ram location of X>;
PUKCL_GCD(nu1YBase) = <Base of the ram location of Y>;
PUKCL_GCD(nu1ABase) = <Base of the ram location of A>;
PUKCL_GCD(nu1ZBase) = <Base of the ram location of Z>;
PUKCL_GCD(nu1WorkSpace) = <Base of the workspace>;
PUKCL_GCD(u2Length) = <Length of X, Y, A and Z>;
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GCD, pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
               {
               // The GCD has been executed correctly
               ...
               }
else // Manage the error
```

37.3.4.12.6 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1XBase, nu1YBase, nu1ABase or nu1ZBase are not aligned on 32-bit boundaries
- {nu1XBase, u2Length}, {nu1YBase, u2Length}, {nu1ABase, u2Length} or {nu1ZBase, u2Length} are not in Crypto RAM
- u2Length is either: < 4, > 0xffc or not a 32-bit length
- {nu1XBase, u2Length} overlaps {nu1YBase, u2Length} or {nu1XBase, u2Length} overlaps {nu1ABase, u2Length} or {nu1XBase, u2Length} overlaps {nu1ZBase, u2Length} or {nu1YBase, u2Length}overlaps

{nu1ABase, u2Length} or {nu1YBase, u2Length} overlaps {nu1ZBase, u2Length} or {nu1ABase, u2Length} overlaps {nu1ZBase, u2Length}

37.3.4.12.7 Status Returned Values

Table 37-38. GCD Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

37.3.4.13 Get Random Number

37.3.4.13.1 Purpose

The purpose of this command is to provide the user with a source of entropy. The options available for this service are:

- Generation of random numbers from a Hardware Random Number Generator (TRNG).
- Generation of random numbers from a Deterministic Random Number Generator (DRNG).

Important:
When using this service, be sure to strictly follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG. If the directives require not to use this service, follow them and use the proposed method to get random numbers.

This service only has the option to get random numbers and does not seed, initialize or start the RNG. Other options are reserved for future use.

Neither continuous testing nor entropy testing is included in this service. If this is needed (FIPS 140, ZKA, ...), this service must not be used and the users develops their own command.

The DRNG is compatible with both ANSI X9.31 and FIPS 186-2 standards (see the important note below). The DRNG is designed according to:

- The algorithm described in the document ANSI Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA) X9.31 dated September 9, 1998.
- The Change recommendation for ANSI X9.0 - 1995 (Part 1) and ANSI X9.31 -1998:

The algorithm B.2.1 Algorithm for computing m Values of x is the one applied in the Toolbox 3 X9.31 DRNG. The DRNG is compatible with:

- The DRNG is described in the document NIST Digital Signature Standard (DSS) FIPS Pub 186-2 January 27, 2000 Appendix 3.1
- The FIPS 186-2 Change Notice 1 dated October 5, 2001 modifies this algorithm.

Important: To apply the FIPS 186-2 algorithm, the parameters XSeed[0] and XSeed[1] must be set to the same value.

37.3.4.13.2 How to Use the Service

37.3.4.13.3 Description

This service has four possible options described in Table 37-41. Two of these options are reserved for future use. This service performs the following operations:

- Generation of a random number from the Hardware RNG
- Generation of a random number from the Deterministic RNG

Generation of a Random Number from the Hardware RNG

This service, activated with the option PUKCL_RNG_GET, makes it possible to get a random number R from the Hardware RNG:

R = HardwareRandomGenerate()

In the Generation of random from the RNG service, the following parameters need to be provided:

- R the generated number area (pointed by{nu1RBase,u2RLength})

37.3.4.13.4 Generation of a Random Number from the Deterministic RNG

This service, activated with the option PUKCL_RNG_X931_GET, makes it possible to get a random number R from the Deterministic Random Number Generator with input parameters the Key XKey and the Seed XSeed:

(XKey, R) = DeterministicRandomGenerateFromSeed (XKey, XSeed, Q)

In the generation of a random number from the Deterministic RNG service, the following parameters need to be provided:

- XKey the input and output Key (pointed by {nu1XKeyBase,u2XKeyLength})

- XSeed the input Seed (pointed by {nu1XseedBase,u2XKeyLength})
- Q the prime number (pointed by {nu1QBase, 20bytes})
- R the generated number area (pointed by {nu1RBase, 20bytes})

37.3.4.13.5 Hardware RNG Parameters Definition

The parameters for the generation of random from the Hardware RNG are described in the following table. This service can easily be accessed through the use of the `PUKCL_Rng()` and `PUKCL()` macros.

Table 37-39. RNG Service Hardware Generated Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Option (see Table 37-41)	Option (see Table 37-41)
nu1RBase	nu1	I	Crypto RAM or Device RAM	u2RLength	Base of R	Base of R filled with random values depending on the option
u2RLength	u2	I	–	–	Length of R	Length of R

37.3.4.13.6 Deterministic RNG Parameters Definition

The parameters for the generation of random from the Deterministic RNG are described in the following table. This service can easily be accessed through the use of the `PUKCL_Rng()` and `PUKCL()` macros.

Table 37-40. RNG Service Deterministic Generated Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Option (see Table 37-41)	Option (see Table 37-41)
nu1XKeyBase	nu1	I/O	Crypto RAM	u2XKeyLength	Base of XKey	Base of XKey filled with the resulting XKey
nu1Workspace	nu1	NA	Crypto RAM	64 bytes	Base of the workspace	Base of the workspace corrupted
nu1Workspace2(1)	nu1	NA	Crypto RAM	2*u1XKeyLength + 4	Base of the workspace 2	Base of the workspace corrupted
nu1XSeedBase	nu1	I/O	Crypto RAM	max (2*u2XKeyLength, 44 bytes)	Base of the values XSeed[0] and XSeed[1]	Base of XSeed filled with the result on 20 bytes
u2XKeyLength	u2	I	–	–	Length of XKey, Xseed[0] and Xseed[1]	Length of XKey, Xseed[0] and Xseed[1]
nu1QBase	nu1	I	Crypto RAM	20 bytes	Base of Q	Base of Q
nu1RBase	nu1	I	Crypto RAM	u2RLength	Base of R	Base of R filled with the result on 20 bytes

Note:

1. The nu1 Workspace2 must be a multiple of 256.

37.3.4.13.7 Options

The option is set by the u2Options input parameter that must take one of the values listed in the following table.
Note: The values, OPTION_RNG_SEED and OPTION_RNG_GETSEED, are reserved for future use.

Table 37-41. RNG Service Options

Option	Purpose	Required Parameters
PUKCL_RNG_SEED	Reserved	Reserved
PUKCL_RNG_GET	Generation of a random number from the RNG	nu1RBase, u2RLength
PUKCL_RNG_X931_GET	Generation of a random number from the Deterministic RNG	nu1XKeyBase, nu1Workspace, nu1XSeedBase, u2XKeyLength, nu1QBase, nu1RBase
PUKCL_RNG_GETSEED	Reserved	Reserved

37.3.4.13.8 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL(u2Option) =...;

// Initializing parameters
PUKCL_Rng(nu1RBase) = <Base of the ram location to store the rng>;
PUKCL_Rng(u2RLength) = <Length of the rng to get>;

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(Rng,pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            // The RNG generation has been executed correctly
            ...
            }
else // Manage the error
```

37.3.4.13.9 Constraints

Random Number Generation
The following conditions must be avoided to ensure that the service works correctly:

- {nu1RBase,u2RLength} not in RAM
- {nu1RBase,u2RLength} not accessible or authorized for writing

Deterministic Random Number Generation
The length of the parameter nu1XSeedbase is: XSeedLength = max(2*u2XKeyLength, 44 bytes) The `max()` macro takes a maximum of two values.

The following conditions must be avoided to ensure that the service works correctly:

- nu1XKeyBase,nu1Workspace, nu1Workspace2, nu1XSeedBase, nu1QBase, nu1RBase are not aligned on 32-bit boundaries
- {nu1XKeyBase, u2XKeyLength}, {nu1Workspace, 64 bytes}, {nu1Workspace2, 2*u1XKeyLength +4}, {nu1XSeedBase, XSeedLength}, {nu1QBase, 24 bytes} or {nu1RBase, 20 bytes} are not in PUKCC RAM
- u2XKeyLength is either: < 20, > 64 or not a 32-bit length
- nu1Workspace2 not multiple of 256.
- Overlaps exist between two or more of the areas: {nu1XKeyBase, u2XKeyLength}, {nu1Workspace,64 bytes}, {nu1XSeedBase, XSeedLength}, {nu1QBase, 24 bytes} or {nu1RBase, 20 bytes}
  The area {nu1RBase, 20} can overlap with {nu1Workspace, 64 bytes} or {nu1QBas, 24 bytes}. The pointer nu1RBase can equal the pointer nu1XSeedBase.

37.3.4.13.10 Status Returned Values

Table 37-42. RNG Service Return Codes

Returned status	Importance	Meaning
PUKCL_OK	Information	Service functioned correctly

37.3.5 Modular Arithmetic Services

This section provides a complete description of the modular arithmetic services, which consists of two sets:

- Modular reductions, which can be used as stand alone operations, or used as a final step of most arithmetic operations (full and small multiplications, squaring).
- Modular operations, which include modular exponentiations (with or without using the CRT) and a probabilistic prime number generation.

These operations work on general data so the modulus has no special form. The modular services are available through:

- a Fast form (may return a congruence of the result, with a high probability to have a Normalized result)
- a Normalized form (returns the exact result, strictly lower than the modulus)
- a Euclidean form (returns the exact result, strictly lower than the modulus)

The following table describes the modes of the modular reduction with the hypothesis:

- In GF(p): The modulus is N with length NLength in bytes
- In GF(2^n): The modulus is P[X] with length NLength in bytes

For the exact calculus of NLength see below.

Table 37-43. Modular Reduction Modes

Modular Reduction Form	Input Dynamic	Result Dynamic	Comments
Fast	GF(p): 0 ≤ Input < (N^2) * (2^{32}) GF(2n): Input < (($P[x]$)^2) * (X^{32})	GF(p): 0 ≤ Res < N * 4 GF(2^n): Res < P[X] * (X^2)	The fastest reduction available, needs a precomputed constant.
Normalized	InputLength < NLength + 4 bytes	GF(p): 0 ≤ Res < N GF(2^n): Res < P[X]	The correction step does not runs in constant time. Needs a precomputed constant. The Normalize function cannot be applied to the product of two numbers of length u2NLength.
Using Euclidean division	InputLength < 2 * NLength + 4 bytes	GF(p): 0 ≤ Res < N GF(2^n): Res < P[X]	Does not need any precomputed constant.

To be able to use these modular reduction services (except the Euclidean division), first the implementer shall call the setup service, providing the modulus as well as one free memory space for the constant (this constant is used to speed up the modular reduction). In most commands (except the modular exponentiation), the quotient is stored in the high order bytes of the number to be reduced, using only eight bytes more than the maximum size of the number to be reduced.

The following rules must be respected to ensure the modular reduction services function correctly:

- The numbers to be reduced can have any significant length, given the fact it CANNOT BE GREATER than 2*u2ModLength + 4 bytes.

- The modulus SHALL ALWAYS HAVE a significant length of <u2ModLength> bytes. The modulus must be provided as a <u2ModLength + 4> bytes long number, padded on the most significant side with a 32-bit word cleared to zero. Not respecting this rule leads to unexpected and wrong results from the modular reduction.
- The normalization operation ALWAYS performs a modular reduction step, and will therefore have the same memory usage as this one.
- The very first operation before any modular operation SHALL BE a modular setup.

37.3.5.1 Modular Reduction

Related Links

37.3.3.4. Aligned Significant Length

37.3.5.1.1 Purpose

This service is used to perform the various steps necessary to perform a modular reduction and accepts as input numbers in GF(p) or polynomials in GF(2^n) .

The available options for this service are:

- Work in the GF(2^n) or in the standard integer arithmetic field GF(p)
- Operation is the generation of the reduction constant.
- Operation is a Modular Reduction.
- Operation is a Normalization.

37.3.5.1.2 How to Use the Service

37.3.5.1.3 Description

This service performs one of the following operations:

- Setup of the Fast or Normalize functions: generation of the reduction constant
- Fast Modular Reduction
- Big Modular Reduction (using Euclide's division)
- Normalization

The service name for this operation is `RedMod`.

37.3.5.1.4 Modular Reduction Setup

This service calculates the constant Cns, computed from the modulus and used to speed up the modular reduction:

Cns = SetupConstant(N)

This service must be processed before the use of the Fast or Normalize functions. In the Setup computations, the following data must be provided:

- N the modulus (pointed by {nu1ModBase,u2ModLength +4}).
- Cns the Setup Constant Result (pointed by {nu1CnsBase,u2ModLength +12}).
- X used as a workspace (pointed by {nu1XBase,2 * u2ModLength + 8}) (include the supplementary bytes; see **Note 2** in Table 37-44
- R used as a workspace (pointed by {nu1RBase,64 or 68bytes}).
- u2ModLength is the Aligned Significant Length of the modulus and is not the byte Significant Length (see *Aligned Significant Length* from Related Links).

37.3.5.1.5 Fast Reductions and Normalization

These commands calculate an approximated or exact Modular Reduction, that is, the result may be greater than the modulus, but is always congruent to the true result.

Important: Before using these functions, ensure that the constant Cns has been calculated with the setup for the Modular Reduction service.

Input and Result significant values verify:

- For the Fast Modular Reduction:

$0 \leq X < N^2 \times 2^{32}$

$R = Xmod(N) + k \times N \quad with \quad 0 \leq k \leq 4$

- For the Normalize:

$XLength < (NLength + 4)bytes \;\; R$

$= Xmod(N)$

In these Fast Modular Reduction and Normalize computations, the following data have to be provided:

- X (pointed by {nu1XBase,2 * u2ModLength +8})
  - The Normalize computation accept as entry a value whose length is lower or equal to u2ModLength + 4 (that is, for example, a value yet reduced but not normalized.). The u2ModLength + 4 MSB bytes are cleared at the beginning of the computation.
  - in case of Fast RedMod computations, the value X mayverify: X < (N^2) *(2^{32}).
  - include the supplementary bytes; see **Note 3** in 37.3.5.1.8. Fast Modular Reductions Service Parameters Definition.
- R (pointed by {nu1RBase,u2Modlength +4})
- N (pointed by {nu1ModBase,u2ModLength +4})
- Cns (pointed by {nu1CnsBase,u2ModLength +12})
- u2ModLength is the Aligned Significant Length of the modulus and is not the byte Significant Length (see *Aligned Significant Length* from Related Links).

The Fast Modular Reduction is able to reduce inputs up to <2*u2ModLength + 4> bytes. The input can come from a multiplication of 2 <u2ModLength + 4> bytes numbers. The input X is considered as a <2*u2ModLength + 8> bytes number.

Important: Additionally the Fast Reduction and Normalize functions need supplemental bytes located on the MSB side of the number to be reduced but these bytes are restored at the end of the operation and are therefore unchanged. However, these bytes are to be taken into account when the mapping is created, and could lead to unexpected results if overlapping with other area used by the function.

The Fast Modular Reduction returns a <u2ModLength + 4> bytes number, but this number is in fact a <u2ModLength + 2> significant bytes number. When using the Fast Modular Reduction, the two MSB bytes of the <u2ModLength + 2> can have a maximum of two lsb bits set (depending on the reduced number and the modulo).

The Normalize computation accepts as entry a resulting value of Fast Modular Reduction and computes an exact result. It can not be applied to the result of the product of two numbers of size NLength: a Fast Modular Reduction must be applied before.

For the Normalize computation, the X value is limited by the preceding formula but the area in memory is bigger as described in 37.3.5.1.8. Fast Modular Reductions Service Parameters Definition.

As input, the Normalize sub-service only accept values, which length is lower or equal to u2ModLength + 4. The Most Significant u2ModLength + 4 bytes are firstly cleared by this service.

37.3.5.1.6 Big Modular Reduction Using Euclide's Division

This command calculates:

$XLength < (2 \times NLength + 4)bytes \;\; R$

$= Xmod(N)$

In this Big Modular Reduction computations, the following data must be provided:

- X (pointed by {nu1XBase,2 * u2ModLength + 8}) (include the supplementary bytes; see **Note 1** in Table 37-46)
- R (pointed by {nu1RBase,u2Modlength +4})
- N (pointed by {nu1ModBase,u2ModLength +4})

- u2ModLength is the Aligned Significant Length of the modulus and is not the byte Significant Length (see *Aligned Significant Length* from Related Links).
- Workspace (pointed by {nu1CnsBase,64 or 68}).

37.3.5.1.7 Modular Reductions Service Parameters Definition

Table 37-44. RedMod Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Options (see below)	Options (see below)
Specific/CarryIn	Bits	I	–	–	Must be set to zero.	–
Specific/Gf2n	Bit	I	–	–	GF(2^n) Bit	–
Specific/CarryOut Zero Violation	Bits	I	–	–	–	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase(1)	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns filled with the Setup Constant
u2ModLength	u2	I	–	–	Length of N	Length of N
nu1RBase	nu1	I	Crypto RAM	GF(p): 64 bytes GF(2n): 68 bytes	Base of R as a workspace	Base of R workspace corrupted
nu1XBase(2)	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of X as a workspace	Base of X workspace corrupted

Notes:

1. The Modulus is to be given as a u2ModLength Aligned Significant Length Bytes however, it has to be provided as a u2ModLength + 4 bytes long number, having the four high-order bytes set to zero.
2. Before the X (pointed by {nu1XBase,2 * u2ModLength + 8}) LSB bytes, four supplementary bytes will be saved/restored. Other four supplementary bytes will also be saved/restored after the X MSB bytes. All these supplementary bytes may be entirely in the Crypto RAM (therefore, do not place the X area too near the end of the Crypto RAM) and shall not overlap with other area used by the service.

37.3.5.1.8 Fast Modular Reductions Service Parameters Definition

Table 37-45. Fast RedMode and Normalize Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Options (see below)	Options (see below)
Specific/CarryIn	Bits	I	–	–	Must be set to zero.	–
Specific/Gf2n	Bit	I	–	–	GF(2n) Bit	–

.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
Specific/CarryOut Zero Violation	Bits	I	–	–	–	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase(1)	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	–	–	Length of N	Length of N
nu1RBase(2)	nu1	I	Crypto RAM	u2ModLength + 4	Base of R	Base of R filled with the result
nu1XBase(3)	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of X the number to reduce	Base of X corrupted

Notes:

1. The Modulus is to be given as a u2ModLength Aligned Significant Length Bytes however, it has to be provided as a u2ModLength + 4 bytes long number, having the four high-order bytes set to zero.
2. To make profitable the space memory, it is possible to set nu1RBase exactly equal to nu1XBase.
3. After the X (pointed by {nu1XBase,2 * u2ModLength + 8}) MSB bytes, supplementary bytes will be saved/restored (8 bytes in case of Fast RedMod, otherwise; 12 bytes). These supplementary bytes may be entirely in the Crypto RAM (therefore, do not place the X area too near the end of the Crypto RAM) and shall not overlap with other area used by the service.

37.3.5.1.9 Big Modular Reduction Parameters Definition

Table 37-46. Big RedMod Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Options (see below)	Options (see below)
Specific/CarryIn	Bits	I	–	–	Must be set to zero	–
Specific/Gf2n	Bit	I	–	–	GF(2n) Bit	–
Specific/CarryOut Zero Violation	Bits	I	–	–	–	Carry Out, Zero Bit and Violation Bit filled according to the result
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	GF(p): 64 bytes GF(2n): 68 bytes	Base of Cns as a workspace	Base of Cns corrupted
u2ModLength	u2	I	–	–	Length of N	Length of N
nu1RBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of R	Base of R filled with the result
nu1XBase(1)	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of X the number to reduce	Base of X filled with the result

Note:

1. Before the X (pointed by {nu1XBase,2 * u2ModLength + 8}) LSB bytes, four supplementary bytes will be saved/restored. Other four supplementary bytes will also be saved/restored after the X MSB bytes. All of these supplementary bytes may be entirely in the Crypto RAM (therefore, do not place the X area too near the end of the Crypto RAM) and shall not overlap with other area used by the service.

37.3.5.1.10 Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Operation Option described in Table 37-47
- if the Operation Option is PUKCL_REDMOD_REDUCTION, the Modular Reduction Sub-Option described in Table 37-48

The u2Options number is calculated by an Inclusive OR of the options. Some Examples in C language are:

- Operation: Setup for the ModularReductions.
  `PUKCL(u2Options) = PUKCL_ REDMOD_SETUP;`
- Operation: Fast ModularReduction.
  `PUKCL(u2Options) = PUKCL_REDMOD_REDUCTION | PUKCL_REDMOD_USING_FASTRED;`

For this command three exclusive options can be specified. The following table lists the operations that can be performed.

Table 37-47. RedMod Service Options

Option	Purpose	Required Parameters
PUKCL_REDMOD_SETUP	Perform the Cns value computation	nu1ModBase, u2ModLength, nu1CnsBase, nu1XBase
PUKCL_REDMOD_REDUCTION	Perform R ≡ X Mod N, see sub-option for details	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase, nu1RBase
PUKCL_REDMOD_NORMALIZE	Perform R = X Mod N	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase, nu1RBase

When selecting the PUKCL_REDMOD_REDUCTION option, one of the two sub-options listed in the following table must be selected.

Table 37-48. RedMode Service Options with PUKCL_RED_MOD_REDUCTION

Option	Purpose	Required Parameters
PUKCL_REDMOD _USING_DIVISION	Perform R = X Mod N	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase
PUKCL_REDMOD _USING_FASTRED	Perform R ≡ X Mod N The entropy is minimized (~2 bits)	nu1ModBase, u2ModLength, nu1CndBase, nu1XBase, nu1RBase

37.3.5.1.11 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL(Specific).CarryIn = 0;
PUKCL(Specific).GF2n = ...;

PUKCL(u2Option) =...;

// Depending on the option specified, not all fields must be filled
PUKCL_RedMod(nu1ModBase) = <Base of the ram location of N>;
PUKCL_RedMod(u2ModLength) = <Length of N>;
PUKCL_RedMod(nu1CnsBase) = <Base of the ram location of Cns>;
...
```

```
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(RedMod,pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
            {
            // operation has correctly been performed
            ...
            }
else // Manage the error
```

37.3.5.1.12 Constraints

Depending on the options chosen the lengths of the R area and Cns area differ:

- For the Setup:
  - RLength = 64bytes
  - CnsLength = u2ModLength +12
- For the Fast Reduction and Normalize:
  - RLength = u2ModLength +4
  - CnsLength = u2ModLength +8
- For the BigRedMod:
  - RLength = u2ModLength +4
  - CnsLength =64

The following combinations of input values must be avoided in the case of a modular reduction ‘alone’, meaning that it has not been requested as an option of any other command:

- nu1ModBase, nu1CnsBase, nu1RBase, nu1XBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2CnsLength}, {nu1XBase, 2*u2XLength + 8 + s} or {nu1RBase, u2RLength} are not in Crypto RAM
- u2ModLength is either: < 4, > 0xffc or not a 32-bit length
- Overlaps exist between two or more of the areas: {nu1ModBase, u2ModLength + 4},{nu1CnsBase, u2CnsLength}, {nu1XBase, 2*u2XLength + 8 + s} or {nu1RBase, u2RLength}

Note: Overlaps between {nu1RBase, RLength} and {nu1XBase, 2*u2XLength + 8} are forbidden; but if the operation is the Fast, Normalized or Big Modular Reduction, the equality between nu1RBase and nu1XBase is authorized.

37.3.5.1.13 Status Returned Values

Table 37-49. RedMod Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly
PUKCL_DIVISION_BY_ZERO	Severe	When computing an Euclidean division, the Modulus was found to be zero. This occurs ONLY when either reducing using an Euclidean division or computing the reduction constant usable for a Fast or Normalize Reduction.
PUKCL_UNEXPLOITABLE_OPTIONS	Severe	A bad combination of options has been detected.
PUKCL_MALFORMED_MODULUS	Severe	The Msw of the modulus is not zero.

37.3.5.2 Modular Exponentiation (Without CRT)

37.3.5.2.1 Purpose

This service is used to perform the Modular Exponentiation computation. This service processes integers in GF(p) only.

The options available for this service are:

- Fast implementation
- Regular implementation

- Exponent is located in Crypto RAM or not in Crypto RAM
- Exponent window size

37.3.5.2.2 How to Use the Service

37.3.5.2.3 Description

Important: Before using these functions, ensure that the constant Cns has been calculated with the Setup of the Modular Reductions service.

This service processes the following operation:

The service name for this operation is `ExpMod`.

$R = X^{Exp} mod(N)$

In this computation, the following parameters need to be provided:

- X: input number (pointed by {nu1XBase,u2ModLength +16})
- N: modulus (pointed by {nu1ModBase,u2ModLength +4}).
- Exp: exponent (pointed by {pfu1ExpBase,u2ExpLength +4})
- Cns: Fast Modular Constant (pointed by {nu1CnsBase,u2ModLength +8})
- Precomp: precomputation workspace (pointed by{nu1PrecompBase,PrecompLen})
- Blinding: exponent blinding value (provided inu1Blinding)

The length PrecompLen depends on the lengths and options chosen; its calculus is detailed in Options below.

Note: The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of N must be at least three 32-bit words.

37.3.5.2.4 Parameters Definition

Table 37-50. ExpMod Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Options (see below)	Options (see below)
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of N	Base of N untouched
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns untouched
u2ModLength	u2	I	–	–	Length of N	Length of N
nu1XBase(1)	nu1	I	Crypto RAM	u2ModLength + 16	Base of X	Base of X Filled with the result
nu1PrecompBase	nu1	I	Crypto RAM	See below	Base of Precomp as a workspace	Base of Precomp workspace corrupted
pfu1ExpBase(2)	pfu1	I	Any place(3)	u2ExpLength + 4	Base of the Exponent	Base of the Exponent untouched
u2ExpLength(4)	u2	I	–	–	Significant length of Exponent	Significant length of Exponent
u1Blinding(5)	u1	I	–	–	Exponent unblinding value	Exponent unblinding value untouched

Notes:

1. This zone contains the number to be exponentiated (u2ModLength bytes) and is used during the computations as a workspace (four 32-bit words longer than the number to be exponentiated). At the end of the computation, it contains the correct result of the operation.
2. The exponent must be given with a supplemental word on the LSB side (low addresses). This word shall be set to zero.
3. If the PUKCL_EXPMOD_EXPINPUKCCRAM option is not set, the location of the exponent MUST NOT be the Crypto RAM, even partially.
4. The u2ExpLength parameter does not take into account the supplemental word needed on the LSB side of the exponent.
5. It is possible to mask the exponent in memory using an 8-bits XOR mask value. Be aware that not only the exponent, but also the supplemental word has to be masked. If masking is not desired, then this parameter must be set to 0.

37.3.5.2.5 Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Calculus Mode Option described in Table 37-51
- the mandatory Window Size Option described in Table 37-52
- the indication of the presence of the exponent in Crypto RAM

Note: Please check precisely if one part of the exponent is in Crypto RAM. If this is the case the PUKCL_EXPMOD_EXPINPUKCCRAM must be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some examples in C language are:

- Operation:Fast Modular Exponentiation with the window size equal to 1 and with no part of the Exponent in the Crypto RAM
  ```
  PUKCL(u2Options) = PUKCL_EXPMOD_FASTRSA | PUKCL_EXPMOD_WINDOWSIZE_1;
  ```
- Operation: Regular Modular Exponentiation with the window size equal to 2 and with one part of the Exponent in the Crypto RAM
  ```
  PUKCL(u2Options) = PUKCL_EXPMOD_REGULARRSA | PUKCL_EXPMOD_WINDOWSIZE_2 |
  PUKCL_EXPMOD_EXPINPUKCCRAM;
  ```

There is no difference on the final result when using any of the options for this service. The choice has to be made according to the available resources (RAM, Time) and also taking into account the expected security level.

For this service, two exclusive Calculus Modes are possible. The following table describes the Calculus Mode Options.

Table 37-51. ExpMod Service Calculus Mode Option

Option	Explanation
PUKCL_EXPMOD_FASTRSA	Performs a Fast computation
PUKCL_EXPMOD_REGULARRSA	Performs a Regular computation, slower than the Fast version, but using Regular calculus methods

For this service, four window sizes are possible. The window size in bits is those of the windowing method used for the exponent.

The choice of the window size is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation workspace.
- Increasing the window size reduces the computation time (may not be relevant for very small exponents).

The following table details the size of the precomputation workspace, depending on the chosen window size option.

Table 37-52. ExpMode Service Window Size Options and Precomputation Space Size

Option specified	Size of the PrecompBase Workspace (bytes)	Content of the Workspace
PUKCL_EXPMOD_WINDOWSIZE_1	3*(u2ModLength + 4) + 8	x
PUKCL_EXPMOD_WINDOWSIZE_2	4*(u2ModLength + 4) + 8	$x\ x^3$
PUKCL_EXPMOD_WINDOWSIZE_3	6*(u2ModLength + 4) + 8	$x\ x^3\ x^5\ x^7$
PUKCL_EXPMOD_WINDOWSIZE_4	10*(u2ModLength + 4) + 8	$x\ x^3\ x^5\ x^7\ x^9\ x^{11}\ x^{13}\ x^{15}$

The exponent can be located in RAM or in the data space. If one part of the exponent is in Crypto RAM this must be mandatory signaled by using the option PUKCL_EXPMOD_EXPINPUKCCRAM.

The following table describes this option.

Table 37-53. ExpMod Service Exponent in Crypto RAM Option

Option	Purpose
PUKCL_EXPMOD_EXPINPUKCCRAM	The exponent can be read from any data space of memory, including Flash, RAM or even Crypto RAM. When at least one word the exponent is in Crypto RAM, this option has to be set.

37.3.5.2.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL(u2Option) =...;

// Depending on the option specified, not all fields must be filled
PUKCL_ExpMod(nu1ModBase) = <Base of the ram location of N>;
PUKCL_ExpMod(u2ModLength) = <Length of N>;
PUKCL_ExpMod(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_ExpMod(nu1XBase) = <Base of the ram location of X>;
PUKCL_ExpMod(nu1PrecompBase) = <Base of the ram location of Precomp>;
PUKCL_ExpMod(pfu1ExpBase) = <Base of the location of Exp>;
PUKCL_ExpMod(u2ExpLength) = <Length of Exp>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ExpMod, pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
            {
            // operation has been performed correctly
            ...
            }
else // Manage the error
```

37.3.5.2.7 Constraints

The following combinations of input values must be avoided in the case of a modular reduction ‘alone’, meaning that it has not been requested as an option of any other command:

- nu1ModBase,nu1CnsBase, nu1XBase,nu1PrecompBase,nu1ExpBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1XBase, u2ModLength +16}, {nu1PrecompBase, <PrecompLength>} are not in Crypto RAM
- {nu1ExpBase,u2ExpLength + 4} has no part in Crypto RAM and PUKCL_EXPMOD_EXPINPUKCCRAM is specified
- u2ModLength or u2ExpLength are either: < 4, > 0xffc or not a 32-bit length
- None or both PUKCL_EXPMOD_REGULARRSA and PUKCL_EXPMOD_FASTRSA are specified.
- {nu1PrecompBase,<PrecompLength>} overlaps with either: {nu1ModBase, u2ModLength +4},{nu1CnsBase, u2ModLength + 8} {nu1XBase, u2ModLength + 16} or {nu1ExpBase, u2ExpLength + 4}

- {nu1XBase,u2ModLength + 16} overlaps with either: {nu1ModBase, u2ModLength + 4},{nu1CnsBase, u2ModLength + 8} or {nu1ExpBase, u2ExpLength + 4}
- {nu1ModBase, u2ModLength + 4} overlaps {nu1CnsBase, u2ModLength +8}

37.3.5.2.8 Maximum Sizes for the Modular Exponentiation

The following table provides the maximum sizes for the Modular Exponentiation, depending on the window size and the presence of the exponent in Crypto RAM.

- The figures below are calculated supposing that u2ExpLength =u2ModLength.
- In case of the PUKCL_EXPMOD_EXPINPUKCCRAM option is specified, for the computation of the maximum acceptable size, it is assumed the Exponent is entirely in the Crypto RAM and its length is equal to the Modulus one.
- Otherwise, the Exponent is entirely out of the Crypto RAM and so the computation do not depend on its length.

Table 37-54. Maximum Exponentiation Sizes

Option Specified	Maximum Modulus Size (bytes)	Maximum Modulus Size (bits)
Exponent in Crypto RAM, 1 bit window	576	4608
Exponent in Crypto RAM, 2 bits window	504	4032
Exponent in Crypto RAM, 3 bits window	400	3200
Exponent in Crypto RAM, 4 bits window	284	2272
Exponent not in Crypto RAM, 1 bit window	672	5376
Exponent not in Crypto RAM, 2 bits window	576	4608
Exponent not in Crypto RAM, 3 bits window	448	3584
Exponent not in Crypto RAM, 4 bits window	308	2464

37.3.5.2.9 Status Returned Values

Table 37-55. ExpMod Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	Service functioned correctly

37.3.5.3 Probable Prime Generation (Using Rabin-Miller)

37.3.5.3.1 Purpose

This service is used to perform probable prime generation or test. This service processes integers in GF(p) only.

The options available for this service are:

- Choice of the number of iterations of the Rabin-Miller test
- Generation or Test of a probable prime number
- Fast Implementation
- Regular Implementation
- Exponent Window Size

37.3.5.3.2 Additional Information

The Rabin-Miller test is a probable-primality testing algorithm. As a consequence, the primality of the generated number is not guaranteed at 100%, however, numerous publications have been issued explaining how to estimate the probability of getting a composite number, giving the size of the number and the number of iterations (the T parameter).

Useful information can be found in the *“Handbook of Applied Cryptography (Discrete Mathematics and Its Applications” by Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone*, in the following sections:

- 4.2.3. “Rabin-Miller Test”

- 4.4. “Prime Number Generation”

37.3.5.3.3 How to Use the Service

37.3.5.3.4 Description

This service processes a test for probable primality or a generation of a probable prime number.

Note: When using this service be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

This service processes one of the following operations: CheckProbablePrimality(N)

or

N = GenerateProbablePrimeFromSeed (NSeed)

In this computation, the following parameters need to be provided:

- N the input number (pointed by {nu1NBase,u2NLength +4})
  - If the requested operation is a test, it is untouched after the operation.
  - If the requested operation is a generation and a probable prime number was found before reaching the Maximum Increment, it contains the resulting probable prime after the operation.
  - If the requested operation was a generation and Maximum Increment was reached before a probable prime number was found, it contains no relevant information.
- Cns as a workspace (pointed by {nu1CnsBase,u2NLength +12})
- Rnd as a workspace (pointed by {nu1RndBase,u2NLength +16})
- Precomp the precomputation workspace (pointed by{nu1PrecompBase,PrecompLen})
- Exp as a workspace (pointed by {pfu1ExpBase,u2ExpLength +4})
- u1MillerRabinIterations the number of Miller Rabin Iterations requested
- u2MaxIncrement, maximum increment of the number in case of probable prime generation

The length PrecompLen depends on the lengths and options chosen; its calculus is detailed in Options below.

The service name for this operation is `PrimeGen`.

37.3.5.3.5 Parameters Definition

Table 37-56. PrimeGen Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1NBase(1)	nu1	I	Crypto RAM	u2NLength + 4	Base of N Number to test or Seed for the generation	Base of N unchanged if test or generation result(2)
nu1CnsBase	nu1	I	Crypto RAM	u2NLength + 12	Base of Cns as a workspace	Base of Cns workspace corrupted
u2NLength	u2	I	–	–	Length of N	Length of N
nu1RndBase	nu1	I	Crypto RAM	Max (u2NLength + 16,64)	Internal Workspace	Internal Workspace corrupted
nu1PrecompBase	nu1	I	Crypto RAM	See Options below	Base of Precomp workspace	Base of Precomp workspace corrupted
nu1RBase(2)	nu1	–	Crypto RAM	–	–	–
nu1ExpBase(3)	nu1	I	Crypto RAM	u2NLength + 4	Base of Exponent (R)	Base of Exponent (R)
u1MillerRabin-Iterations	u1	I	–	–	Miller Rabin’s T parameter	Miller Rabin’s T parameter
u2MaxIncrement	u2	I	–	–	Maximum Increment(4)	Maximum Increment

Notes:

1. This zone contains the number to be either tested or used as a seed for generation. It has to be provided with one zero word on the MSB side. This area has supplementary constraints (see the following Important note).

1. This parameter does not have to be provided and is used as an internal value for computing the reduction's constant.
2. The area {nu1ExpBase, u2NLength + 4} must be entirely in the Crypto RAM.
3. The generation starts from the number in {nu1NBase,u2NLength + 4} and increments it until a number is found as probable prime. However, the generation may stop for two reasons: The number has been incremented in a way it is bigger than <u2NLength> bytes, or the original number has been incremented by more than <u2MaxIncrement>.

In case of probable prime generation, ensure that the addition of NSeed and Maximum Increment is not a number with more bytes than u2NLength, as this would produce an overflow.

Important:
One additional word is used on the LSB side of the NBase parameter; this word is restored at the end of the calculus. As a consequence, the parameter nu1NBase must never be at the beginning of the Crypto RAM, but at least at one word from the beginning.

One additional word is used on the MSB side of the NBase parameter; this word is not corrupted. As a consequence the Area {nu1NBase, u2NLength} must not be at the end of the Crypto RAM but at least at one word from the end.

Prime numbers of a size lower than 96 bits (three 32-bit words) cannot be generated or tested by this service.

37.3.5.3.6 Options

Some of the Prime Generation options configure the Modular Exponentiation steps and so are very similar to the Modular Exponentiation options.

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Operation Option described in Table 37-57
- the mandatory Calculus Mode Option described in Table 37-58
- the mandatory Window Size Option described in Table 37-59

The u2Options number is calculated by an "Inclusive OR" of the options. Some Examples in C language are:

- Operation: Probable Prime Testing with Fast Modular Exponentiation and the window size equal to 1
  ```
  PUKCL(u2Options) = PUKCL_PRIMEGEN_TEST | PUKCL_EXPMOD_FASTRSA |
  PUKCL_EXPMOD_WINDOWSIZE_1;
  ```
- Operation: Probable Prime Generate with Regular Modular Exponentiation and the window size equal to 2
  ```
  PUKCL(u2Options) = PUKCL_EXPMOD_REGULARRSA | PUKCL_EXPMOD_WINDOWSIZE_2;
  ```

The following table describes the PrimeGen service features available from the various options.

Table 37-57. PrimeGen Service Options

Option	Method Used
PUKCL_PRIMEGEN_TEST	This option is used to specify that only tests will be made on the provided number. When this option is not specified, a prime generation algorithm is selected, starting from the given seed and incrementing it.
PUKCL_EXPMOD_WINDOWSIZE_1,2,3 or 4	Depending on this option, different bit-window sizes will be used. For long exponents, the bigger the window, the faster the computation. However, this has also an impact on the size of the precomputations table.

For this service, two exclusive Calculus Modes are possible. The following table describes the Calculus Mode Options.

Table 37-58. PrimeGen Service Calculus Mode Options

Option	Explanation
PUKCL_EXPMOD_FASTRSA	Perform a Fast computation.
PUKCL_EXPMOD_REGULARRSA	Performs a Regular computation, slower than the Fast version, but using regular calculus methods.

The length of the Precomp area depends on the window size W and u2NLength. The Precomp area length is:

PrecompLen = max(2*(u2NLength + 4) + 2W-1 * (u2NLength + 4), u2NLength + 8 + 64) + 8

Note: Please calculate precisely the length PrecompLen with the formula and the `max()` macro, which takes a maximum of two values.

The following table shows the size of the precomputation workspace (PrecompLen), depending on the chosen window size option.

Table 37-59. PrimeGen Service Precomputation Space Size

Option Specified	Size of the PrecompBase Workspace (bytes)	Content of the Workspace
PUKCL_EXPMOD_WINDOWSIZE_1	max(3*(u2NLength + 4), u2NLength + 72) + 8	x
PUKCL_EXPMOD_WINDOWSIZE_2	max(4*(u2NLength + 4), u2NLength + 72) + 8	$x\ x^3$
PUKCL_EXPMOD_WINDOWSIZE_3	max(6*(u2NLength + 4), u2NLength + 72) + 8	$x\ x^3\ x^5\ x^7$
PUKCL_EXPMOD_WINDOWSIZE_4	max(10*(u2NLength + 4) u2NLength + 72) + 8	$x\ x^3\ x^5\ x^7\ x^9\ x^{11}\ x^{13}\ x^{15}$

The following table provides the maximum sizes for the Prime Generation depending on the window size.

Table 37-60. PrimeGen Service Maximum Sizes

Characteristics of the Operation	Maximum Prime Sizes (bits)
1 bit window	4608
2 bits window	4032
3 bits window	3200
4 bits window	2272

37.3.5.3.7 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip PUKCL(u2Option) =...;
// Depending on the option specified, not all fields must be filled
PUKCL_PrimeGen(nu1NBase) = <Base of the ram location of N>;
PUKCL_PrimeGen(u2NLength)  =  <Length  of  N>;
PUKCL_PrimeGen(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_PrimeGen(nu1PrecompBase) = <Base of the ram location of Precomp>;
PUKCL_PrimeGen(pfu1ExpBase) = <Base of the location of Exp>;
PUKCL_PrimeGen(u2ExpLength) = <Length of Exp>;
PUKCL_PrimeGen(u1MillerRabinIterations) = <Number of iterations>;
PUKCL_PrimeGen(u2MaxIncrement) = <Maximum Increment>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(PrimeGen, pvPUKCLParam);
```

```
if (PUKCL_Param.Status == PUKCL_NUMBER_IS_PRIME)
            {
            // The number is probably prime
            ...
            }
else if (PUKCL_Param.Status == PUKCL_NUMBER_IS_NOT_PRIME)
            {
            // The number is not prime
            ...
            }
else // Manage the error
```

37.3.5.3.8 Constraints

The following combinations of input values must be avoided in the case of a modular reduction 'alone', meaning that it has not been requested as an option of any other service:

- nu1NBase,nu1CnsBase, nu1RndBase,nu1PrecompBase,nu1ExpBase are not aligned on 32-bit boundaries
- {nu1NBase, u2NLength + 4}, {nu1CnsBase, u2NLength + 12}, {nu1RndBase, u2NLength +12}, {nu1PrecompBase, <PrecompLength>} are not in Crypto RAM
- u2NLength is either: < 12, > 0xffc or not a 32-bit length
- Both PUKCL_EXPMOD_REGULARRSA and PUKCL_EXPMOD_FASTRSA are specified.
- {nu1PrecompBase,<PrecompLength>} overlaps with either: {nu1NBase, u2NLength + 4},{nu1CnsBase, u2NLength + 12} {nu1RndBase, u2NLength + 12} or {nu1ExpBase, u2ExpLength + 4}
- {nu1RndBase,3*u2NLength + 24} overlaps with either: {nu1NBase, u2NLength + 4},{nu1CnsBase, u2NLength + 12} {nu1XBase, u2NLength + 12} or {nu1ExpBase, u2ExpLength + 4}
- {nu1NBase, u2NLength + 4} overlaps {nu1CnsBase, u2NLength +12}

37.3.5.3.9 Status Returned Values

Table 37-61. PrimeGen Service Return Codes

Returned Status	Importance	Meaning
PUKCL_NUMBER_IS_PRIME	Information	The generated or tested number has been detected as probably prime.
PUKCL_NUMBER_IS_NOT_PRIME	Information	The generated or tested number has been detected as composite.

37.3.5.4 Modular Exponentiation (With CRT)

37.3.5.4.1 Purpose

The purpose of this service is to perform the Modular Exponentiation with the Chinese Remainders Theorem (CRT). This service processes integers in GF(p) only.

The options available for this service are:

- Fast implementation
- Regular implementation
- Exponent is located in Crypto RAM or not
- Exponent window size

37.3.5.4.2 How to Use the Service

37.3.5.4.3 Description

This service processes a Modular Exponentiation with the Chinese Remainder Theorem:

$R = X^D mod(N)$ *with* $N = P * Q$

Important: For this service, be sure to follow the directives given for the RSA implementation on the chip you use.

This service requires that the modulus N is the product of two co-primes P and Q and that the decryption exponents D is co-prime with the product ((P-1)*(Q-1)).

The Input data are P, Q, EP, EQ, Rvalue, and X. P and Q are the co-primes so that N = P*Q.

X is the number to exponentiate.

EP, EQ and Rval are calculated as follows:

$EP = Dmod(P - 1)\ EQ = Dmod(Q - 1)\ Rval = P^{-1}mod(Q)$

In some cases, the decryption exponent D may not be available and the encryption exponent E may be available instead. The possibilities to calculate the parameters are:

- Calculate D from E with the formula:
  $D = E^{-1}mod((P - 1) \times (Q - 1))$
- Calculate the parameters from E:
  $EP = E^{-1}mod(P - 1)\ EQ = E^{-1}mod(Q - 1)\ Rval = P^{-1}mod(Q)$

In this computation, the following parameters need to be provided:

- X the input number (pointed by {nu1XBase,2*u2ModLength +16})
- P and Q the primes (pointed by {nu1ModBase,2*u2ModLength +8}).
- EP and EQ the reduced exponents (pointed by {pfu1ExpBase,2*u2ExpLength +8})
- Rval and Precomp (pointed by{nu1PrecompBase,RAndPrecompLen})
- Blinding the exponent blinding value (provided inu1Blinding)

The length RAndPrecompLen depends on the lengths and options chosen; its calculus is detailed in Options below.

The service for this operation is CRT.

Note: The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of P or Q must be at least three 32-bit words.

37.3.5.4.4 Parameters Definition

The following table shows the parameter block for the CRT service.

Many parameters have complex placement in memory; therefore, detailed figures are provided in CRT Service Placement below.

Table 37-62. CRT Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
u2Options	u2	I	–	–	Options (see below)	Options (see below)
nu1ModBase	nu1	I	Crypto RAM	2*u2ModLength + 8	Base of P, Q	Base of P, Q untouched
u2ModLength	u2	I	–	–	Length of P or Q greater than or equal to 12	Length of P or Q
nu1XBase(1)	nu1	I	Crypto RAM	2*u2ModLength + 16	Base of X	Base of X Filled with the result
nu1PrecompBase	nu1	I	Crypto RAM	See Options below	Base of Rvalue and Pre computations workspace	Corrupted
pfu1ExpBase(2)	pfu1	I	Any place	2*u2ExpLength + 8	Base of EP, EQ	Base of EP, EQ untouched

.....continued						
Parameter	**Type**	**Direction**	**Location**	**Data Length**	**Before Executing the Service**	**After Executing the Service**
u2ExpLength	u2	I	–	–	Significant length of EP or EQ	Significant length of EP or EQ
u1Blinding(3)	u4	I	–	–	Exponent unblinding value	Exponent unblinding value

Notes:

1. This zone contains the number to be exponentiated (u2ModLength bytes) and is used during the computations as a workspace (four 32-bit words longer than the number to be exponentiated). At the end of the computation, it contains the correct result of the operation.
2. If the PUKCL_EXPMOD_EXPINPUKCCRAM option is not set, the location of the exponent MUST NOT be placed in the Crypto RAM, even partially.
3. It is possible to mask the exponent in memory using a 32-bit XOR mask value. Be aware that not only the exponent, but also the supplemental spill word has to be masked. If masking is not desired, the parameter must be set to 0.

37.3.5.4.5 Options

Most of the CRT options configure the Modular Exponentiation steps of the CRT and so are very similar to the Fast Modular Exponentiation options.

The options are set by the u2Options input parameter, which is composed of:

- the mandatory Calculus Mode Option described in Table 37-63
- the mandatory Window Size Option described in Table 37-64
- the indication of the presence of the exponent in Crypto RAM

Important: Please check precisely if one part of the exponent area (containing EP and EQ) is in Crypto RAM. If this is the case, the PUKCL_EXPMOD_EXPINPUKCCRAM option must be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some Examples in C language are:

- Operation: CRT using the Fast Modular Exponentiation with the window size equal to 1 and with no part of the Exponent area in the Crypto RAM
  ```
  PUKCL(u2Options) = PUKCL_EXPMOD_FASTRSA | PUKCL_EXPMOD_WINDOWSIZE_1;
  ```
- Operation:CRT using the Regular Modular Exponentiation with the window size equal to 2 and with one part the Exponent area in the Crypto RAM
  ```
  PUKCL(u2Options) = PUKCL_EXPMOD_REGULARRSA | PUKCL_EXPMOD_WINDOWSIZE_2 |
  PUKCL_EXPMOD_EXPINPUKCCRAM;
  ```

For this service, two exclusive Calculus Modes for the Modular Exponentiation steps of the CRT are possible. The following table describes the Calculus Mode Options.

Table 37-63. CRT Service Calculus Mode Options

Option	Explanation
PUKCL_EXPMOD_FASTRSA	Perform a Fast computation.
PUKCL_EXPMOD_REGULARRSA	Performs a Regular computation, slower than the Fast version, but using regular calculus methods.

For this service, four window sizes for the Modular Exponentiation Steps are possible. The window size in bits is those of the windowing method used for the exponent.

The choice of the window size is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation workspace.
- Increasing the window size reduces the computation time (may not be relevant for very small exponents). The length of the Rval and Precomp area depends on the window size W and u2ModLength.

The Rval and Precomp area length is:

RandPrecompLen = 4 * (u2ModLength + 4) + max(64 , $2^{(W-1)}$ * (u2ModLength + 4)) + 8

Important: Please calculate precisely the length RandPrecompLen with the formula and the `max()` macro, which takes the maximum of two values.

The following table shows the size of the Rval and Precomp area, depending on the chosen window size option.

Table 37-64. CRT Service Window Size Options and Rval and Precomp Area Size

Option Specified	Size of the Rval and Precomp Area (bytes)	Precomputation Values
PUKCL_EXPMOD_WINDOWSIZE_1	4*(u2ModLength + 4) + max(64 , (u2ModLength + 4)) + 8	x
PUKCL_EXPMOD_WINDOWSIZE_2	4*(u2ModLength + 4) + max(64 , 2*(u2ModLength + 4)) + 8	x x^3
PUKCL_EXPMOD_WINDOWSIZE_3	4*(u2ModLength + 4) + max(64 , 4*(u2ModLength + 4)) + 8	x x^3 x^5 x^7
PUKCL_EXPMOD_WINDOWSIZE_4	10*(u2ModLength + 4) + max(64 , 8*(u2ModLength + 4)) + 8	x x^3 x^5 x^7 x^9 x^{11} x^{13} x^{15}

The exponent area can be located in RAM or in the data space. If one part of the exponent area is in Crypto RAM this must be mandatory signaled by using the PUKCL_EXPMOD_EXPINPUKCCRAM option.

The following table describes this option.

Table 37-65. CRT Service Crypto RAM Option Exponent Area

Option	Purpose
PUKCL_EXPMOD_EXPINPUKCCRAM	The exponent area can be read from any data space of memory, including Crypto RAM. When at least one word the exponent is in Crypto RAM, this option has to be set.

37.3.5.4.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL(u2Option) =...;

// Depending on the option specified, not all fields must be filled PUKCL_CRT(nu1ModBase) =
<Base of the ram location of P and Q>; PUKCL_CRT(u2ModLength) = <Length of P or Q>;
PUKCL_CRT(nu1XBase) = <Base of the ram location of X>;
PUKCL_CRT(nu1PrecompBase) = <Base of the ram location of RVal and Precomp>;
PUKCL_CRT(pfu1ExpBase) = <Base of the ram location of EP and EQ>;
PUKCL_CRT(u2ExpLength) = <Length of EP or EQ>;
PUKCL_CRT(u1Blinding) = <Blinding value>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(CRT, pvPUKCLParam);
if (PUKCL_Param.Status == PUKCL_OK)
            {
            // operation has been performed correctly
            ...
```

```
            }
else // Manage the error
```

37.3.5.4.7 Constraints

The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1XBase, nu1PrecompBase, pfu1ExpBase are not aligned on 32-bit boundaries
- {nu1XBase, 2*u2ModLength + 16}, {nu1ModBase, 2*u2ModLength + 8},{nu1PrecompBase,<PrecompLength>} are not in Crypto RAM
- {nu1ExpBase,2*u2ExpLength + 8} is not in Crypto RAM and PUKCL_EXPMOD_EXPINPUKCCRAM is specified
- u2ModLength or u2ExpLength are either: < 4, > 0xffc or not a 32-bit length
- None or both PUKCL_EXPMOD_REGULARRSA and PUKCL_EXPMOD_FASTRSA are specified.
- {nu1XBase,2*u2ModLength + 16} overlaps with either: {nu1ModBase, 2*u2ModLength +8},{nu1PrecompBase, <PrecompLength>} or {pfu1ExpBase, 2*u2ExpLength + 8}
- {nu1ModBase,2*u2ModLength + 8} overlaps with either: {nu1PrecompBase, <PrecompLength>} or {pfu1ExpBase, 2*u2ExpLength + 8}
- {nu1PrecompBase, <PrecompLength>} overlaps {pfu1ExpBase, 2*u2ExpLength +8}

37.3.5.4.8 CRT Service Parameter Placement

The parameters' placements are described in detail in the following figures.

Figure 37-2. Modulus P and Q in {nu1ModBase, 2*u2ModLength + 8}

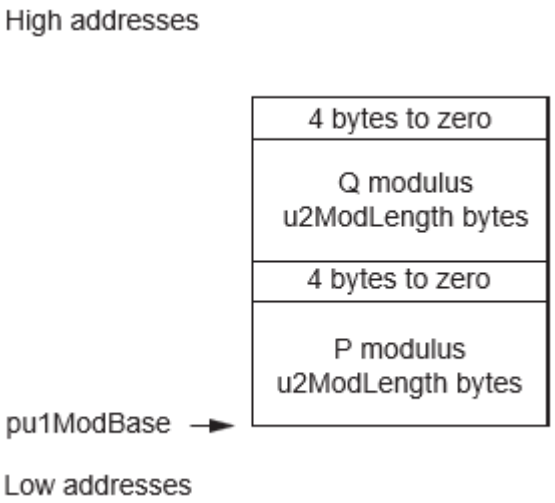


Figure 37-3. Value X in {nu1XBase, 2*u2ModLength + 16}

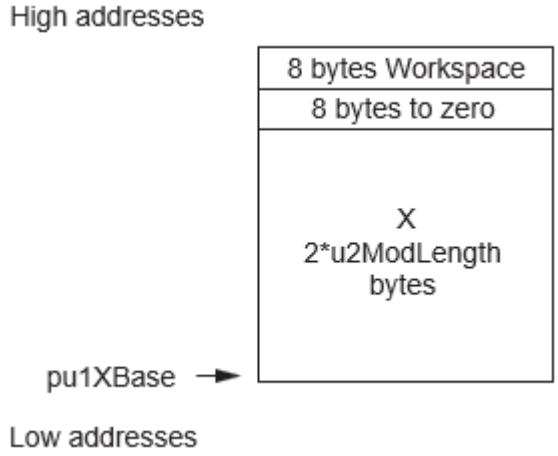


Figure 37-4. Exponents EP and EQ in {fnu1ExpBase, 2*u2ExpLength + 8}

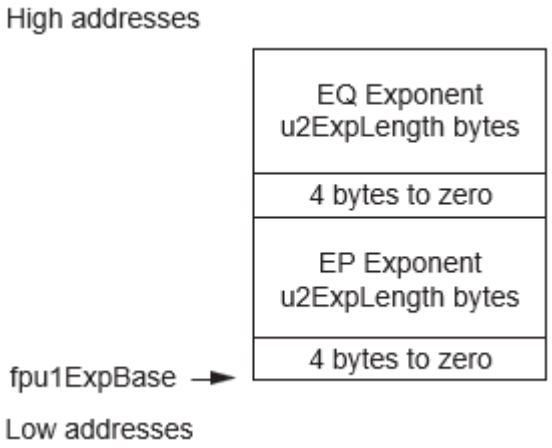
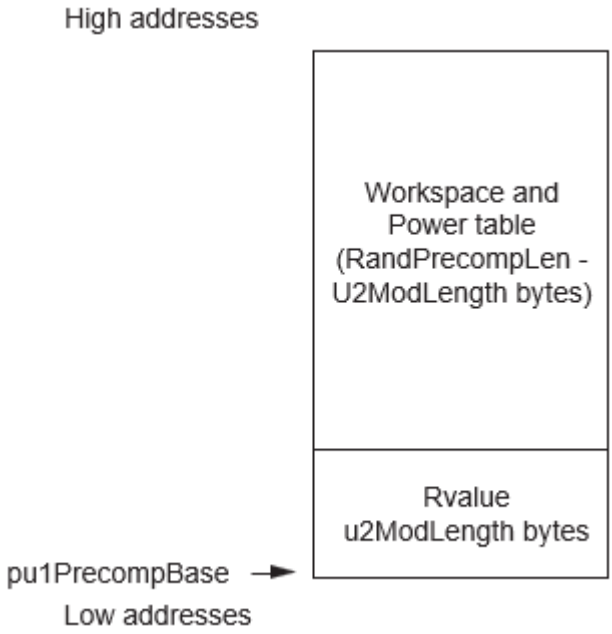


Figure 37-5. Value Rval and Precomp in {nu1PrecompBase, RandPrecompLen}



37.3.5.4.9 CRT Service Modular Exponentiation Maximum Size

The following table details the maximum size in bits of P or Q, of N and of EP or EQ.

- The maximum size in bits of P or Q equals:
  <Max Size Bits P> = <Max Size Bits Q> = 8 * <Max u2ModLength bytes>
- The maximum size in bits of N=P*Q equals:
  <Max Size Bits N> = 2 * <Max Size Bits P>
- The maximum size in bits of EP or EQ equals:
  <Max Size Bits EP> = <Max Size Bits EQ> = 8 * <Max u2ExpLength bytes>
- In case of the PUKCL_EXPMOD_EXPINPUKCCRAM option is specified, for the computation of the maximum acceptable size, it is assumed the Exponent is entirely in the Crypto RAM and its length equal the Modulus one.
- Otherwise, the Exponent is entirely out of the Crypto RAM and so the computation do not depend on its length.

Table 37-66. CRT Service Maximum Sizes

Characteristics of the Operation	P or Q Max Bit Sizes	N Max Bit Sizes	EP or EQ Max Bit Sizes
Exponent in Crypto RAM, 1 bit window	2912	5824	2912
Exponent in Crypto RAM, 2 bits window	2688	5376	2688
Exponent in Crypto RAM, 3 bits window	2464	4928	2464
Exponent in Crypto RAM, 4 bits window	2304	4608	2304
Exponent not in Crypto RAM, 1 bit window	3584	7168	<application dependent>
Exponent not in Crypto RAM, 2 bits window	3232	6464	<application dependent>
Exponent not in Crypto RAM, 3 bits window	2912	5824	<application dependent>
Exponent not in Crypto RAM, 4 bits window	2688	5376	<application dependent>

37.3.5.4.10 Status Returned Values

Table 37-67. CRT Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	Information	Service functioned correctly

37.3.6 Elliptic Curves Over GF(p) Services

This section provides a complete description of the currently available elliptic curve over Prime Fields services. These services process integers in GF(p) only.

The offered services cover the basic operations over elliptic curves such as:

- Adding two points over a curve
- Doubling a point over a curve
- Multiplying a point by an integral constant
- Converting a point's projective coordinates (resulting from a doubling or an addition) to the affine coordinates, and oppositely converting a point's affine coordinates to the projective coordinates.
- Testing the point presence on the curve.

Additionally, some higher level services covering the needs for signature generation and verification are offered:

- Generating an ECDSA signature (compliant with FIPS186-2)
- Verifying an ECDSA signature (compliant with FIPS186-2) The supported curves use the following curve equation:

$Y^2 = X^3 + aX + b$

37.3.6.1 Coordinate Systems

Related Links

37.3.5.1. Modular Reduction

37.3.6.1.1 General Considerations

In this implementation, several choices have been made related to the coordinate systems managed by the elliptic curve primitives.

There are two systems currently managed by the library:

- Affine Coordinates System where each curve point has two coordinates (X, Y)
- Projective Coordinates System where each point is represented with three coordinates (X,Y, Z)

Converting from the affine coordinates system to a projective coordinates system is performed by extending its representation with Z = 1:

$(X, Y) \Rightarrow (X, Y, Z= 1)$

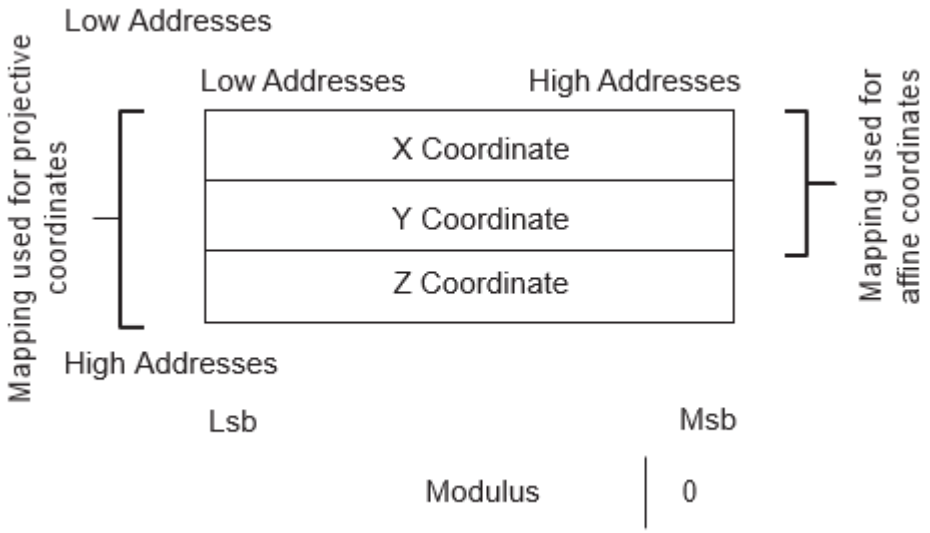
Converting from a projective coordinate to an affine one is a service offered by the PUKCL. The formula to perform this conversion is:

$(X, Y, Z) \Rightarrow (X / Z^2, Y / Z^3)$

37.3.6.1.2 Points Representations

Depending on the representation (Projective or Affine), points are represented tn memory, as shown in the following figure.

Figure 37-6. Points Representation in Memory



In this figure, the modulus is represented as a reference, and to show that coordinates are always to be provided on the length of the modulus plus one 32-bit word.

The different types of representations are as follows:

$$\text{Affine representation} \qquad Pt = \begin{bmatrix} X_{Affine} < P \times 2^{15} \\ Y_{Affine} < P \times 2^{15} \end{bmatrix}$$

$$\text{Projective representation} \qquad Pt = \begin{bmatrix} X_{Projective} < P \times 2^{15} \\ Y_{Projective} < P \times 2^{15} \\ Z_{Projective} < P \times 2^{15} \end{bmatrix}$$

Notes:

1. The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of the modulus must be at least three 32-bit words.
2. In some cases the point can be the infinite point. In this case, it is represented with its Z coordinates equal or congruent to zero.

37.3.6.1.3 Modulus and Modular Constant Parameters

In most of the services the following parameters must be provided:

- P the Modulus (often pointed by {nu1ModBase,u2ModLength + 4}): This parameter contains the Modulus Integer prime P defining the Galois Field used in points coordinates computations. The Modulus must be u2ModLength bytes long, while having a supplemental zeroed 32-bit word on the MSB side.

Note: Most of the Elliptic Curve computations are reduced modulo P. In many functions the reductions are made with the Fast Reduction.

- Cns the Modular Constant (often pointed by {nu1CnsBase,u2ModLength + 12}): This parameter contains the Modular Constant associated to the Modulus

Important: The Modular Constant must be calculated before using the GF(p) Elliptic Curves functions by a call to the Setup for Modular Reductions with the GF(p) option (see *Modular Reduction Setup* in the *Modular Reduction* from Related Links).

37.3.6.2 Point Addition

37.3.6.2.1 Purpose

This service is used to perform a point addition, based on a given elliptic curve over GF(p). Please note that:

- This service is not intended to add the same point twice. In this particular case, use the doubling service (see 37.3.6.4. Fast Point Doubling).

37.3.6.2.2 How to Use the Service

37.3.6.2.3 Description

The operation performed is:

$Pt_C = Pt_A + Pt_B$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3*u2ModLength + 12}). This point can be the Infinite Point.
- B the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBBase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 5*u2ModLength +32}

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This Point can be the Infinite Point.

The service name for this operation is `ZpEccAddFast`. This service uses Fast mode and Fast Modular Reduction for computations.

Important: Before using this service, ensure that the constant Cns has been calculated with the Setup of the Modular Reduction functions.

37.3.6.2.4 Parameters Definition

Table 37-68. ZpEccAddFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of Modulus P	Base of Modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulo	Length of modulo
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)

.....continued						
Parameter	**Type**	**Direction**	**Location**	**Data Length**	**Before Executing the Service**	**After Executing the Service**
nu1PointBBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point B (projective coordinates)	Input point B
nu1Workspace	nu1	I	Crypto RAM	5*u2ModLength + 32	–	Corrupted workspace

37.3.6.2.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _ZpEccAdd(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccAdd(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEccAdd(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccAdd(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEccAdd(nu1PointBBase) = <Base of the ram location of the B point>;
PUKCL _ZpEccAdd(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccAddFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.2.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointBBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PointBBase, 3*u2ModLength + 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8},{nu1PointABase, 3*u2ModLength + 12}, {nu1PointBBase, 3*u2ModLength + 12} and {nu1Workspace, 5*u2ModLength + 32}

37.3.6.2.7 Status Returned Values

Table 37-69. ZpEccAddFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.6.3 Point Addition and Subtraction

37.3.6.3.1 Purpose

This service is used to perform a point addition and point subtraction, based on a given elliptic curve over GF(p). Please note that:

- This service is not intended to add the same point twice. In this particular case, use the doubling service (see 37.3.6.4. Fast Point Doubling).

37.3.6.3.2 How to Use the Service

37.3.6.3.3 Description

The operation performed is:

$Pt_C = Pt_A \pm Pt_B$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3*u2ModLength + 12}). This point can be the Infinite Point.
- B the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBBase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 5*u2ModLength +32}
- The operator filled with the operation to perform (Addition or Subtraction)

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This Point can be the Infinite Point.

The service name for this operation is `ZpEccAddSubFast`. This service uses Fast mode and Fast Modular Reduction for computations.

Note: Before using this service, ensure that the constant Cns has been calculated with the setup of the modular reduction functions.

37.3.6.3.4 Parameters Definition

Table 37-70. ZpEccAddSubFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of Modulus P	Base of Modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulo	Length of modulo
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1PointBBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point B (projective coordinates)	Input point B
u2Operator	u2	I	-	-	Addition or Subtraction	Addition or Subtraction
nu1Workspace	nu1	I	Crypto RAM	5*u2ModLength + 32	–	Corrupted workspace

37.3.6.3.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL (u2Option) = 0;

PUKCL _ZpEccAddSub(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccAddSub(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEccAddSub(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccAddSub(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEccAddSub(nu1PointBBase) = <Base of the ram location of the B point>;
PUKCL _ZpEccAddSub(nu1Workspace) = <Base of the ram location of the workspace>;
```

```
PUKCL _ZpEccAddSub(u2Operator) = <Operation to perform (PUKCL_ZPECCADD or PUKCL_ZPECCSUB)>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccAddSubFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
        {
        ...
        }
else // Manage the error
```

37.3.6.3.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointBBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PointBBase, 3*u2ModLength + 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8},{nu1PointABase, 3*u2ModLength + 12}, {nu1PointBBase, 3*u2ModLength + 12} and {nu1Workspace, 5*u2ModLength + 32}

37.3.6.3.7 Status Returned Values

Table 37-71. ZpEccAddFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.6.4 Fast Point Doubling

37.3.6.4.1 Purpose

This service is used to perform a Point Doubling, based on a given elliptic curve over GF(p).

37.3.6.4.2 How to Use the Service

37.3.6.4.3 Description

These two services process the Point Doubling:

$Pt_C = 2 \times Pt_A$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4*u2ModLength +28}
- The a parameter relative to the elliptic curve (pointed by {nu1ABase,u2ModLength +4})
- The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the same location than the input point A. This point can be the Infinite Point.

The service name for this operation is `ZpEccDblFast`. This service uses Fast mode and Fast Modular Reduction for computations.

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reduction service.

37.3.6.4.4 Parameters Definition

Table 37-72. ZpEccDblFastService

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1ABase	u2	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Parameter a of the elliptic curve
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 28	–	Corrupted workspace

37.3.6.4.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL (u2Option) = 0;

PUKCL _ZpEccDbl(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccDbl(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccDbl(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEccDbl(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEccDbl(nu1ABase) = <Base of the a parameter of the elliptic curve>;
PUKCL _ZpEccDbl(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccDblFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.4.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1ABase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1ABase, u2ModLength + 4} and {nu1Workspace, 4*u2ModLength + 28}

37.3.6.4.7 Status Returned Values

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.6.5 Fast Multiplying by a Scalar Number of a Point

37.3.6.5.1 Purpose

This service is used to multiply a point by an integral constant K on a given elliptic curve over GF(p).

37.3.6.5.2 How to Use the Service

37.3.6.5.3 Description

These two services process the Multiplying by a scalar number:

$Pt_C = K \times Pt_A$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8*u2ModLength +44}
- The a parameter relative to the elliptic curve (pointed by {nu1ABase,u2ModLength +4})
- K the scalar number (pointed by {nu1ScalarNumber,u2ScalarLength +4})

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This point can be the Infinite Point.

The service name for this operation is `ZpEccMulFast`. This service uses Fast mode and Fast Modular Reduction for computations.

Note: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reduction service.

37.3.6.5.4 Parameters Definition

Table 37-73. ZpEccMulFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1KBase	nu1	I	Crypto RAM	u2KLength	Scalar number used to multiply the point A	Unchanged
u2KLength	u2	I	–	–	Length of scalar K	Length of scalar K
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1ABas	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	–	Corrupted workspace

37.3.6.5.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL (u2Option) = 0;
```

```
PUKCL _ZpEccMul(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccMul(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccMul(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEccMul(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEccMul(nu1ABase) = <Base of the ram location of the parameter A of the elliptic
curve>;
PUKCL _ZpEccMul(nu1KBase) = <Base of the ram location of the scalar number>;
PUKCL _ZpEccMul(nu1Workspace) = <Base of the ram location of the workspace>;
PUKCL_ZpEccMul(u2KLength) = <Byte length of the Scalar Number K>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccMulFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.5.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase,nu1CnsBase, nu1PointABase, nu1ABase, nu1ScalarNumber, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12}, {nu1ABase, u2ModLength + 4}, {nu1ScalarNumber, u2ScalarLength} or {nu1Workspace, 8*u2ModLength + 44} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1ABase, u2ModLength + 4}, {nu1ScalarNumber, u2ScalarLength} and {nu1Workspace, 8*u2ModLength + 44}

37.3.6.5.7 Status Returned Values

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.6.6 Quick Dual Multiplying by Two Scalar Numbers and Two Points

37.3.6.6.1 Purpose

This service is used to multiply two points by two integral constants K1 and K2, and then provide the addition of these multiplications results.

Important: This service has a quick implementation without additional security.

37.3.6.6.2 How to Use the Service

37.3.6.6.3 Description

This service processes the dual Multiplying by two scalar numbers:

$PtC = K_1 \times Pt_A + K_2 \times Pt_B$

In this computation, the following parameters need to be provided:

- A the first input point is filled in projective coordinates (X,Y,Z) (pointed by {pu1PointABase,(3*(u2ModLength + 4)) * (2(WA-2))}). This point can be the Infinite Point.
- B the 2nd input point is filled in projective coordinates (X,Y,Z) (pointed by {pu1PointBBase,(3*(u2ModLength + 4)) * (2(WB-2))}). This point can be the Infinite Point.

- P the modulus filled and Cns the Fast Modular Constant filled (pointed by {pu1ModCnsBase,2*u2ModLength + 16})
- The a parameter filled and the workspace not initialized (pointed by {pu1AWorkBase, 9*u2ModLength +48}
- KAB the scalar numbers (pointed by {pu1KABBase, 2*u2KLength +8})
- The options are set by the u2Options input parameter, which is composed of:
  - wA: Size of window for Point A between 2 and15
  - wB: Size of window for Point B between 2 and15
  - PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM flag: to set only if the scalars are entirely in Classic RAM with no part in PUKCC RAM

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at (pu1AWorkBase + u2ModLength + 4). This point can be the Infinite Point.

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reduction service.

37.3.6.6.4 Parameters Definition

WA is the Point A window size and WB is the Point B window size (see Options below for details).

Important: Please calculate precisely the length of areas with the formulas. Ensure that the pu1 type is a pointer on 4 bytes and contains the full address (see 37.3.3.4. Aligned Significant Length).

Table 37-74. ZpEccQuickDualMulFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
pu1ModCnsBase	pu1	I	Crypto RAM	2 * u2ModLength + 16	Base of modulus P, Base of Cns	Base of modulus P, Base of Cns
u2Option	u2	I	–	–	Option related to the called service (see below)	–
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
pu1KABBase	pu1	I	Any RAM	2 * u2KLength + 8	Scalar numbers used to multiply the points A and B	Unchanged
u2KLength	u2	I	–	–	Length of scalars KA and KB	Length of scalars KA and KB
pu1PointABase	pu1	I/O	Crypto RAM	(3*(u2ModLength + 4)) * ($2^{(WA-2)}$) (1)	Input point A (projective coordinates)	Unchanged
pu1PointBBase	pu1	I	Crypto RAM	(3*(u2ModLength + 4)) * ($2^{(WB-2)}$) (2)	Input point B (projective coordinates)	Unchanged

.....continued						
Parameter	**Type**	**Direction**	**Location**	**Data Length**	**Before Executing the Service**	**After Executing the Service**
pu1AWorkBase	pu1	I	Crypto RAM	9*u2ModLength + 48	Parameter a of the elliptic curve	Resulting point C (projective coordinates) in pu1AWorkBase Base + u2ModLength + 4

Notes:

1. The precalculus table size for the point A is calculated from chosen window size “WA”.
2. The precalculus table size for the point B is calculated from chosen window size “WB”.

37.3.6.6.5 Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory windows sizes WA and WB
- the indication of the presence of the scalars in system RAM

Note: Please check precisely if one part of the scalars is in Crypto RAM. If this is the case, the PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM option must not be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some Examples in C language are:

- ```
  // Scalars are in system RAM
  // The Point A window size is 3
  // The Point B window size is 4
  PUKCL(u2Options) = PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM |
  PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(3) |
  PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(4);
  ```
- ```
  // Scalars are in the PUKCC Cryptographic RAM
  // The Point A window size is 2
  // The Point B window size is 5
  PUKCL(u2Options) = PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(2) |
  PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(5);
  ```

For this service, many window sizes are possible. The window sizes in bits are those of the windowing method used for the scalar multiplying.

The choice of the window sizes is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation table size.
- Increasing the window size to the optimum reduces the computation time.

The following table details the size of the point and the precomputation table, depending on the chosen window size option.

Table 37-75. ZpEccQuickDualMulFast Service Window Size Options and Precomputation Table Size

Option Specified	Size of the Point and the Precomputation Table
PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(WA) WA in [2, 15]	(3*(u2ModLength + 4)) * ($2^{(WA-2)}$)
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(WB) WB in [2, 15]	(3*(u2ModLength + 4)) * ($2^{(WB-2)}$)

The scalars can be located in PUKCC RAM or in system RAM. If both scalars are entirely in system RAM with no part in PUKCC RAM this can be signaled by using the option PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM. In all other cases this option must not be used.

The following table describes this option.

Table 37-76. ZpEccQuickDualMulFast Service System RAM Scalar Options

Option	Purpose
PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM	The scalars can be located in Crypto RAM or in system RAM. If both scalars are entirely in system RAM with no part in Crypto RAM this can be signaled by using this option . In all other cases this option must not be used.

37.3.6.6.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL(u2Option) = <Configure scalar numbers location and windows sizes>;
PUKCL_ZpEccQuickDualMulFast(pu1ModCnsBase) = <Base of the ram location of P and Cns>;
PUKCL_ZpEccQuickDualMulFast(u2ModLength) = <Byte length of P>;
PUKCL_ZpEccQuickDualMulFast(u2KLength) = <Byte length of scalars>;
PUKCL_ZpEccQuickDualMulFast(pu1PointABase) = <Base of the ram location of the A point>;
PUKCL_ZpEccQuickDualMulFast(pu1PointBBase) = <Base of the ram location of the B point>;
PUKCL_ZpEccQuickDualMulFast(pu1AWorkBase) = <Base of the ram location of the parameter A of
the elliptic curve and workspace>;
PUKCL_ZpEccQuickDualMulFast(pu1KABBase) = <Base of the ram location of the scalar numbers KA
and KB>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccQuickDualMulFast, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.6.7 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- pu1ModCnsBase,pu1PointABase, pu1PointBBase, pu1AWorkBase, pu1KABBase are not aligned on 32-bit boundaries
- {pu1ModCnsBase, 2*u2ModLength + 16}, {pu1PointABase, (3*(u2ModLength + 4)) *($2^{(WA-2)}$)}, {pu1PointBBase, (3*(u2ModLength + 4)) * ($2^{(WB-2)}$)} or { pu1AWorkBase, 9*u2ModLength + 48} are not in PUKCC RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- Alloverlapping between {pu1ModCnsBase, 2*u2ModLength + 16}, {pu1PointABase, (3*(u2ModLength + 4)) * ($2^{(WA-2)}$)}, {pu1PointBBase, (3*(u2ModLength + 4)) * ($2^{(WB-2)}$)} or {pu1AWorkBase, 9*u2ModLength + 48}.

37.3.6.6.8 Parameters Placement

The parameters' placement is described in the following figures.

Figure 37-7. Modulus P and Cns{pu1ModCnsBase, 2*u2ModLength + 16}

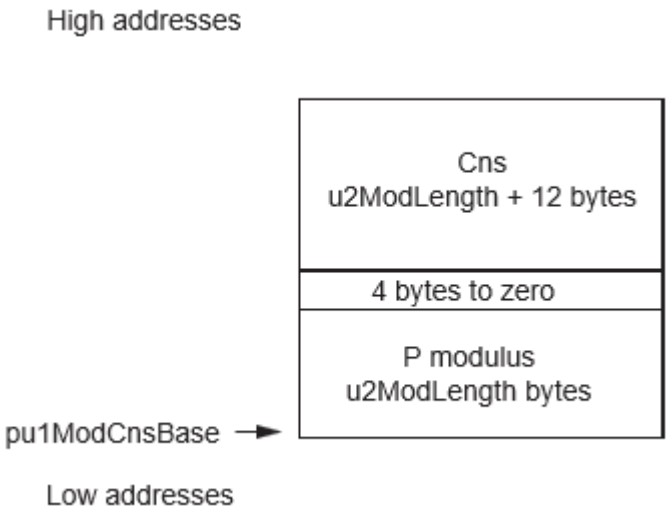


Figure 37-8. Points A and B {pu1PointABase, [(3*(u2ModLength + 4)) * ($2^{(WA-2)}$)] Or [(3*(u2ModLength + 4)) * ($2^{(WB-2)}$)]}

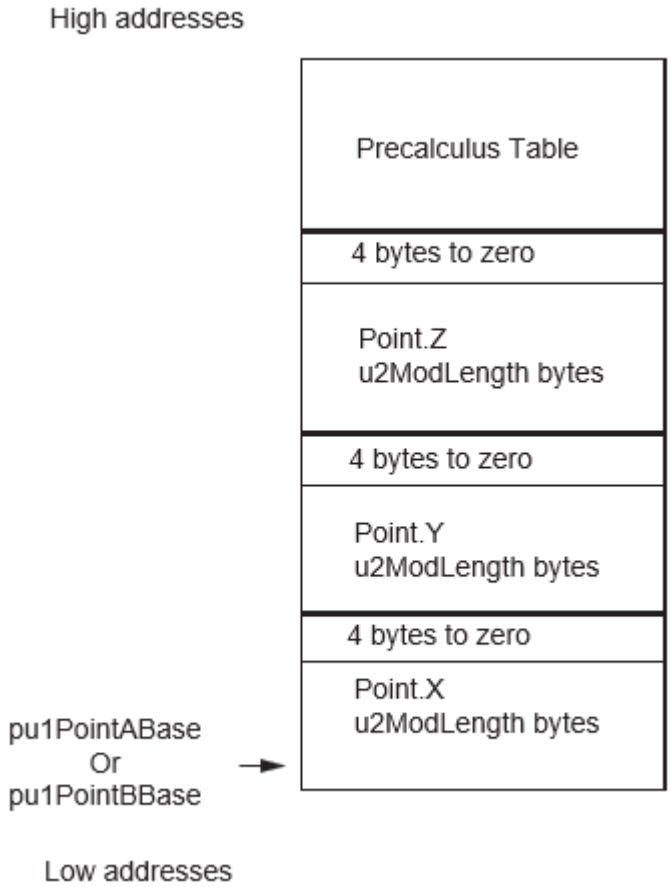


Figure 37-9. Scalars KA and KB {pu1KABBase, 2 * u2KLength + 8}

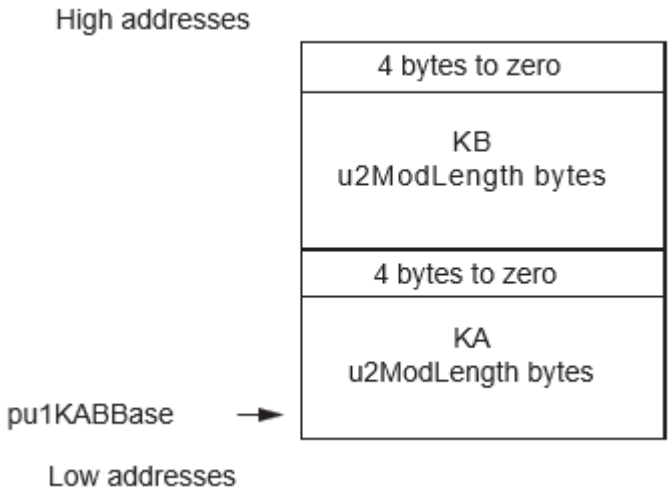
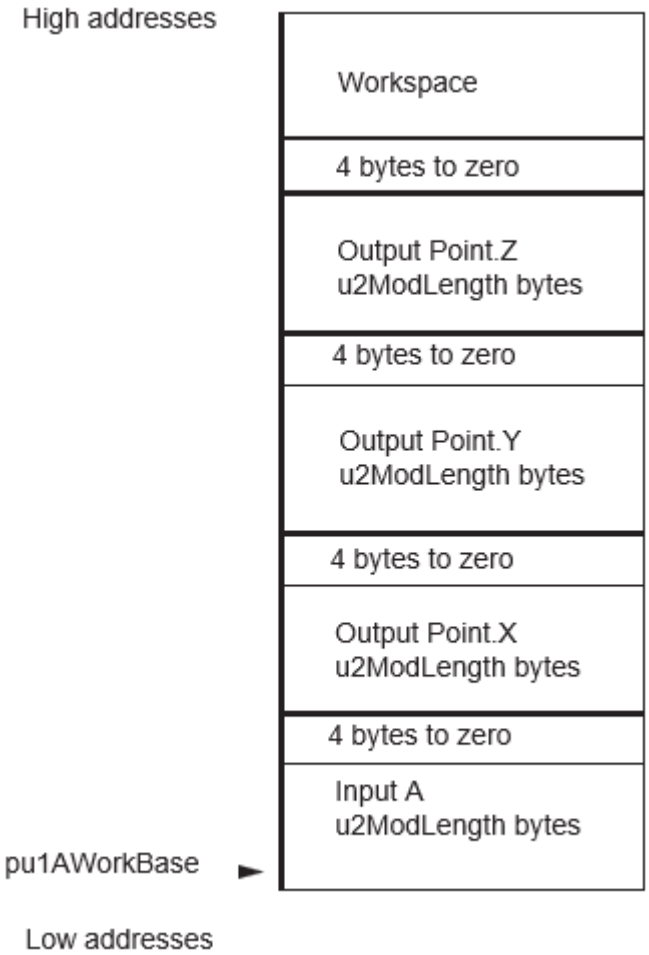


Figure 37-10. The a parameter and Workspace {pu1AWorkBase, 9*u2ModLength + 48}



37.3.6.6.9 Status Returned Values

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.6.7 Projective to Affine Coordinates Conversion

37.3.6.7.1 Purpose

This service is used to perform a point coordinates conversion from projective representation to affine.

37.3.6.7.2 How to Use the Service

37.3.6.7.3 Description

The operation performed is:

$$Pt_{X\ Affine\ \ coordinate} = \left[\frac{Pt_{X\text{Projective coordinate}}}{\left(Pt_{Z\ \text{Projective coordinate}}\right)2}\right]$$

$$Pt_{Y\ Affine\ \ coordinate} = \left[\frac{Pt_{Y\ \text{Projective coordinate}}}{\left(Pt_{Z\ \text{Projective coordinate}}\right)3}\right]$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) or affine coordinates for X and Y, and setting Z to 1(pointed by {nu1PointABase,3*u2ModLength + 12}). The Point A can be the point at infinity. In this case, the u2Status returned is PUKCL_POINT_AT_INFINITY.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4*u2ModLength +48}

The result is the point A with its (X,Y) coordinates converted to affine, and the Z coordinate set to 1. The service for this operation is ZpEcConvProjToAffine.

Important: Before using this service, ensure that the constant Cns has been calculated with the Setup of the fast Modular Reductions service.

37.3.6.7.4 Parameters Definition

Table 37-77. ZpEccConvAffineToProjective Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointABase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point A	Resulting point A in affine coordinates
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 48	–	Workspace

37.3.6.7.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
```

```
PUKCL (u2Option) = 0;

PUKCL _ZpEcConvProjToAffine(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEcConvProjToAffine(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcConvProjToAffine(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcConvProjToAffine(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEcConvProjToAffine(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcConvProjToAffine,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.7.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8},{nu1PointABase, 3*u2ModLength+ 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8},

{nu1PointABase, 3*u2ModLength + 12} and {nu1Workspace, 4*u2ModLength + 48}

37.3.6.7.7 Status Returned Values

Table 37-78. ZpEccConvAffineToProjective Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it's a representation of the infinite point.

37.3.6.8 Affine to Projective Coordinates Conversion

37.3.6.8.1 Purpose

This service is used to perform a point coordinates conversion from an affine point representation to projective.

37.3.6.8.2 How to Use the Service

37.3.6.8.3 Description

The operation performed is:

affine(*Xa*, *Ya*) → *projective*(*Xp*, *Yp*, *Zp*)

In this computation, the following parameters need to be provided:

- A the input point is filled in affine coordinates for X and Y, and setting Z to 1 (pointed by {nu1PointABase,3*u2ModLength + 4}).
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 2*u2ModLength +16}

The result is the point A with its (X,Y,Z) projective coordinates.

The service for this operation is ZpEcConvAffineToProjective

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

37.3.6.8.4 Parameters Definition

Table 37-79. ZpEccConvAffineToProjective Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointABase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point A	Resulting point A in affine coordinates
nu1Workspace	nu1	I	Crypto RAM	2*u2ModLength + 16	–	Workspace

37.3.6.8.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL (u2Option) = 0;

PUKCL _ZpEcConvAffineToProjective(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEcConvAffineToProjective(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcConvAffineToProjective(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcConvAffineToProjective(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _ZpEcConvAffineToProjective(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcConvAffineToProjective,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.8.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, and {nu1Workspace, 2*u2ModLength + 16}

37.3.6.8.7 Status Returned Values

Table 37-80. ZpEccConvAffineToProjective Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.6.9 Randomize a Coordinate

37.3.6.9.1 Purpose

This service is used to convert the projective representation of a point to another projective representation.

37.3.6.9.2 How to Use the Service

37.3.6.9.3 Description

The operation performed is:

Projective(X_1, Y_1, Z_1) → *Projective*(X_2, Y_2, Z_2)

In this computation, the following parameters need to be provided:

- The input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBase,3*u2ModLength + 12}). This Point must not be the point at infinity.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 3*u2ModLength +28}
- The random number (pointed by {nu1RandomBase, u2ModLength +4}).

The result is the point nu1PointBase with its (X,Y,Z) coordinates randomized.

The service for this operation is ZpEcRandomiseCoordinate.

Important: Before using this service:

- Ensure that the constant Cns has been calculated with the setup of the Modular Reduction service.
- Be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG

.

37.3.6.9.4 Parameters Definition

Table 37-81. ZpEccRandomiseCoordinate Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	Resulting point
nu1RandomBase	nu1	I	Crypto RAM	u2ModLength + 4	Random	Corrupted
nu1Workspace	nu1	I	Crypto RAM	3*u2ModLength + 28	–	Workspace

37.3.6.9.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _ZpEccRandomiseCoordinate(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEccRandomiseCoordinate(u2ModLength) = <Byte length of P>;
PUKCL _ZpEccRandomiseCoordinate(nu1CnsBase) = <Base of the ram location of Cns>;
```

```
PUKCL_ZpEccRandomiseCoordinate(nu1RandomBase) = <Base of the ram location where the the RNG
is stored>;
PUKCL _ZpEccRandomiseCoordinate(nu1PointBase) = <Base of the ram location of the point>;
PUKCL _ZpEccRandomiseCoordinate(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEccRandomiseCoordinate,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.9.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1RandomBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4} and {nu1Workspace, 3*u2ModLength + 28}

37.3.6.9.7 Status Returned Values

Table 37-82. ZpEccRandomiseCoordinate Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.6.10 Point is on Elliptic Curve

37.3.6.10.1 Purpose

This service is used to test whether or not the point is on the curve.

37.3.6.10.2 How to Use the Service

37.3.6.10.3 Description

The operation performed is:

Status = *IsPointOnCurve*(*X*, *Y*, *Z*)

In this computation, the following parameters need to be provided:

- The input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBase,3*u2ModLength + 4}). This Point can be the point at infinity.
- AParam and BParam are the Elliptic Curve Equation parameters. (pointed by{nu1AParam, u2ModLength+4} and {nu1BParam, u2ModLength+4}).
- Cns the Fast Modular Constant filled (pointed by{nu1CnsBase,u2ModLength+8}).
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4}).
- The workspace not initialized (pointed by {nu1WorkSpace, 4*u2ModLength +28}.

The result is the status of the point (X,Y,Z) regarding the Elliptic Curve Equation.

The service name for this operation is `ZpEcPointIsOnCurve`.

Note: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reduction service.

37.3.6.10.4 Parameters Definition

Table 37-83. ZpEcPointIsOnCurve Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	unchanged
nu1AParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter a	The parameter a
nu1BParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter b	The parameter b
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 28	–	Workspace

37.3.6.10.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL (u2Option) = 0;

PUKCL _ZpEcPointIsOnCurve(nu1ModBase) = <Base of the ram location of P>;
PUKCL _ZpEcPointIsOnCurve(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcPointIsOnCurve(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcPointIsOnCurve(nu1AParam) = <Base of the ram location of the parameter a>;
PUKCL _ZpEcPointIsOnCurve(nu1BParam) = <Base of the ram location of the parameter b>;
PUKCL _ZpEcPointIsOnCurve(nu1PointBase) = <Base of the ram location of the point>;
PUKCL _ZpEcPointIsOnCurve(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcPointIsOnCurve,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.10.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1AParam, nu1BParam, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength+4}, {nu1CnsBase, u2ModLength+8}, {nu1PointABase, 3*u2ModLength +12}, {nu1AParam, u2ModLength + 4}, {nu1BParam, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM.
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length.
- All overlapping between {nu1ModBase, u2ModLength+4}, {nu1CnsBase,u2ModLength+8}, {nu1PointABase, 3*u2ModLength+12}, {nu1AParam, u2ModLength+4}, {nu1AParam, u2ModLength + 4} and {nu1Workspace, 4*u2ModLength+28}.

37.3.6.10.7 Status Returned Values

Table 37-84. ZpEcPointIsOnCurve Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The point is on the curve.

.....continued		
Returned Status	**Importance**	**Meaning**
PUKCL_POINT_IS_NOT_ON_ CURVE	Warning	The point is not on the curve.
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it's a representation of the infinite point.

37.3.6.11 Generating an ECDSA Signature (Compliant with FIPS 186-2)

37.3.6.11.1 Purpose

This service is used to generate an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Generation. A hash value *(HashVal)* must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

A scalar number must be provided too as described in the FIPS 186-2. The result (R,S) is computed by this service.

37.3.6.11.2 How to Use the Service

37.3.6.11.3 Description

The operation performed is:

(*R*, *S*) = *EcDsaSign*(Pt_A, *HashVal*, *k*, *CurveParameters*, *PrivateKey*)

This service processes the following checks:

- If the Scalar Number k is out of the range [1, PointOrder -1], the calculus is stopped and the status is set to PUKCL_WRONG_SELECT_NUMBER.
- If R equals zero, the calculus is stopped and the status is set to PUKCL_WRONG_SELECT_NUMBER.
- If S equals zero, the calculus is stopped and the status is set to PUKCL_WRONG_SELECT_NUMBER.

In this computation, the following parameters need to be provided:

- A the input point is filled in "mixed" coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointABase,3*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase,u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8*u2ModLength + 44}
- The a parameter relative to the elliptic curve (pointed by {nu1ABase, u2ModLength + 4})
- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase, u2ScalarLength + 4})
- k the input Scalar Number beforehand generated and filled (pointed by{nu1ScalarNumber,u2ScalarLength + 4})
- HashVal the hash value beforehand generated and filled (pointed by {nu1HashBase, u2ScalarLength + 4})
- The Private Key (pointed by {nu1PrivateKey, u2ScalarLength +4})
- Generally, u2ScalarLength is equal to (u2ModLength) or (u2ModLength + 4)

Important:
For the ECDSA signature generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

The scalar number k must be selected at random. This random must be generated before the call of the ECDSA signature. For this random generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

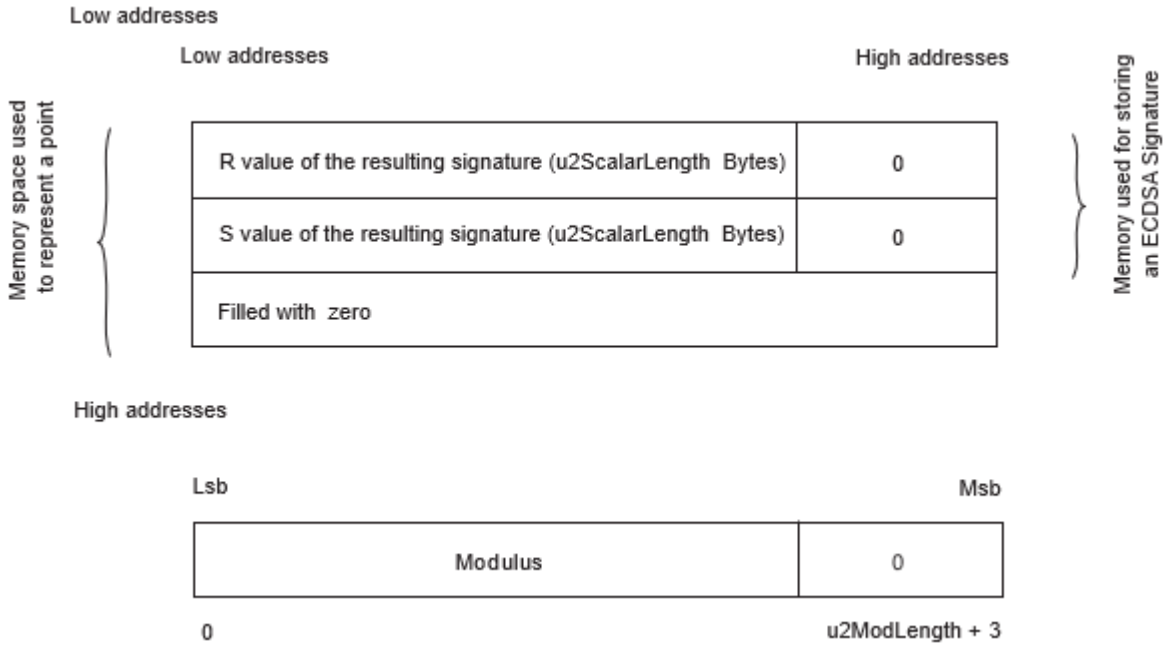
The operation performed is:

- Compute the ECDSA (R,S) as described in FIPS 186-2, but leaving the user the role of computing the input Hash Value, thus leaving the freedom of using any other algorithm than SHA-1.
- Compute a R value using the input A point and the scalar number.

- Compute a S value using R, the scalar number, the private key and the provided hash value. Note that the resulting signature (R,S) is stored at the place of the input A point.
- If all is correct and S is different from zero, the status is set to PUKCL_OK. If all is correct and S equals zero,the status is set to PUKCL_WRONG_SELECT_NUMBER. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

The service name for this operation is `ZpEcDsaGenerateFast`. This service uses Fast mode and Fast Modular Reduction for computation.

- The signature (R,S), when resulting from a computation is given back at address of the A point:
  - R output is at offset 0 and has length (u2ScalarLength + 4)bytes.
  - S output is at offset (u2ScalarLength + 4) bytes and has length (u2ScalarLength + 4) bytes.
  - The MSB 4 zero bytes may be suppressed to get the R and S values on u2ScalarLength bytes



37.3.6.11.4 Parameters Definition

Table 37-85. ZpEcDsaGenerateFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1ScalarNumber	nu1	I	Crypto RAM	u2ScalarLength + 4	Scalar Number used to multiply the point A	Unchanged
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PrivateKey	nu1	I/O	Crypto RAM	u2ScalarLength + 4	Base of the Private Key	Unchanged

.....continued						
Parameter	**Type**	**Direction**	**Location**	**Data Length**	**Before Executing the Service**	**After Executing the Service**
nu1HashBase(1)	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Unchanged
u2ScalarLength	u2	I	–	–	Length of scalar (same length as the length of order)	Length of scalar
nu1PointABase(2)	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (three coordinates (X,Y) affine and Z = 1)	Resulting signature (R,S,0)
nu1ABase	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	–	Corrupted workspace

Notes:

1. The hash value calculus is defined by the ECDSA norm and depends on the elliptic curve domain parameters. To construct the input parameter, the 4 Most Significant Bytes must be set to zero.
2. The resulting signature format is different from the point A format (see Description above for information on the point A format).

37.3.6.11.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip


PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _ZpEcDsaGenerate(nu1ModBase) = <Base of the ram location of P>; PUKCL
_ZpEcDsaGenerate(u2ModLength) = <Byte length of P>;
PUKCL _ZpEcDsaGenerate(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _ZpEcDsaGenerate(nu1PointABase) = <Base of the A point>;
PUKCL _ZpEcDsaGenerate(nu1PrivateKey) = <Base of the Private Key>;
PUKCL _ZpEcDsaGenerate(nu1ScalarNumber) = <Base of the ScalarNumber>;
PUKCL _ZpEcDsaGenerate(nu1OrderPointBase) = <Base of the order of A point>;
PUKCL _ZpEcDsaGenerate(nu1ABase) = <Base of the a parameter of the curve>;
PUKCL _ZpEcDsaGenerate(nu1Workspace) = <Base of the workspace>;
PUKCL _ZpEcDsaGenerate(nu1HashBase) = <Base of the SHA resulting hash>;
PUKCL_ZpEcDsaGenerate(u2ScalarLength)    = < Length of ScalarNumber>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcDsaGenerateFast, pvPUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.6.11.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PrivateKey, nu1ScalarNumber, nu1OrderPointBase,nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12},{nu1PrivateKey, u2ScalarLength + 4},{nu1ScalarNumber, u2ScalarLength + 4},{nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PrivateKey, u2ScalarLength + 4}, {nu1ScalarNumber, u2ScalarLength + 4}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

37.3.6.11.7 Status Returned Values

Table 37-86. ZpEcDsaGenerateFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem. The signature is the good one.
PUKCL_WRONG_SELECTNUMBER	Warning	The given value for nu1ScalarNumber is not good to perform this signature generation.

37.3.6.12 Verifying an ECDSA Signature (Compliant with FIPS186-2)

37.3.6.12.1 Purpose

This service is used to verify an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Verification.

A hash value (HashVal) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

As second significant input, the Signature is provided to be checked. This service checks the signature and fills the status accordingly.

37.3.6.12.2 How to Use the Service

37.3.6.12.3 Description

The operation performed is:

Verify = *EcDsaVerifySignature*(Pt_A, *HashVal*, *Signature*, *CurveParameters*, *PublicKey*)

The points used for this operation are represented in different coordinate systems. In this computation, the following parameters need to be provided:

- A the input point is filled with the affine values (X,Y) and Z = 1 (pointed by{nu1PointABase,3*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase,u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8*u2ModLength + 44}
- The a parameter relative to the elliptic curve (pointed by {nu1ABase,u2ModLength + 4})
- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase,u2ScalarLength + 4})
- HashVal the hash value is generated prior and filled (pointed by {nu1HashBase,u2ScalarLength + 4})
- The Public Key point is filled in “mixed” coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointPublicKeyGen, 3*u2ModLength + 12})
- The input signature (R,S), even if it is not a Point, is represented in memory like a point in affine coordinates (X,Y) (pointed by {nu1PointSignature, 2*u2ScalarLength + 8})
  Note: For the ECDSA signature verification be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

- The operation consists in obtaining a V value with all these input parameters and checking that V equals the provided R. If all is correct and the signature is the good one, the status is set to PUKCL_OK. If all is correct and the signature is wrong, the status is set to PUKCL_WRONG_SIGNATURE. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

37.3.6.12.4 Parameters Definition

Table 37-87. ZpEcDsaVerifyFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PointSignature	nu1	I	Crypto RAM	2*u2ScalarLength + 8	Signature(r, s)	Corrupted
nu1HashBase(1)	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Corrupted
u2ScalarLength	u2	I	–	–	Length of scalar	Length of scalar
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Generator point	Corrupted
nu1PointPublicKeyGen	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Public point	Corrupted
nu1ABase	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	–	Corrupted workspace

Note:

1. The hash value calculus is defined by the ECDSA norm and depends on the elliptic curve domain parameters. To construct the input parameter, the 4 Most Significant Bytes must be set to zero.

37.3.6.12.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL(u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL_ZpEcDsaVerify(nu1ModBase) = <Base of the ram location of P>;
PUKCL_ZpEcDsaVerify(u2ModLength) = <Byte length of P>;
PUKCL_ZpEcDsaVerify(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_ZpEcDsaVerify(nu1PointABase) = <Base of the A point>;
PUKCL_ZpEcDsaVerify(nu1PrivateKey) = <Base of the Private Key>;
PUKCL_ZpEcDsaVerify(nu1ScalarNumber) = <Base of the ScalarNumber>;
PUKCL_ZpEcDsaVerify(nu1OrderPointBase) = <Base of the order of A point>;
PUKCL_ZpEcDsaVerify(nu1ABase) = <Base of the a parameter of the curve>;
PUKCL_ZpEcDsaVerify(nu1Workspace) = <Base of the workspace>;
```

```
PUKCL_ZpEcDsaVerify(nu1HashBase) = <Base of the SHA resulting hash>;
 PUKCL_ZpEcDsaVerify(u2ScalarLength)    = < Length of ScalarNumber>;

...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcDsaVerifyFast, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
            {
            ...
            }ou
else
            if(PUKCL(u2Status) == PUKCL_WRONG_SIGNATURE)
                        {
                        ...
                        }
            else // Manage the error
```

37.3.6.12.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointPublicKeyGen, nu1PointSignature, nu1OrderPointBase,nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12}, {nu1PointPublicKeyGen, 3*u2ModLength + 12}, {nu1PointSignature,2*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PointPublicKeyGen, 3*u2ModLength + 12}, {nu1PointSignature, 2*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

37.3.6.12.7 Status Returned Values

Table 37-88. ZpEcDsaVerifyFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem. The signature is the good one.
PUKCL_WRONG_SIGNATURE	Warning	The signature is wrong.

37.3.6.13 Quick Verifying an ECDSA Signature (Compliant with FIPS 186-2)

37.3.6.13.1 Purpose

This service is used to verify an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Verification using Quick Dual Multiplying to perform computation.

A hash value (HashVal) must be provided as input, it has to be previously computed from the message whose signature is verified using a secure hash algorithm.

As second significant input, the Signature is provided to be checked.

This service checks the signature and fills the status accordingly.

Important: This service has a quick implementation without additional security.

37.3.6.13.2 How to Use the Service

37.3.6.13.3 Description

The operation performed is:

Verify = *EcDsaVerifySignature*(Pt_A, *HashVal*, *Signature*, *CurveParameters*, *PublicKey*)

The points used for this operation are represented in different coordinate systems.

In this computation, the following parameters need to be provided (such that u2MaxLength = max(u2ModLength, u2ScalarLength)):

- A the input point is filled with the affine values (X,Y) and Z = 1 (pointed by {pu1PointABase,(3*(u2ModLength + 4)) * $(2^{(WA-2)})$})
- P the modulus filled and Cns the working space for the Fast Modular Constant not initialized (pointed by {pu1ModBase, u2ModLength + u2MaxLength + 16})
- The a parameter relative to the elliptic curve filled and workspace not initialized (pointed by {pu1AWorkBase,8*u2MaxLength + u2ModLength + 48})
- The order of the Point A on the elliptic curve (pointed by {pu1OrderPointBase,u2ScalarLength +4})
- HashVal the hash value beforehand generated and filled (pointed by {pu1HashBase,u2MaxLength +4})
- The Public Key point is filled in “mixed” coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointPublicKeyGen, (3*(u2ModLength + 4)) * $(2^{(WB-2)})$})
- The input signature (R,S), even if it is not a Point, is represented in memory like a point in affine coordinates (X,Y) (pointed by {nu1PointSignature, 2*u2ScalarLength + 8})

The operation consists of obtaining a V value with all input parameters and checks that V equals the provided R. If all is correct and the signature is the good one, the status is set to PUKCL_OK. If all is correct and the signature is wrong, the status is set to PUKCL_WRONG_SIGNATURE. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

37.3.6.13.4 Parameters Definition

To place the parameters correctly the maximum of u2ModLength and u2ScalarLength must be calculated: u2MaxLength = max(u2ModLength, u2ScalarLength)

WA is the Point A window size and WB is the Point Public Key window size (see Options below for details).

Important: Please calculate precisely the length of areas with the formulas and the `max()` service which takes the maximum of two values. Ensure that the pu1 type is a pointer on 4 bytes and contains the full address (see 37.3.3.4. Aligned Significant Length for details).

Table 37-89. ZpEcDsaQuickVerify Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
pu1ModCnsBase	pu1	I	Crypto RAM	u2ModLength + 4 + u2MaxLength + 12	Base of modulus P	Base of modulus P
u2Option	u2	I	–	–	Option related to the called service (see below)	–
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
pu1OrderPointBase	pu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
pu1PointSignature	pu1	I	Any RAM	2*u2ScalarLength + 8	Signature(r, s)	Corrupted

.....continued

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
pu1HashBase (see **Note 1**)	pu1	I	Crypto RAM	u2MaxLength + 4	Base of the hash value resulting from the previous SHA	Corrupted
u2ScalarLength	u2	I	–	–	Length of scalar	Length of scalar
pu1PointABase	pu1	I/O	Crypto RAM	(3*u2ModLength + 12) * (2(WA-2))	Generator point	Corrupted
pu1PointPublicKeyGen	pu1	I/O	Crypto RAM	(3*u2ModLength + 12) * (2(WB-2))	Public Key point	Corrupted
pu1AWorkBase	pu1	I	Crypto RAM	(u2ModLength + 4) + (8*u2MaxLength + 44)	Parameter a of the elliptic curve and Workspace	Corrupted

Note:

1. 1. The hash value calculus is defined by the ECDSA norm and depends on the elliptic curve domain parameters. To construct the input parameter, the 4 Most Significant Bytes must be set to zero.

A suggested parameters placement in Crypto RAM is:

- ModCnsBase
- OrderPointBase
- Signature may be placed here or in Classical RAM
- HashBase
- PointABase
- PointPublicKeyGen
- AWorkBase

37.3.6.13.5 Options

The options are set by the u2Options input parameter, which is composed of:

- the mandatory windows sizes WA (window for Point A) and WB (window for Point Public Key)
- the indication of the presence of the Point Signature in system RAM

Important: Please check precisely if the Point Signature is in Crypto RAM. If this is the case the PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM must not be used.

The u2Options number is calculated by an “Inclusive OR” of the options. Some Examples in C language are:

- ```
  // Point Signature in system RAM
  // The Point A window size is 3

  // The Point Public Key window size is 4

  PUKCL(u2Options) = PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM |

  PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(3) |

  PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(4);
  ```
- ```
  // Point Signature in the Cryptographic RAM
  // The Point A window size is 2
  ```

```
// The Point Public Key window size is 5
PUKCL(u2Options) = PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(2) |
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(5);
```

For this service, many window sizes are possible. The window sizes in bits are those of the windowing method used for the scalar multiplying.

The choice of the window sizes is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation table size.
- Increasing the window size to the optimum reduces the computation time.

The following table details the estimated windows WA and WB optimum and possible for some curves.

Table 37-90. ZpEcDsaQuickVerify Service Estimated WA and WB Window Size

Curve Size (bits)	Optimum Window size	Possible Window Sizes (WA, WB) or (WB, WA)
192	5	5, 5
256	5	5, 5
384	6	5, 5
521	6	4, 5

The following table details the size of the point and the precomputation table, depending on the chosen window size option.

Table 37-91. ZpEcDsaQuickVerify Service Window Size and Precomputation Table Size Options

Option Specified	Point and Precomputation Table Size
PUKCL_ZPECCMUL_WINSIZE_A_VAL_TO_OPT(WA) WA in [2, 15]	(3*(u2ModLength + 4)) * ($2^{(WA-2)}$)
PUKCL_ZPECCMUL_WINSIZE_B_VAL_TO_OPT(WB) WB in [2, 15]	(3*(u2ModLength + 4)) * ($2^{(WB-2)}$)

The Point Signature can be located in PUKCC RAM or in system RAM. If the Point Signature is entirely in system RAM with no part in PUKCC RAM this can be signaled by us ing the option PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM. In all other cases this option must not be used.

The following table describes this option.

Table 37-92. ZpEcDsaQuickVerify Service Point Signature in Classical RAM Option

Option	Purpose
PUKCL_ZPECCMUL_SCAL_IN_CLASSIC_RAM	The Point Signature can be located in Crypto RAM or in system RAM. If the Point Signature is entirely in system RAM with no part in PUKCC RAM this can be signaled by using this option. In all other cases this option must not be used.

37.3.6.13.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
PUKCL(u2Option) = <Point Signature location and windows sizes>;
PUKCL_ZpEcDsaQuickVerify(pu1ModCnsBase) = <Base of the ram location of P and Cns>;
PUKCL_ZpEcDsaQuickVerify(u2ModLength) = <Byte length of P>;
PUKCL_ZpEcDsaQuickVerify(pu1PointABase) = <Base of the ram location of the A point>;
PUKCL_ZpEcDsaQuickVerify(pu1PointPublicKeyGen) = <Base of the Public Key>;
PUKCL_ZpEcDsaQuickVerify(pu1PointSignature) = <Base of the Signature (r, s)>;
PUKCL_ZpEcDsaQuickVerify(pu1OrderPointBase) = <Base of the order of the A point>;
PUKCL_ZpEcDsaQuickVerify(pu1AWorkBase) = <Base of the ram location of the parameter A of the
elliptic curve and workspace>;
PUKCL_ZpEcDsaQuickVerify(pu1HashBase) = <Base of the SHA resulting hash>;
PUKCL_ZpEcDsaQuickVerify(u2ScalarLength) = <Byte length of R and S in Point Signature>;
. . .
```

```
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(ZpEcDsaQuickVerify, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
                {
                ...
                }
else
                if ( PUKCL(u2Status) = PUKCL_WRONG_SIGNATURE )
                                {
                                ...
                                }
                else // Manage the error
```

37.3.6.13.7 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- pu1ModCnsBase, pu1PointABase, pu1PointPublicKeyGen, pu1PointSignature,pu1OrderPointBase, pu1AWorkBase or pu1HashBase are not aligned on 32-bit boundaries
- {pu1ModCnsBase, u2ModLength + 4 + u2MaxLength + 12}, {pu1PointABase, (3 * u2ModLength + 12)* ($2^{(WA-2)}$)}, {pu1PointPublicKeyGen, (3 * u2ModLength + 12) * ($2^{(WPub-2)}$)}, {pu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {pu1AWorkBase, (u2ModLength + 4) + (8 * u2MaxLength + 44)} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {pu1ModCnsBase, u2ModLength + 4 + u2MaxLength + 12},{pu1PointABase, (3 * u2ModLength + 12) * ($2^{(WA-2)}$)}, {pu1PointPublicKeyGen, (3 * u2ModLength + 12) *($2^{(WPub-2)}$)}, {pu1OrderPointBase, u2ScalarLength + 4}, {pu1PointSignature, 2 * u2ScalarLength + 8}, {nu1ABase, u2ModLength + 4}, {pu1AWorkBase, (u2ModLength + 4) + (8 * u2MaxLength + 44)} and {nu1HashBase, u2ScalarLength + 4}

37.3.6.13.8 Status Returned Values

Table 37-93. ZpEcDsaQuickVerify Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem. The signature is the good one.
PUKCL_WRONG_SIGNATURE	Warning	The signature is wrong.

37.3.6.13.9 Parameter Placement

The parameters' placement is described in detail in the following figures.

Figure 37-11. Modulus P and Cns{pu1ModCnsBase, u2ModLength + 4 + u2MaxLength + 12}

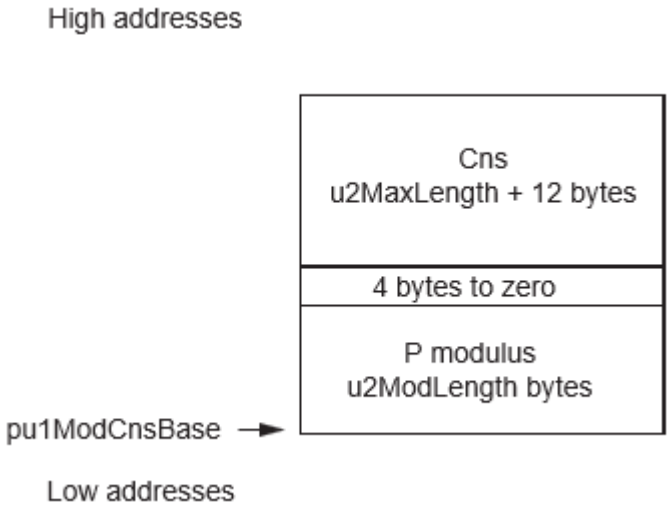


Figure 37-12. Points A {pu1PointABase, (3*(u2ModLength + 4)) * (2$^{(WA-2)}$)} and Public Key Gen {pu1PointPublicKeyGen, (3*(u2ModLength + 4)) * (2$^{(WB-2)}$)}

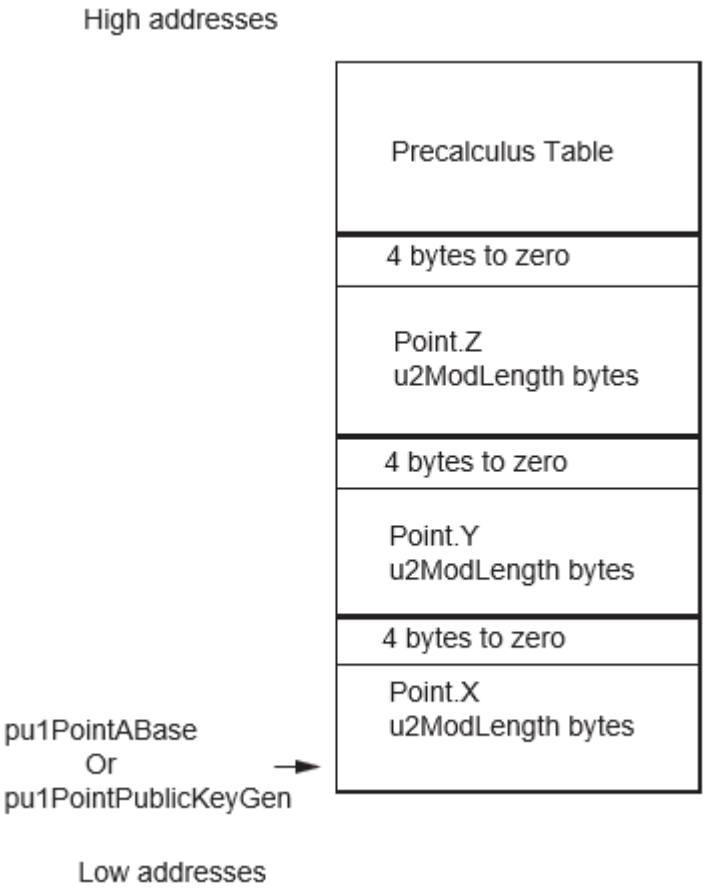


Figure 37-13. PointSignature {pu1PointSignature, 2 * u2ScalarLength + 8}

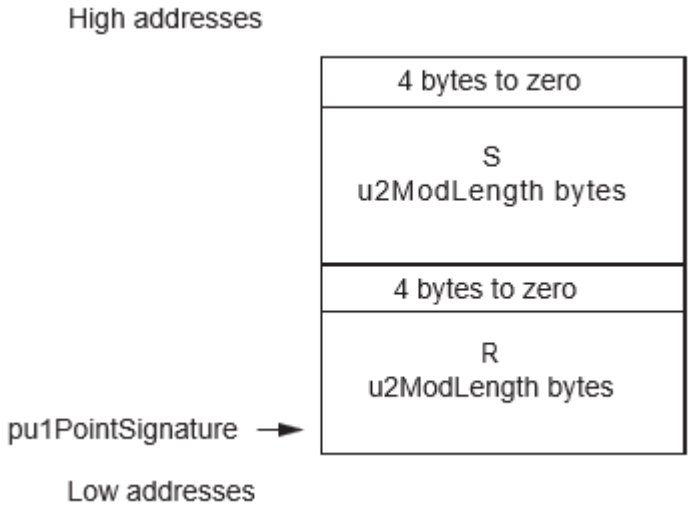
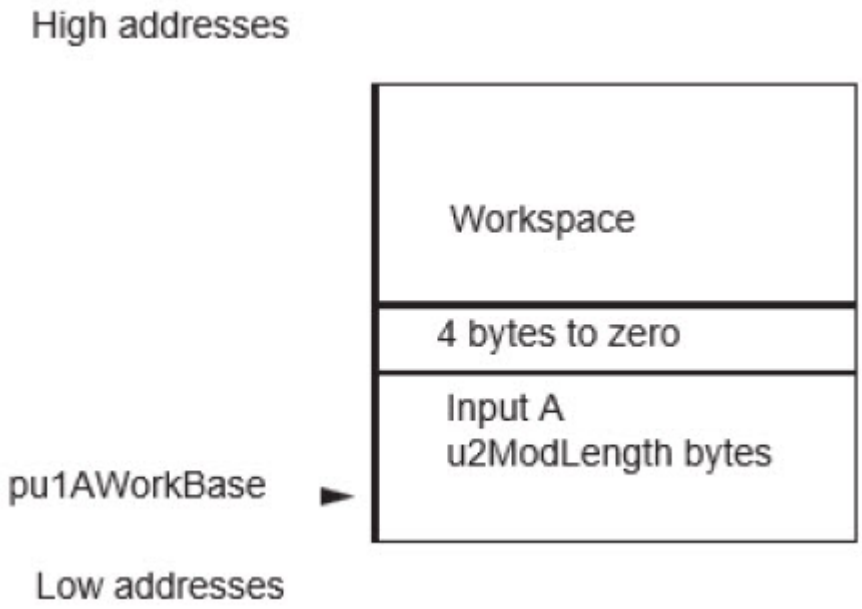


Figure 37-14. The a parameter and Workspace {pu1AWorkBase, 9*u2ModLength + 48}



37.3.7 Elliptic Curves Over GF(2^n) Services

This section provides a complete description of the currently available elliptic curve over Polynomials in GF(2^n) services.

These services process Polynomials in GF(2^n) only.

The offered services cover the basic operations over elliptic curves such as:

- Adding two points over a curve
- Doubling a point over a curve
- Multiplying a point by an integral constant
- Converting a point's projective coordinates (resulting from a doubling or an addition) to the affine coordinates, and oppositely converting a point's affine coordinates to the projective coordinates.
- Testing the point presence on the curve.

Additionally, some higher level services covering the needs for signature generation and verification are offered:

- Generating an ECDSA signature (compliant with FIPS186-2)
- Verifying an ECDSA signature (compliant with FIPS 186-2) The supported curves use the following curve equation in GF(2^n):

$Y^2 + XY = X^3 + aX + b$

37.3.7.1 Parameters Format

Related Links

37.3.5.1. Modular Reduction

37.3.3.4. Aligned Significant Length

37.3.7.1.1 Polynomials in GF(2^n)

Polynomials in GF(2^n) are binary polynomials reduced modulo the polynomial P[X]. This polynomial is called the modulus and may be abbreviated to P in this document. The storage of these polynomials in memory area is described in Aligned Significant Length (see *Aligned Significant Length* from Related Links).

For notation simplicity the comparison signs “<“ or “>” may be used for polynomials, this is to be interpreted as a comparison between the degree of the polynomials.

In GF(2^n) fully reduced polynomials are of degree strictly lower than degree(P[X]). In many cases the polynomials used in this library are only partially reduced and so have a degree higher or equal than degree(P[X]), but this degree is maintained strictly lower than (degree(P[X]) + 15).

37.3.7.1.2 Coordinates System

In this implementation, several choices have been made related to the coordinate systems managed by the elliptic curve primitives.

There are two systems currently managed by the library:

- Affine Coordinates System where each curve point has two coordinates (X,Y)
- Projective Coordinates System where each point is represented with three coordinates (X,Y,Z)

Converting from the affine coordinates system to a projective coordinates system and is performed by extending its representation having Z = 1:

$(X,Y) \Rightarrow (X,Y, Z= 1)$

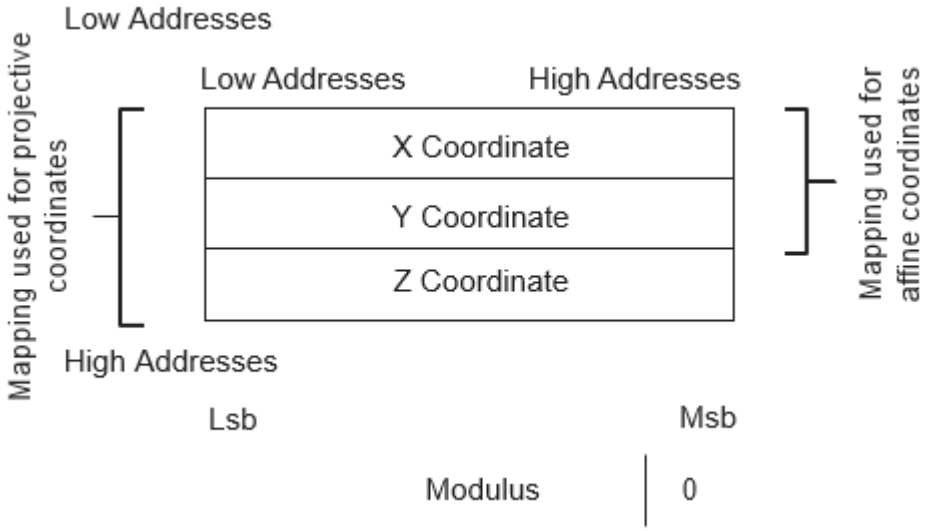
Converting from a projective coordinate to an affine one is a service offered by the library. The formula to perform this conversion is:

$(X,Y, Z) \Rightarrow (X \Rightarrow Z, Y/Z^2)$

37.3.7.1.3 Points Representation in Memory

Depending on the representation (Projective or Affine), points are represented in memory as shown in the following figure.

Figure 37-15. Point Representation in Memory



In this figure, the modulus is represented as a reference, and to show that coordinates are always to be provided on the length of the modulus plus one 32-bit word.

Different types of representations are listed here:

Affine representation: $Pt = \begin{bmatrix} X_{Affine} < P \times X^{15} \\ Y_{Affine} < P \times X^{15} \end{bmatrix}$

Projective representation: $Pt = \begin{bmatrix} X_{\mathrm{Projective}} < P \times X^{15} \\ Y_{\mathrm{Projective}} < P \times X^{15} \\ Z_{\mathrm{Projective}} < P \times X^{15} \end{bmatrix}$

Notes:

1. The minimum value for u2ModLength is 12 bytes. Therefore, the significant length of the modulus must be at least three 32-bit words.
2. In some cases the point can be the infinite point. In this case it is represented with its Z coordinates equal or congruent to zero.

37.3.7.1.4 Modulus and Modular Constant Parameters

In most of the services the following parameters must be provided:

- P the Modulus (often pointed by {nu1ModBase,u2ModLength + 4}): This parameter contains the Modulus Polynomial P[X] defining the Galois Field used in points coordinates computations. The Modulus must be u2ModLength bytes long, while having a supplemental zeroed 32-bit word on the MSB side.
  Note: Most of the Elliptic Curve computations are reduced modulo P. In many functions the reductions are made with the Fast Reduction.
- Cns the Modular Constant (often pointed by {nu1CnsBase,u2ModLength + 12}): This parameter contains the Modular Constant associated to the Modulus.

Important: The Modular Constant must be calculated before using the GF(2^n) Elliptic Curves functions by a call to the Setup for Modular Reductions with the GF(2^n) option (see *Modular Reduction* from Related Links).

37.3.7.1.5 Curve Parameters in Memory

Some services need one or both of the Elliptic Curve Equation Parameters a and b. In this case these values are organized in memory as follows:

- The a Parameter relative to the Elliptic Curve Equation (often pointed by {nu1ABase,u2ModLength +4}). The a Parameter is written in a classical way in memory. It is u2ModLength bytes long and has a supplemental zeroed 32-bit word on the MSB side.
- The a and b Parameters relative to the Elliptic Curve Equation (often pointed by {nu1ABBase,2*u2ModLength + 8}):
  - The a Parameter is written in memory on u2ModLength bytes long, with a supplemental zeroed 32-bit word on the MSB side.
  - The b Parameter is written in memory after the a Parameter at an offset of (u2ModLength + 4) bytes. It is written in memory on u2ModLength bytes long, with a supplemental zeroed 32-bit word on the MSB side.

37.3.7.2 Point Addition

37.3.7.2.1 Purpose

This service is used to perform a point addition, based on a given elliptic curve over GF(2^n).

Please note that this service is not intended to add the same point twice. In this particular case, use the doubling service (see 37.3.7.3. Point Doubling).

37.3.7.2.2 How to Use the Service

37.3.7.2.3 Description

The operation performed is:

$Pt_C = Pt_A + Pt_B$

In this computation, the following parameters need to be provided:

- Point A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Point B the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBBase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 12})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The a parameter relative to the elliptic curve equation (pointed by {nu1ABase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 7*u2ModLength + 40}

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the same place than the input point A. This Point can be the Infinite Point.

The services for this operation are:

- Service GF2NEccAddFast: The fast mode is used, the fast modular reduction is used in the computations.

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Modular Reductions service.

37.3.7.2.4 Parameters Definition

Table 37-94. GF2NEccAddFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of Modulus P	Base of Modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulo	Length of modulo

.....continued						
Parameter	**Type**	**Direction**	**Location**	**Data Length**	**Before Executing the Service**	**After Executing the Service**
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1PointBBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point B (projective coordinates)	Input point B
nu1ABBase	nu1	I	Crypto RAM	u2ModLength + 4	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	7*u2ModLength + 40	–	Corrupted workspace

37.3.7.2.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;
//Depending on the function the Random Number Generator
//must be initialized and started
//following the directives given for the RNG on the chip
PUKCL(u2Option) = 0;
PUKCL_GF2NEccAdd(nu1ModBase) = <Base of the ram location of P>;
PUKCL_GF2NEccAdd(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_GF2NEccAdd(u2ModLength) = <Byte length of P>;
PUKCL_GF2NEccAdd(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL_GF2NEccAdd(nu1PointBBase) = <Base of the ram location of the B point>;
PUKCL_GF2NEccAdd(nu1ABBase) = <Base of the ram location of the a Parameter>;
PUKCL_GF2NEccAdd(nu1Workspace) = <Base of the ram location of the workspace>;
. . .
// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEccAddFast, pvPUKCLParam);
if (PUKCL(u2Status) == PUKCL_OK)
               {
               ...
               }
else // Manage the error
```

37.3.7.2.6 Constraints

No overlapping between either input and output are allowed The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase,nu1CnsBase, nu1PointABase, nu1PointBBase, nu1ABBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12}, {nu1PointBBase, 3*u2ModLength + 12}, {nu1ABase,u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PointBBase, 3*u2ModLength + 12}, {nu1ABase,u2ModLength + 4} and {nu1Workspace, 5*u2ModLength + 32}

37.3.7.2.7 Status Returned Values

Table 37-95. GF2NEccAddFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without errors.

37.3.7.3 Point Doubling

37.3.7.3.1 Purpose

This service is used to perform a Point Doubling, based on a given elliptic curve over GF(2^n).

37.3.7.3.2 How to Use the Service

37.3.7.3.3 Description

The operation performed is:

$Pt_C = 2 \times Pt_A$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength +8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4*u2ModLength +28}
- The a and b Parameters relative to the Elliptic Curve Equation (pointed by {nu1ABBase,2*u2ModLength+ 8})
- The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This point can be the Infinite Point.

The service name for this operation is `GF2NEccDblFast`. This service uses Fast mode and Fast Modular Reduction for computation.

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

37.3.7.3.4 Parameters Definition

Table 37-96. GF2NEccDblFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1ABBase	u2	I	Crypto RAM	2*u2ModLength + 8	Parameters a and b of the elliptic curve	Parameter a and b of the elliptic curve
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 28	–	Corrupted workspace

37.3.7.3.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _GF2NEccDbl(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEccDbl(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEccDbl(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEccDbl(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _GF2NEccDbl(nu1ABBase) = <Base of the a and b parameters of the elliptic curve>;
```

```
PUKCL _GF2NEccDbl(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEccDblFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.7.3.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1ABBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12}, {nu1ABBase, 2*u2ModLength + 8}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1ABase, u2ModLength + 4} and {nu1Workspace, 4*u2ModLength + 28}

37.3.7.3.7 Status Returned Values

Table 37-97. GF2NEccDblFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.7.4 Scalar Point Multiply

37.3.7.4.1 Purpose

This service is used to multiply a point by an integral constant K on a given elliptic curve over GF(2^n).

37.3.7.4.2 How to Use the Service

37.3.7.4.3 Description

The operation performed is:

$Pt_C = K \times Pt_A$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointABase,3*u2ModLength + 12}). This point can be the Infinite Point.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8*u2ModLength + 44}
- The a and b parameters relative to the elliptic curve (pointed by {nu1ABBase,2*u2ModLength + 8})
- K the scalar number (pointed by {nu1ScalarNumber,u2ScalarLength + 4})

The resulting C point is represented in projective coordinates (X,Y,Z) and is stored at the very same place than the input point A. This point can be the Infinite Point.

The service name for this operation is `GF2NEccMulFast`. This service uses Fast mode and Fast Modular Reduction for computation.

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

37.3.7.4.4 Parameters Definition

Table 37-98. GF2NEccMulFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1KBase	nu1	I	Crypto RAM	u2KLength	Scalar number used to multiply the point A	Unchanged
u2KLength	u2	I	–	–	Length of scalar K	Length of scalar K
nu1PointBase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (projective coordinates)	Resulting point C (projective coordinates)
nu1ABase	nu1	I	Crypto RAM	2*u2ModLength + 8	Parameters a and b of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	–	Corrupted workspace

37.3.7.4.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

PUKCL (u2Option) = 0;

PUKCL _GF2NEccMul(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEccMul(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEccMul(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEccMul(nu1PointBase) = <Base of the ram location of the A point>;
PUKCL _GF2NEccMul(nu1ABase) = <Base of the ram location of the parameters a and b of the
elliptic
curve>;
PUKCL _GF2NEccMul(nu1KBase) = <Base of the ram location of the scalar number>;
PUKCL _GF2NEccMul(nu1Workspace) = <Base of the ram location of the workspace>;
PUKCL _GF2NEccMul(u2KLength) = <Length of the ram location of the scalar number>;

...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEccMulFast,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.7.4.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointBase, nu1ABase, nu1KBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointBase, 3*u2ModLength+ 12}, {nu1ABase, 2*u2ModLength + 8}, {nu1KBase, u2KLength} or {nu1Workspace, 8*u2ModLength + 44} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length

- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointBase, 3*u2ModLength + 12}, {nu1ABase, 2*u2ModLength + 8}, {nu1KBase, u2KLength} and {nu1Workspace, 8*u2ModLength + 44}

37.3.7.4.7 Status Returned Values

Table 37-99. GF2NEccMulFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.7.5 Projective to Affine Coordinates Conversion

37.3.7.5.1 Purpose

This service is used to perform a point coordinates conversion from a projective representation to an affine.

37.3.7.5.2 How to Use the Service

37.3.7.5.3 Description

The operation performed is:

$$Pt_{X\ Affine\ \ coordinate} = \left[\frac{Pt_{X\mathrm{Projective\ coordinate}}}{\left(Pt_{Z\ \mathrm{Projective\ coordinate}}\right)}\right]$$

$$Pt_{Y\ Affine\ \ coordinate} = \left[\frac{Pt_{Y\ \mathrm{Projective\ coordinate}}}{\left(Pt_{Z\ \mathrm{Projective\ coordinate}}\right)2}\right]$$

In this computation, the following parameters need to be provided:

- A the input point is filled in projective coordinates (X,Y,Z) or affine coordinates for X and Y, and setting Z to 1 (pointed by {nu1PointABase,3*u2ModLength + 12}). The Point A can be the point at infinity. In this case, the u2Status returned is PUKCL_POINT_AT_INFINITY.
- Cns the Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 4*u2ModLength + 48}

The result is the point A with its (X,Y) coordinates converted to affine, and the Z coordinate set to 1.

The service name for this operation is `GF2NEcConvProjToAffine`.

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

37.3.7.5.4 Parameters Definition

Table 37-100. GF2NEcConvProjToAffine Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointABase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point A	Resulting point A in affine coordinates

.....continued						
Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 48	–	Workspace

37.3.7.5.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

PUKCL _GF2NEcConvProjToAffine(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcConvProjToAffine(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcConvProjToAffine(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcConvProjToAffine(nu1PointABase) = <Base of the ram location of the A point>;
PUKCL _GF2NEcConvProjToAffine(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcConvProjToAffine,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.7.5.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8},{nu1PointABase, 3*u2ModLength + 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12} and {nu1Workspace, 4*u2ModLength + 48}

37.3.7.5.7 Status Returned Values

Table 37-101. GF2NEcConvProjToAffine Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it is a representation of the infinite point.

37.3.7.6 Affine to Projective Coordinates Conversion

37.3.7.6.1 Purpose

This service is used to perform a point coordinates conversion from an affine point representation to projective.

37.3.7.6.2 How to Use the Service

37.3.7.6.3 Description

The operation performed is:

affine(*Xa*, *Ya*) → *projective*(*Xp*, *Yp*, *Zp*)

In this computation, the following parameters need to be provided:

- A the input point is filled in affine coordinates for X and Y, and setting Z to 1 (pointed by {nu1PointABase,3*u2ModLength + 4}).
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 2*u2ModLength +16} The result is the point A with its (X,Y,Z) projective coordinates.

The service name for this operation is `GF2NEcConvAffineToProjective`.

Important: Before using this service, ensure that the constant Cns has been calculated with the setup of the Fast Modular Reductions service.

37.3.7.6.4 Parameters Definition

Table 37-102. GF2NEcConvAffineToProjective Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointABase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point A	Resulting point A in affine coordinates
nu1Workspace	nu1	I	Crypto RAM	2*u2ModLength + 16	–	Workspace

37.3.7.6.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

PUKCL _GF2NEcConvAffineToProjective(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcConvAffineToProjective(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcConvAffineToProjective(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcConvAffineToProjective(nu1PointABase) = <Base of the ram location of the A
point>;
PUKCL _GF2NEcConvAffineToProjective(nu1Workspace) = <Base of the ram location of the
workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcConvAffineToProjective,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.7.6.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1Workspace are not aligned on 32-bit boundaries

- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, and {nu1Workspace, 2*u2ModLength + 16}

37.3.7.6.7 Status Returned Values

Table 37-103. GF2NEcConvAffineToProjective Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.7.7 Randomize Coordinate

37.3.7.7.1 Purpose

This service is used to convert the Projective representation of a point to another Projective representation.

37.3.7.7.2 How to Use the Service

37.3.7.7.3 Description

The operation performed is:

$Projective(X_1, Y_1, Z_1) \rightarrow Projective(X_2, Y_2, Z_2)$

In this computation, the following parameters need to be provided:

- The input point is filled in projective coordinates (X,Y,Z) (pointed by {nu1PointBase,3*u2ModLength + 12}). This Point must not be the point at infinity.
- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase,u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 3*u2ModLength + 28}
- The random number (pointed by {nu1RandomBase, u2ModLength + 4}) The result is the point nu1PointBase with its (X,Y,Z) coordinates randomized. The service for this operation is GF2NEcRandomiseCoordinate.

Important:
Before using this service:

- Ensure that the constant Cns has been calculated with the Setup of the fast Modular Reductions service.
- Be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

37.3.7.7.4 Parameters Definition

Table 37-104. GF2NEcRandomiseCoordinate Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	Resulting point
nu1RandomBase	nu1	I	Crypto RAM	u2ModLength + 4	Random	Corrupted
nu1Workspace	nu1	I	Crypto RAM	3*u2ModLength + 28	–	Workspace

37.3.7.7.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _GF2NEcRandomiseCoordinate(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcRandomiseCoordinate(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcRandomiseCoordinate(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL_GF2NEcRandomiseCoordinate(nu1RandomBase) = <Base of the ram location where the the rng
is stored>;
PUKCL _GF2NEcRandomiseCoordinate(nu1PointBase) = <Base of the ram location of the point>;
PUKCL _GF2NEcRandomiseCoordinate(nu1Workspace) =
<Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcRandomiseCoordinate,&PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.7.7.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1RandomBase, nu1Workspace are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1RandomBase, u2ModLength + 4} and {nu1Workspace, 3*u2ModLength + 28}

37.3.7.7.7 Status Returned Values

Table 37-105. GF2NEcRandomiseCoordinate Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.

37.3.7.8 Point is on Elliptic Curve

37.3.7.8.1 Purpose

This service is used to test whether the point is on the curve.

37.3.7.8.2 How to Use the Service

37.3.7.8.3 Description

The operation performed is:

Status = IsPointOnCurve(X, Y, Z);

In this computation, the following parameters need to be provided:

- The input points filled in projective coordinates (X, Y, Z) (pointed by {nu1PointBase, 3*U2ModLength + 4}). This point can be point at infinity.
- AParam and BParam are the Elliptic Curve Equation parameters (pointed by {nu1AParam, u2ModLength+ 4} and {nu1BParam, u2ModLength + 4}).

- Cns the Fast Modular Constant filled (pointed by {nu1CnsBase, u2ModLength + 8})
- P the modulus filled (pointed by {nu1ModBase, u2ModLength + 8})
- The workspace not initialized (pointed by {nu1WorkSpace, 4*u2ModLength + 28})

The service name for this operation is `GF2NEcPointIsOnCurve`.

Important: Before using this service, the constant Cns must have been calculated with the Fast Modular Reduction service.

37.3.7.8.4 Parameters Definition

Table 37-106. GF2NEcPointIsOnCurve Service Parameters

Parameter	Type	Dir.	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ModLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1PointBase	nu1	I	Crypto RAM	3*u2ModLength + 12	Input point	Unchanged
nu1AParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter a	Unchanged
nu1BParam	nu1	I	Crypto RAM	u2ModLength + 4	The parameter b	Unchanged
nu1Workspace	nu1	I	Crypto RAM	4*u2ModLength + 28	N/A	Workspace

37.3.7.8.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _GF2NEcPointIsOnCurve(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcPointIsOnCurve(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcPointIsOnCurve(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcPointIsOnCurve(nu1PointABase) = <Base of the A point>;
PUKCL _GF2NEcPointIsOnCurve(nu1AParam) = <Base of the ram location of the parameter a>;
PUKCL _GF2NEcPointIsOnCurve(nu1BParam) = <Base of the ram location of the parameter b>;
PUKCL _GF2NEcPointIsOnCurve(nu1PointBase) = <Base of the ram location of the point>;
PUKCL _GF2NEcPointIsOnCurve(nu1Workspace) = <Base of the ram location of the workspace>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKC L_Process(GF2NEcPointIsOnCurve,
pvPUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.7.8.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1AParam, nu1BParam and nu1Workspace are not aligned on 32-bit boundaries

- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1AParam, u2ModLength + 4}, {nu1BParam, u2ModLength + 4}, {nu1Workspace, 4*u2ModLength + 28} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1AParam, u2ModLength + 4}, {nu1BParam, u2ModLength + 4} and {nu1Workspace, 4*u2ModLength + 28}

37.3.7.8.7 Status Returned Values

Table 37-107. GF2NEcPointIsOnCurve Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The point is on the curve.
PUKCL_POINT_IS_NOT_ON_CURVE	Warning	The point is not on the curve.
PUKCL_POINT_AT_INFINITY	Warning	The input point has its Z equal to zero, so it's a representation of the infinite point.

37.3.7.9 Generating an ECDSA Signature (Compliant with FIPS 186-2)

37.3.7.9.1 Purpose

This service is used to generate an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Generation. A hash value (HashVal) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

A scalar number must be provided, as described in the FIPS 186-2.

The result (R,S) is computed by this service. If S equals zero, the status is set to PUKCL_WRONG_SELECT_NUMBER.

37.3.7.9.2 How to Use the Service

37.3.7.9.3 Description

The operation performed is:

(*R*, *S*) = *EcDsaSign*(Pt_A, *HashVal*, *k*, *CurveParameters*, *PrivateKey*)

This service processes the following checks:

- If the Scalar Number k is out of the range [1, PointOrder -1], the calculus is stopped and the status is set to PUKCL_WRONG_SELECT_NUMBER.
- If R equals zero, the calculus is stopped and the status is set to PUKCL_WRONG_SELECT_NUMBER.
- If S equals zero, the calculus is stopped and the status is set to PUKCL_WRONG_SELECT_NUMBER. In this computation, the following parameters need to be provided:
- A the input point is filled in "mixed" coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointABase,3*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase,u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})
- The workspace not initialized (pointed by {nu1WorkSpace, 8*u2ModLength + 44}
- The a and b parameters relative to the elliptic curve equation (pointed by {nu1ABBase, 2*u2ModLength + 8})
- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase, u2ScalarLength + 4})
- k the input Scalar Number beforehand generated and filled (pointed by{nu1ScalarNumber,u2ScalarLength + 4})
- HashVal the hash value beforehand generated and filled (pointed by {nu1HashBase, u2ScalarLength +4})
- The Private Key (pointed by {nu1PrivateKey, u2ScalarLength +4})
- Generally u2ScalarLength is equal to (u2ModLength) or (u2ModLength + 4)

Important:
For the ECDSA signature generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

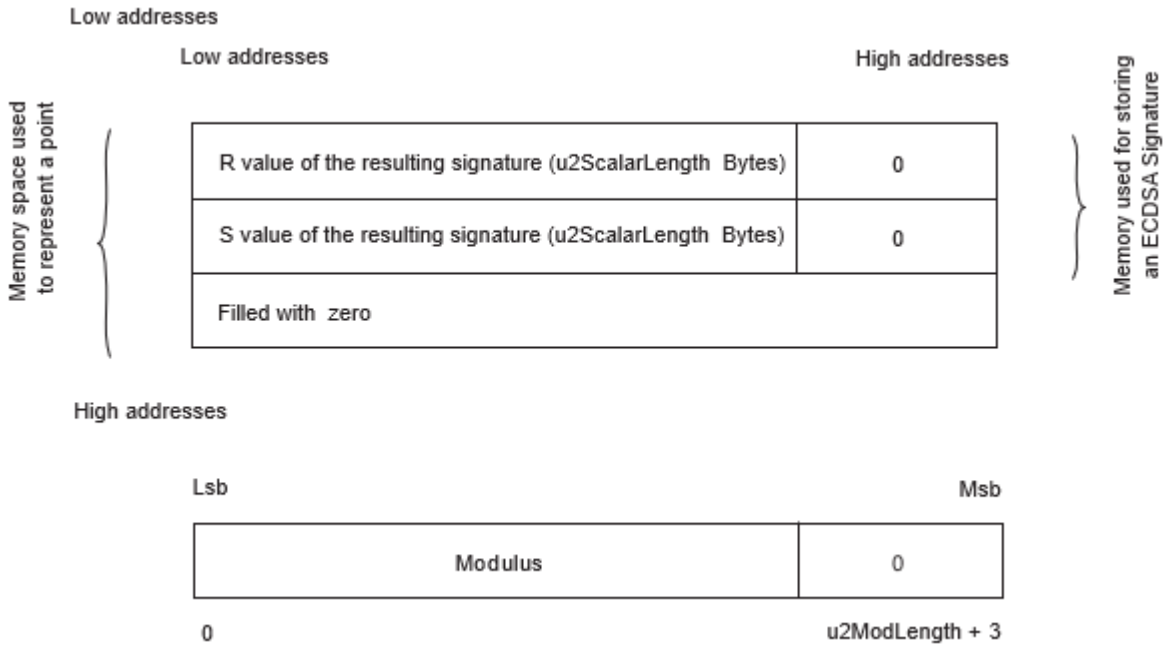
The scalar number k must be selected at random. This random must be generated before the call of the ECDSA signature. For this random generation be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

The operation performed is:

- Compute the ECDSA (R,S) as described in FIPS 186-2, but leaving the user the role of computing the input Hash Value, thus leaving the freedom of using any other algorithm than SHA-1.
- Compute a R value using the input A point and the scalar number.
- Compute a S value using R, the scalar number, the private key and the provided hash value. Note that the resulting signature (R,S) is stored at the place of the input A point.
- If all is correct and S is different from zero, the status is set to PUKCL_OK. If all is correct and S equals zero,the status is set to PUKCL_WRONG_SELECT_NUMBER. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

The service name for this operation is GF2NEcDsaGenerateFast. The fast mode is used, the fast modular reduction is used in the computations.

- The signature (R,S), when resulting from a computation is given back at address of the A point:
  - The R value result with u2ModLength + 4 bytes (padded with zeros).
  - The S value result with u2ModLength + 4 bytes (padded with zeros)
  - The u2NLength + 4 following bytes (space for the third coordinate of A) are filled with zeros.



37.3.7.9.4 Parameters Definition

Table 37-108. GF2NEcDsaGenerateFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 12	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1ScalarNumber	nu1	I	Crypto RAM	u2ScalarLength + 4	Scalar Number used to multiply the point A	Unchanged
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PrivateKey	nu1	I/O	Crypto RAM	u2ScalarLength + 4	Base of the Private Key	Unchanged
nu1HashBase(1)	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Unchanged
u2ScalarLength	u2	I	–	–	Length of scalar (same length as the length of order)	Length of scalar
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Input point A (three coordinates (X,Y) affine and Z = 1)	Resulting signature (R,S,0)
nu1ABase	nu1	I	Crypto RAM	2*u2ModLength + 8	Parameter a of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	–	Corrupted workspace

Note:

1. Whatever the chosen SHA, the resulting hash value may have a length inferior or equal to the modulo length and be padded with zeros until its total length is u2ModLength + 4.

37.3.7.9.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled
PUKCL _GF2NEcDsaGenerate(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcDsaGenerate(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcDsaGenerate(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcDsaGenerate(nu1PointABase) = <Base of the A point>;
PUKCL _GF2NEcDsaGenerate(nu1PrivateKey) = <Base of the Private Key>;
PUKCL _GF2NEcDsaGenerate(nu1ScalarNumber) = <Base of the ScalarNumber>;
PUKCL _GF2NEcDsaGenerate(nu1OrderPointBase) = <Base of the order of A point>;
PUKCL _GF2NEcDsaGenerate(nu1ABase) = <Base of the a parameter of the curve>; PUKCL
_GF2NEcDsaGenerate(nu1Workspace) = <Base of the workspace>;
PUKCL _GF2NEcDsaGenerate(nu1HashBase) = <Base of the SHA resulting hash>;
```

```
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcDsaGenerateFast, pvPUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else // Manage the error
```

37.3.7.9.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PrivateKey, nu1ScalarNumber, nu1OrderPointBase,nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12},{nu1PrivateKey, u2ScalarLength + 4},{nu1ScalarNumber, u2ScalarLength + 4},{nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PrivateKey, u2ScalarLength + 4}, {nu1ScalarNumber, u2ScalarLength + 4}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

37.3.7.9.7 Status Returned Values

Table 37-109. GF2NEcDsaGenerate Fast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without problem.
PUKCL_WRONG_SELECTNUMBER	Warning	The given value for nu1ScalarNumber is not good to perform this signature generation.

37.3.7.10 Verifying an ECDSA Signature (Compliant with FIPS 186-2)

37.3.7.10.1 Purpose

This service is used to verify an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Verification.

A hash value (HashVal) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

As second significant input, the Signature is provided to be checked. This service checks the signature and fills the status accordingly.

37.3.7.10.2 How to Use the Service

37.3.7.10.3 Description

The operation performed is:

Verify = *EcDsaVerifySignature*(Pt_A, *HashVal*, *Signature*, *CurveParameters*, *PublicKey*)

The points used for this operation are represented in different coordinate systems. In this computation, the following parameters need to be provided:

- A the input point is filled with the affine values (X,Y) and Z = 1 (pointed by{nu1PointABase,3*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase,u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength +4})

- The workspace not initialized (pointed by {nu1WorkSpace, 8*u2ModLength +44} The a and b parameters relative to the elliptic curve (pointed by {nu1ABase,2*u2ModLength + 8})
- The order of the Point A on the elliptic curve (pointed by {nu1OrderPointBase,u2ScalarLength +4})
- HashVal the hash value beforehand generated and filled (pointed by {nu1HashBase,u2ScalarLength +4})
- The Public Key point is filled in “mixed” coordinates (X,Y) with the affine values and Z = 1 (pointed by {nu1PointPublicKeyGen, 3*u2ModLength + 12})
- The input signature (R,S), even if it is not a Point, is represented in memory like a point in affine coordinates (X,Y) (pointed by {nu1PointSignature, 2*u2ScalarLength + 8})

Important: For the ECDSA signature verification be sure to follow the directives given for the RNG on the chip you use (particularly initialization, seeding) and compulsorily start the RNG.

- The operation consists in obtaining a V value with all these input parameter and check that V equals the provided R. If all is correct and the signature is the good one, the status is set to PUKCL_OK. If all is correct and the signature is wrong, the status is set to PUKCL_WRONG_SIGNATURE. If an error occurs, the status is set to the corresponding error value (see Status Returned Values below).

The service name for this operation is `GF2NEcDsaVerifyFast`. This service uses Fast mode and Fast Modular Reduction for computation.

37.3.7.10.4 Parameters Definition

Table 37-110. GF2NEcDsaVerifyFast Service Parameters

Parameter	Type	Direction	Location	Data Length	Before Executing the Service	After Executing the Service
nu1ModBase	nu1	I	Crypto RAM	u2ModLength + 4	Base of modulus P	Base of modulus P
nu1CnsBase	nu1	I	Crypto RAM	u2ScalarLength + 8	Base of Cns	Base of Cns
u2ModLength	u2	I	–	–	Length of modulus P	Length of modulus P
nu1OrderPointBase	nu1	I	Crypto RAM	u2ScalarLength + 4	Order of the Point A in the elliptic curve	Unchanged
nu1PointSignature	nu1	I	Crypto RAM	2*u2ScalarLength + 8	Signature(r, s)	Corrupted
nu1HashBase(1)	nu1	I	Crypto RAM	u2ScalarLength + 4	Base of the hash value resulting from the previous SHA	Corrupted
u2ScalarLength	u2	I	–	–	Length of scalar	Length of scalar
nu1PointABase	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Generator point	Corrupted
nu1PointPublicKeyGen	nu1	I/O	Crypto RAM	3*u2ModLength + 12	Public point	Corrupted
nu1ABase	nu1	I	Crypto RAM	2*u2ModLength + 8	Parameter a and b of the elliptic curve	Unchanged
nu1Workspace	nu1	I	Crypto RAM	8*u2ModLength + 44	–	Corrupted workspace

Note:

1. Whatever the chosen SHA, the resulting hash value may have a length inferior or equal to the modulo length and be padded with zeros until its total length is u2ModLength + 4.

37.3.7.10.5 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;

// ! The Random Number Generator must be initialized and started
// ! following the directives given for the RNG on the chip

PUKCL (u2Option) = 0;

// Depending on the option specified, not all fields must be filled PUKCL
_GF2NEcDsaVerify(nu1ModBase) = <Base of the ram location of P>;
PUKCL _GF2NEcDsaVerify(u2ModLength) = <Byte length of P>;
PUKCL _GF2NEcDsaVerify(nu1CnsBase) = <Base of the ram location of Cns>;
PUKCL _GF2NEcDsaVerify(nu1PointABase) = <Base of the A point>;
PUKCL _GF2NEcDsaVerify(nu1PrivateKey) = <Base of the Private Key>;
PUKCL _GF2NEcDsaVerify(nu1ScalarNumber) = <Base of the ScalarNumber>;
PUKCL _GF2NEcDsaVerify(nu1OrderPointBase) = <Base of the order of A point>;
PUKCL _GF2NEcDsaVerify(nu1ABase) = <Base of the a parameter of the curve>; PUKCL
_GF2NEcDsaVerify(nu1Workspace) = <Base of the workspace>;
PUKCL _GF2NEcDsaVerify(nu1HashBase) = <Base of the SHA resulting hash>;
...

// vPUKCL_Process() is a macro command, which populates the service name
// and then calls the library...
vPUKCL_Process(GF2NEcDsaVerifyFast, &PUKCLParam);
if (PUKCL (u2Status) == PUKCL_OK)
            {
            ...
            }
else
            if(PUKCL(u2Status) == PUKCL_WRONG_SIGNATURE)
            {
            ...
            }
else // Manage the error
```

37.3.7.10.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PointPublicKeyGen, nu1PointSignature, nu1OrderPointBase,nu1ABBase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PointPublicKeyGen, 3*u2ModLength + 12}, {nu1PointSignature,2*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABBase, 2*u2ModLength + 8}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PointPublicKeyGen, 3*u2ModLength + 12}, {nu1PointSignature, 2*u2ScalarLength + 8}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABBase, 2*u2ModLength + 8}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

37.3.7.10.7 Status Returned Values

Table 37-111. GF2NEcDsaVerifyFast Service Return Codes

Returned Status	Importance	Meaning
PUKCL_OK	–	The computation passed without errors. The signature is correct.
PUKCL_WRONG_SIGNATURE	Warning	The signature is incorrect.

37.3.8 PUKCL Requirements and Performance

37.3.8.1 Services Stack Usage

This library is using the main core to execute its computations, and therefore is also sharing some resources with the application.

It may be important for the application to know RAM usage by the library functions and to be aware that the library does not use any global variables.

The following table provides the minimum number of bytes used by the library that have to be available on the stacks to ensure that the functionality can be executed correctly. In some cases, the library may use less bytes than the specified number for some options. This table contains estimated values.

Table 37-112. Services Stack Usage

PUKCL Service	STACK Usage (Bytes)
SelfTest	112
ClearFlags	0
Swap	8
Fill	8
CondCopy	24
FastCopy	16
Smult	16
Smult (with reduction)	88
Comp	8
Fmult	24
Fmult (with reduction)	96
Square	16
Square (with reduction)	88
Div	144
GCD	136
RedMod (Setup)	160
RedMod (using fast reduction)	80
RedMod (randomize)	80
RedMod (Normalize)	80
RedMod (Using Division)	184
ExpMod	200
PrimeGen	416
CRT	304
ZpEccAddFast	104
ZpEccAddSubFast	92
ZpEcConvProjToAffine	280
ZpEcConvAffineToProjective	64

.....continued

PUKCL Service	STACK Usage (Bytes)
ZpEccDblFast	96
ZpEccMulFast	168
ZpEccQuickDualMulFast	216
ZpEcDsaGenerateFast	392
ZpEcDsaVerifyFast	456
ZpEcDsaQuickVerify	368
ZpEcRandomiseCoordinate	56
GF2NEccAddFast	128
GF2NEcConvProjToAffine	264
GF2NEcConvAffineToProjective	56
GF2NEccDblFast	136
GF2NEccMulFast	208
GF2NEcDsaGenerateFast	376
GF2NEcDsaVerifyFast	440
GF2NEcRandomiseCoordinate	56

37.3.8.2 Parameter Size Limits for Different Services

The following table lists parameter size limits for different services.

For the services ModExp, PrimeGen, and CRT, additional details are available in the service description.

Table 37-113. Parameter Size Limits

API	Min/Max Sizes	Comments
SelfTest	—	—
ClearFlags	—	—
Swap	4 bytes to 2044 bytes	Per block to be swapped
Fill	4 bytes to 4088 bytes	—
Fast Copy/Clear	4 bytes to 2044 bytes	Supposing Length(R) = Length(X)
Conditional Copy/Clear	4 bytes to 2044 bytes	Supposing Length(R) = Length(X)
Smult	4 bytes to 2040 bytes	Supposing Length(R) = Length(X) + 4 Bytes, No Z Parameter, No Reduction
Compare	4 bytes to 2044 bytes	Supposing Length(X) = Length(Y)
FMult	Input: 4 bytes to 1020 bytes Output: 4bytes to 2040 bytes	Supposing Length(Y) = Length(X), No Z Parameter, No Reduction
Square	Input: 4 bytes to 1020 bytes Output: 4 bytes to 2040 bytes	Supposing No Z Parameter, No Reduction
Euclidean Division	Divider: 8 to 1016 bytes Num.: 8 to 2032 bytes	Supposing Length(Num) = 2*Length(Divider)

.....continued

API	Min/Max Sizes	Comments
Mod. inv. / GCD	8 to 1012 bytes	—
ModRed	Modulus: 12 to 1016 bytes Input: 24 to 2032 bytes	Supposing RBase = XBase
Fast ModExp Exp in Crypto RAM	12 to 576 bytes (96 to 4608 bits)	Supposing Length(Exponent) = Length(Modulus), Window Size = 1 With the Exponent in Crypto RAM
Fast ModExp Exp not in Crypto RAM	12 to 672 bytes (96 to 5376 bits)	Supposing Length(Exponent) = Length(Modulus), Window Size = 1 With the Exponent not in Crypto RAM
Prime Gen.	Prime Number: 12 to 448 bytes (96 to 3584 bits)	Supposing Window Size = 1
CRT	Modulus = Two Primes: Size of one prime from 24 to 448 bytes Modulus = from 48 to 896 bytes (384 to 7168 bits)	Supposing Length(Exponent) = Length(Modulus), Window Size = 1
ECC Addition qnd Subtraction GF(p)	Modulus: 12 to 308 bytes	—
ECC Doubling GF(p)	Modulus: 12 to 400 bytes	—
ECC Multiplication GF(p)	Modulus: 12 to 264 bytes	Supposing Length(Scalar) = Length(Modulus)
ECC Quick Dual Multiplication GF(p)	Modulus: 12 to 152 bytes	—
ECDSA Generate GF(p)	Modulus: 12 to 220 bytes (up to 521 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECDSA Verify GF(p)	Modulus: 12 to 188 bytes (up to 521 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECC Addition GF(2n)	Modulus: 12 to 248 bytes	—
ECC Doubling GF(2n)	Modulus: 12 to 364 bytes	—
ECC Multiplication GF(2n)	Modulus: 12 to 250 bytes	Supposing Length(Scalar) = Length(Modulus)
ECDSA Generate GF(2n)	Modulus: 12 to 208 bytes (up to 571 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECDSA Verify GF(2n)	Modulus: 12 to 180 bytes (up to 571 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)
ECDSA Quick Verify GF(2n)	Modulus: 12 to 140 bytes (up to 571 bits for common curves)	Supposing Length(Scalar) = Length(Modulus)

37.3.8.3 Service Timing

The values in the following tables are estimated performances for CPU clock of 64 MHz. The CPU and PUKCC are operated at the same frequency. Due to possible change in the parameters values, the measurements show approximated values.

Other test conditions:

- PUKCL library data in Crypto RAM
- Test code and test data in SRAM
- ICache and DCache are disabled

37.3.8.3.1 Service Timing for RSA

RSA uses the ExpMod service for encryption and decryption. Following tables show service timing, where 'W' indicates window size.

Table 37-114. RSA1024

Operation	Clock Cycles	Timing one block
RSA 1024 decryption / signature generation. No CRT, Regular implementation, W=4	3.05 MCycles	47.799 ms
RSA 1024 decryption / signature generation. With CRT, Regular implementation, W=4	1.09 MCycles	17.109 ms
RSA 1024 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=3	0.07 MCycles	1.141 ms
RSA 1024 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=0x10001	0.07 MCycles	1.129 ms

Table 37-115. RSA2048

Operation	Clock Cycles	Timing One block
RSA 2048 decryption / signature generation. No CRT, Regular implementation, W=4	21.6 MCycles	338.249 ms
RSA 2048 decryption / signature generation. With CRT, Regular implementation, W=4	6.36 MCycles	99.408 ms
RSA 2048 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=3	0.24 MCycles	3.843 ms
RSA 2048 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=0x10001	0.24 MCycles	3.827 ms

Table 37-116. RSA4096

Operation	Clock Cycles	Timing One block
RSA 4096 Decryption / signature generation. No CRT, Regular implementation, W=1	209 MCycles	3.2742s
RSA 4096 Decryption / signature generation. With CRT, Regular implementation, W=3	46.1 MCycles	720.95 ms
RSA 4096 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=3	0.91 MCycles	14.346 ms

.....continued		
Operation	**Clock Cycles**	**Timing One block**
RSA 4096 encryption / signature verification. No CRT, Fast implementation, W=1 Exponent=0x10001	0.91 MCycles	14.337 ms

37.3.8.3.2 Service Timing for Prime Generation

Prime generation uses the PrimeGen service.

Table 37-117. Prime Generation

Operation	Clock Cycles	Timing One Block
Regular Generation of two primes, Prime_Length=512 bits, W=4, Rabin Miller Iterations Number = 3, (average of 200 samples)	Mean = 47.4 MCycles	Mean = 0.4s
Regular Generation of two primes, Prime_Length=512 bits, W=4, Rabin Miller Iterations Number = 3, (Standard Deviation for 200 samples)	Std Dev = 30.3 Mcycles	Std Dev = 0.47s
Regular Generation of two primes, Prime_Length=1024 bits, W=4, Rabin Miller Iterations Number = 3, (average of 200 samples)	Mean = 419.71 MCycles	Mean = 6.558s
Regular Generation of two primes, Prime_Length=1024 bits, W=4, Rabin Miller Iterations Number = 3, (Standard Deviation for 200 samples)	Std Dev = 294 Mcycles	Std Dev = 4.59s
Regular Generation of two primes, Prime_Length=2048 bits, W=4, Rabin Miller Iterations Number = 3, (average of 200 samples)	Mean = 4.78 GCycles	Mean = 74.68s
Regular Generation of two primes, Prime_Length=2048 bits, W=4, Rabin Miller Iterations Number = 3, (Standard Deviation for 200 samples)	Std Dev = 3.05 GCycles	Std Dev = 47.65s

37.3.8.3.3 Service Timing for ECDSA on Prime Field

In the following table, ECDSA signature generation uses the ZpEcDsaGenerateFast service and signature verification uses ZpEcDsaQuickVerify

Table 37-118. ECDSA GF(p)

Operation	Clock Cycles	Timing One block
ECDSA GF(p) 256 Generate Fast	2.67 MCycles	41.864 ms
ECDSA GF(p) 256 Verify Quick W=(4,4) Scalar in PUKCC RAM	1.84 MCycles	28.888 ms
ECDSA GF(p) 384 Generate Fast	6.18 MCycles	96.712 ms
ECDSA GF(p) 384 Verify Quick W=(4,4) Scalar in PUKCC RAM	4.15 MCycles	64.868 ms
ECDSA GF(p) 521 Generate Fast	13.36 MCycles	208.869 ms
ECDSA GF(p) 521 Verify Quick W=(4,4) Scalar in PUKCC RAM	8.81 MCycles	137.711 ms

37.3.8.3.4 Service Timing for ECDSA on Binary Field

In the following table, ECDSA signature generation uses the GF2NEcDsaGenerateFast service and signature verification uses GF2NEcDsaVerifyFast

Table 37-119. ECDSA GF(2^n)

Operation	CPU Cycles	Timing One block
ECDSA GF(2^n) B283 Generate Fast	3.21 MCycles	50.301 ms
ECDSA GF(2^n) B283 Verify	6.40 MCycles	100.150 ms
ECDSA GF(2^n) B409 Generate Fast	6.94 Mcycles	108.554 ms
ECDSA GF(2^n) B409 Verify	13.73 Mcycles	214.571 ms
ECDSA GF(2^n) B571 Generate Fast	15.08 Mcycles	235.704 ms
ECDSA GF(2^n) B571 Verify	30.07 MCycles	469.972 ms

38. Analog-to-Digital Converter (ADC)

38.1 Overview

The PIC32CX-BZ2 12-bit High Speed Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC) includes the following features:

- 12-bit resolution
- One ADC module, up to 2 Msps conversion rate
- Single-ended and/or differential input
- Supported in Sleep mode
- Two digital comparators
- Two digital filters supporting two modes:
  - Oversampling mode
  - Averaging mode
- Designed for motor control, power conversion and general purpose applications

The PIC32CX-BZ2 has one shared ADC module. This ADC module incorporates a multiplexer on the input to facilitate a group of inputs and provides a flexible automated scanning option through the input scan logic.

For the ADC module, the analog inputs are connected to the Sample and Hold (S&H) capacitor. The ADC module performs the conversion of the input analog signal based on the configurations set in the registers. When the conversion is complete, the final result is stored in the result buffer for the specific analog input and is passed to the digital filter and digital comparator if configured to use data from this particular sample.

Equation 38-1. ADC Throughput Rate

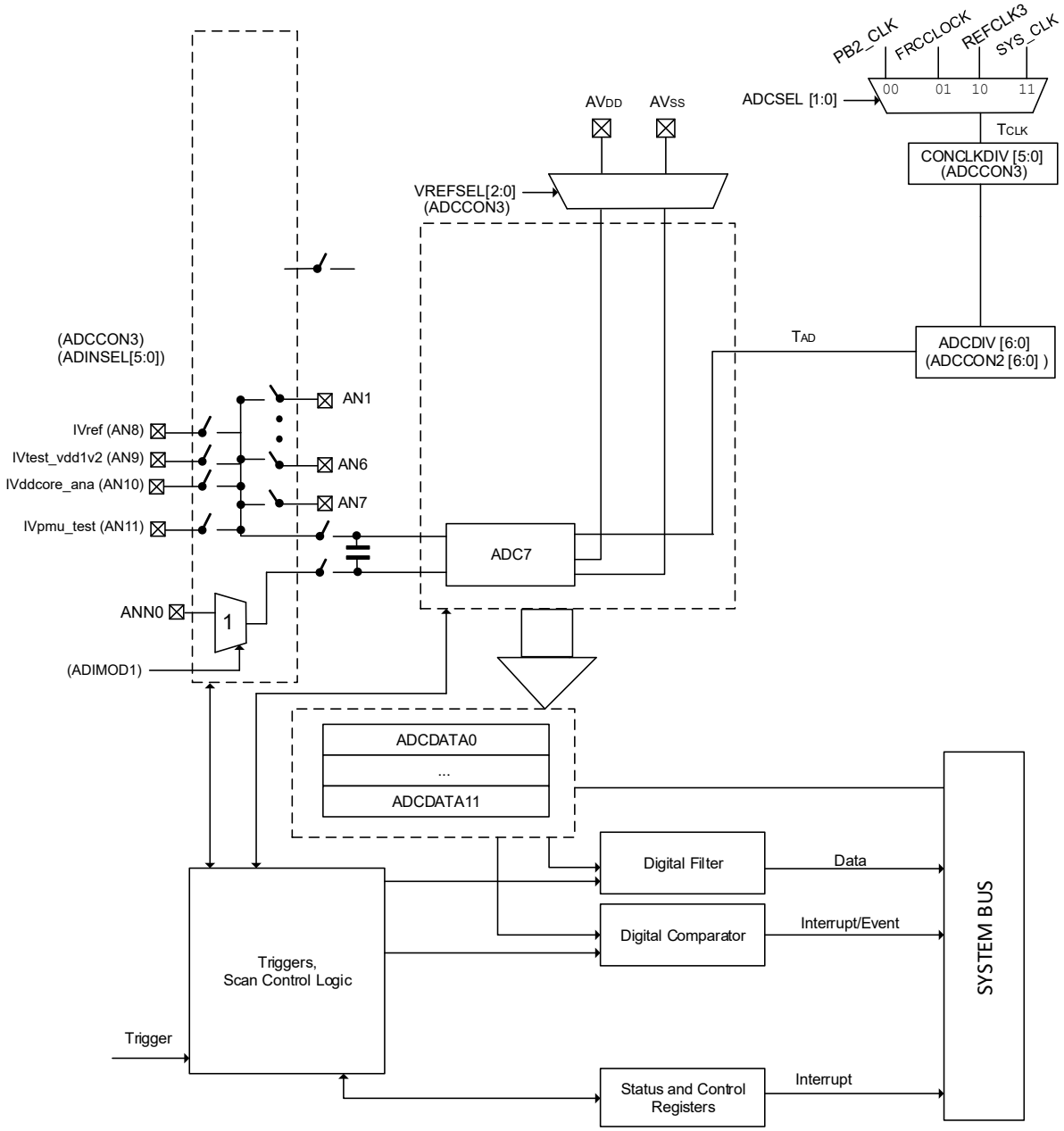
$$FTP = \frac{T_{AD}}{T_{SAMP} + T_{CONV}}$$

Where,

- T_{AD} = The frequency of the individual ADC module.

A block diagram of the ADC module is illustrated in the following figure.

Figure 38-1. ADC Block Diagram



38.2 ADC Operation

The High Speed Successive Approximation Register (SAR) ADC is designed to support power conversion and motor control applications and consists of one shared ADC module. The shared ADC module has multiple analog inputs connected to its S&H circuit through a multiplexer. Multiple analog inputs share this ADC; therefore, it is termed the shared ADC module. The shared ADC module is used to measure analog signals of lower frequencies and signals that are static in nature (in other words, do not change significantly with time). However, this ADC module is capable of up to 2 Msps sample rate.

The analog inputs connected to the shared ADC module are Class 2 and Class 3 inputs. The number of inputs designated for each class depends on the specific device. For the PIC32CX-BZ2, the following arrangement is provided.

- Class 2 = AN0 to AN5
- Class 3 = AN6 to AN7

The property of each class of analog input is described in the following table.

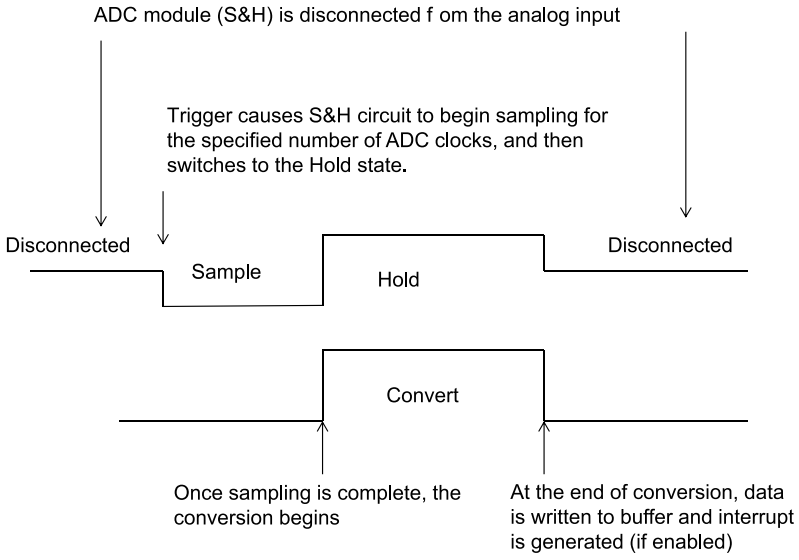
Table 38-1. Analog Input Class

ADC Module	Analog Input Class	Trigger	Trigger Action
Shared ADC module	Class 2	Individual trigger source or scan trigger	Starts sampling sequence or begins scan sequence
Shared ADC module with input scan	Class 3	Scan trigger	Starts scan sequence

Class 2 and Class 3 analog input properties:

- Class 2 inputs are used on the shared ADC module, either individually triggered or as part of a scan list. When used individually, they are triggered by their unique trigger selected by the ADCTRGx register.
- The analog inputs on the shared ADC have a natural order of priority (for example, AN6 has a higher priority than AN7).
- Class 3 inputs are used exclusively for scanning and share a common trigger source (scan trigger).
- Class 3 analog inputs share both the ADC module and the trigger source; therefore, the only method possible to convert them is to scan them sequentially for each incoming scan trigger event, where scanning occurs in the natural order of priority.
- The arrival of a trigger in the shared ADC module only starts the sampling. When the trigger arrives, the ADC module goes into sampling mode for the sampling time decided by the SAMC[9:0] bits (ADCCON2[25:16]). At the end of sampling, the ADC starts conversion. Upon completion of conversion, the ADC module is used to convert the next in line Class 2 or Class 3 inputs according to the natural order of priority. When a shared analog input (Class 2 or Class 3) has completed all conversion and no trigger is pending, the ADC module is disconnected from all analog inputs

  Figure 38-2. Sample and Conversion Sequence for Shared ADC Modules

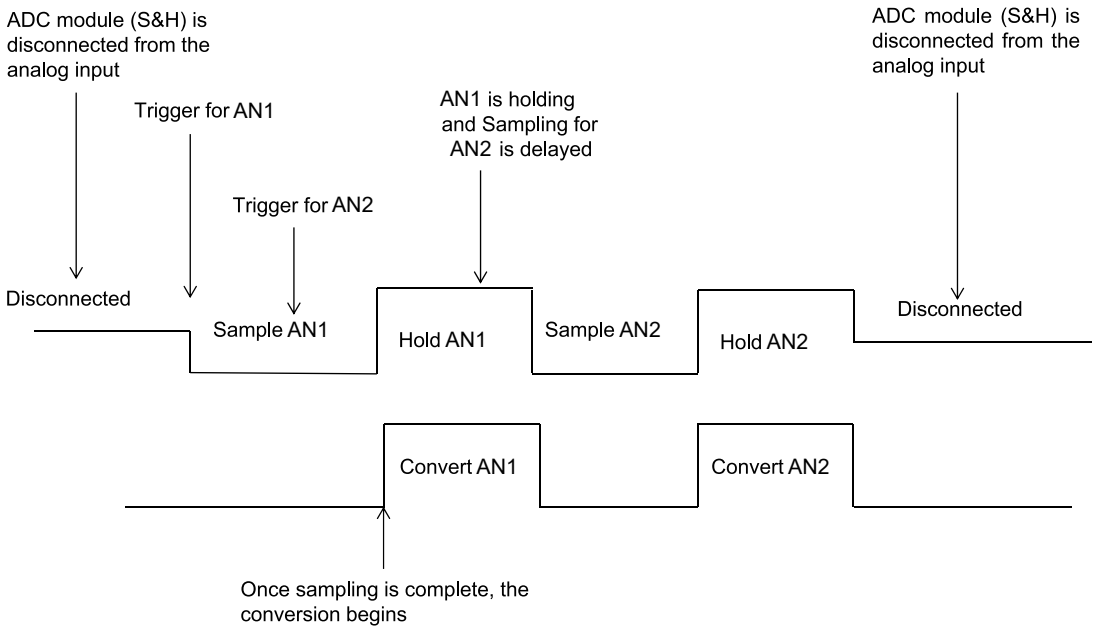


38.2.1 Class 2 Triggering

When a single Class 2 input is triggered, it is sampled and converted by the shared S&H using the sequence illustrated in Sample and Conversion Sequence for the Shared ADC Modules figure; see *Sample and Conversion Sequence for Shared ADC Modules* figure in the *ADC Operation* from Related Links. When multiple Class 2 inputs are triggered, it is important to understand the consequences of trigger timing. If a conversion is underway and

another Class 2 trigger occurs, then the sample-hold-conversion for the new trigger is stalled until the in-process, sample-hold cycle is complete, as shown in the following figure.

Figure 38-3. Multiple Independent Class 2 Trigger Conversion Sequence



When multiple inputs to the shared S&H are triggered simultaneously, the processing order is determined by their natural priority (the lowest numbered input has the highest priority). As an example, if AN1, AN2 and AN3 are triggered simultaneously, AN1 is sampled and converted first, followed by AN2 and finally, AN3. When using the independent Class 2 triggering on the shared S&H, the SAMC[9:0] bits (ADCCON2[25:16]) determine the sample time for all inputs while the appropriate TRGSRC[4:0] bits in the ADCTRGx Register (see *ADCTRG1* register from Related Links) determine the trigger source for each input.

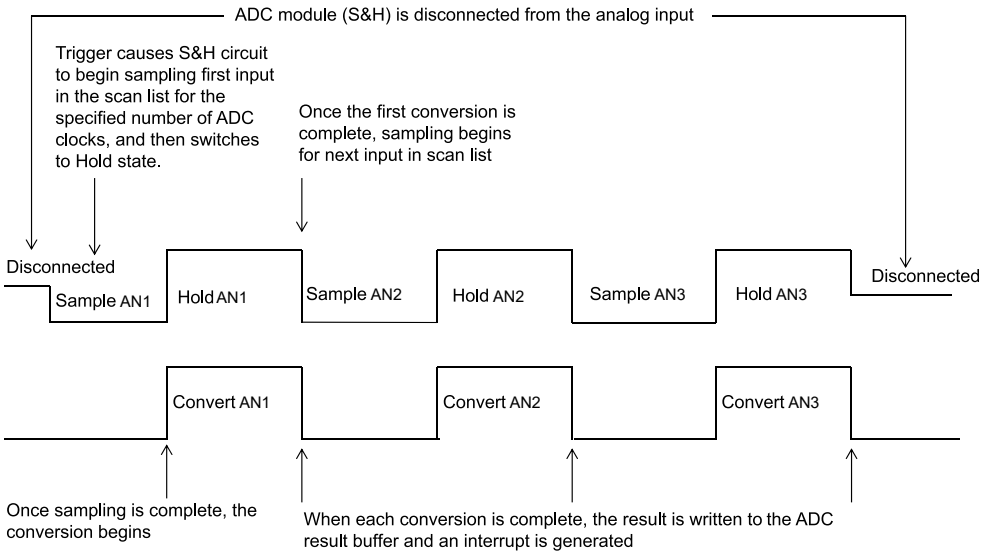
Related Links

38.11.15. ADCTRG1
38.2. ADC Operation

38.2.2 Input Scan

Input scanning is a feature that allows an automated scanning sequence of multiple Class 2 or Class 3 inputs. All Class 2 and Class 3 inputs are scanned using the single shared S&H. The selection of analog inputs for scanning is done with the CSSx bits of the ADCCSS1 registers. Class 2 inputs are triggered using STRIG selection in the ADCTRGx register, and Class 3 inputs are triggered using the STRGSRC[4:0] of the ADCCON1[20:16] register. When a trigger occurs for Class 2 or Class 3 inputs, the sampling and conversion occur in the natural input order is used; lower number inputs are sampled before higher number inputs.

Figure 38-4. Input Scan Conversion Sequence for Three Class 2 Inputs

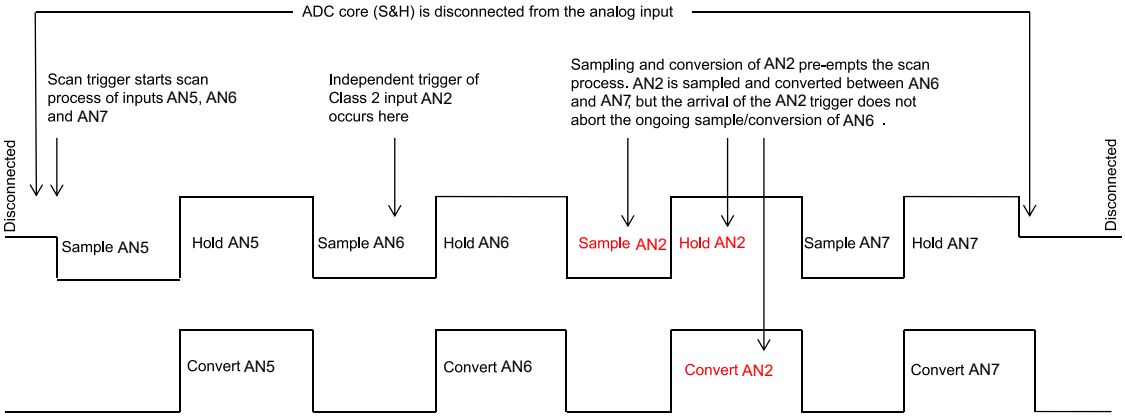


When using the shared analog inputs in scan mode, the SAMC[9:0] bits in the ADC Control Register 2 (ADCCON2[25:16]) determine the sample time for all inputs, while the Scan Trigger Source Selection bits (STRGSRC[4:0]) in the ADC Control Register 1 (ADCCON1[20:16]) determine the trigger source.

To ensure predictable results, a scan must not be retriggered until a sampling of all inputs is complete. Ensure system design to preclude retriggering a scan while a scan is in progress.

Individual Class 2 triggers that occur during a scan preempts the scan sequence if they are a higher priority than the sample currently being processed. In the following figure, a scan of AN5, AN6 and AN7 is underway when an independent trigger of Class 2 input AN2 takes place. The scan is interrupted for the sampling and conversion of AN2.

Figure 38-5. Scan Conversion Pre-empted by Class 2 Input Trigger



38.3 ADC Module Configuration

Operation of the ADC module is directed through bit settings in the specific registers. The following instructions summarize the actions and the settings. The options and details for each configuration step are provided in the subsequent sections.

To configure the ADC module, perform the following steps:

1. Configure the analog port pins as described in 38.3.1. Configuring the Analog Port Pins.
2. Select the analog inputs to the ADC multiplexers as described in 38.3.2. Selecting the ADC Multiplexer Analog Inputs.
3. Select the format of the ADC result as described in 38.3.3. Selecting the Format of the ADC Result.
4. Select the conversion trigger source as described in 38.3.4. Selecting the Conversion Trigger Source.
5. Select the voltage reference source as described in 38.3.5. Selecting the Voltage Reference Source.
6. Select the scanned inputs as described in 38.3.6. Selecting the Scanned Inputs.
7. Select the analog-to-digital conversion clock source and prescaler as described in 38.3.7. Selecting the Analog-to-Digital Conversion Clock Source and Prescaler.
8. Specify any additional acquisition time (if required) as described in 38.9. ADC Sampling Requirements.
9. Turn on the ADC module as described in 38.3.8. Turning ON the ADC.
10. Poll (or wait for the interrupt) for the voltage reference to be ready as described in 38.3.5. Selecting the Voltage Reference Source.
11. Enable the analog and bias circuit for the required ADC modules, and, after the ADC module wakes up, enable the digital circuit as described in 38.6.3. Low-Power Mode.
12. Configure the ADC interrupts (if required) as described in 38.5. Interrupts.

38.3.1 Configuring the Analog Port Pins

The ANSELx registers for the I/O ports associated with the analog inputs are used to configure the corresponding pin as an analog or a digital pin. A pin is configured as an analog input when the corresponding ANSELx bit = '1'. When the ANSELx bit = '0', the pin is set to digital control. The ANSELx registers are set when the device comes out of Reset, causing the ADC input pins to be configured as analog inputs by default.

The TRISx registers control the digital function of the port pins. The port pins that are required as analog inputs must have their corresponding bit set in the specific TRISx register, configuring the pin as an input. If the I/O pin associated with an ADC input is configured as an output by clearing the TRISx bit, the port's digital output level (V_{OH} or V_{OL}) is converted. After a device Reset, all of the TRISx bits are set. For more information on port pin configuration, see *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.

Note: When reading a PORT register that shares pins with the ADC, any pin configured as an analog input reads as '0' when the PORT latch is read. Analog levels on any pin that is defined as a digital input but not configured as an analog input, may cause the input buffer to consume the current that exceeds the device specification.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

38.3.2 Selecting the ADC Multiplexer Analog Inputs

The ADC module has two inputs, referred to as the positive and negative inputs. Input selection options vary as described in the following sections.

38.3.2.1 Selection of Positive Inputs

For the shared ADC module, the positive input is shared among all Class 2 and Class 3 inputs. Input connection of the analog input ANx to the shared ADC is automatic for either the Class 2 input trigger or during a scan of Class 2 and or Class 3 inputs. Selecting inputs for scanning is described in *Selecting the Scanned Inputs* from Related Links.

Related Links

38.3.6. Selecting the Scanned Inputs

38.3.2.2 Selection of Negative Inputs

Negative input selection is determined by the setting of the DIFFx bit of the ADCIMCON1 register. The DIFFx bit allows the inputs to be rail-to-rail and either single-ended or differential. The SIGNx and DIFFx bits in the ADCIMCON1 register scale the internal ADC analog inputs and reference voltages and configure the digital result to align with the expected full-scale output range.

For the shared ADC module, the analog inputs have individual settings for the DIFFx bit. Therefore, the user has the ability to select certain inputs as single-ended and others as differential while being connected to the same shared ADC module. While sampling, the signal changes on-the-fly as single-ended or differential according to its corresponding DIFFx bit setting.

Table 38-2. Negative Input Selection

ADCIMCON1		Input Configuration	Input Voltage		Output
DIFFx	SIGNx				
1	1	Differential 2's complement	Minimum input	$V_{IN}P - V_{IN}N = -V_{REF}$	-2048
			Maximum input	$V_{IN}P - V_{IN}N = V_{REF}$	+2047
1	0	Differential unipolar	Minimum input	$V_{IN}P - V_{IN}N = -V_{REF}$	0
			Maximum input	$V_{IN}P - V_{IN}N = V_{REF}$	+4095
0	1	Single-ended 2's complement	Minimum input	$V_{IN}P = V_{REF}$	-2048
			Maximum input	$V_{IN}P - V_{IN}N = V_{REF}$	+2047
0	0	Single-ended unipolar	Minimum input	$V_{IN}P = V_{REF}$	0
			Maximum input	$V_{IN}P - V_{IN}N = V_{REF}$	+4095

Legend:

- $V_{IN}P$ = Positive S&H input
- $V_{IN}N$ = Negative S&H input
- $V_{REF} = V_{REFH} - V_{REFL}$

Note: For proper operation and to prevent device damage, input voltage levels must not exceed the limits listed in the Electrical Specifications.

38.3.3 Selecting the Format of the ADC Result

The data in the ADC Result register can be read in any of the four supported data formats. The user can select from unsigned integer, signed integer, unsigned fractional or signed fractional. Integer data is right-justified and fractional data is left-justified.

- The integer or fractional data format selection is specified globally for all analog inputs using the Fractional Data Output Format bit, FRACT (ADCCON1[23]).
- The signed or unsigned data format selection can be independently specified for each individual analog input using the SIGNx bits in the ADCIMCONx registers
  .

The following table provides how a result is formatted.

Table 38-3. ADC Result Format

FRACT	SIGNx	Description	32-bit Output Data Format			
0	0	Unsigned integer	0000	0000	0000	0000
			0000	dddd	dddd	dddd
0	1	Signed integer	ssss	ssss	ssss	ssss
			ssss	sddd	dddd	dddd
1	0	Fractional	dddd	dddd	dddd	0000
			0000	0000	0000	0000
1	1	Signed fractional	sddd	dddd	dddd	dddd
			0000	0000	0000	0000

The following code is an example for ADC Class 2 configuration and fractional format.

```
int main(int argc, char** argv) {
int result[3];

/* Configure ADCCON1 */
ADCCON1bits.FRACT = 1;    // use Fractional output format ADCCON1bits.SELRES = 3;   // ADC
resolution is 12 bits ADCCON1bits.STRGSRC = 0;  // No scan trigger.

/* Configure ADCCON2 */
ADCCON2bits.SAMC = 5;         // ADC sampling time = 5 * TAD7
ADCCON2bits.ADCDIV = 1;       // ADC clock freq is half of control clock = TAD7

/* Initialize warm up time register */ ADCANCON = 0;
ADCANCONbits.WKUPCLKCNT = 5;  // Wakeup exponent = 32 * TADx

/* Clock setting */ ADCCON3 = 0;
ADCCON3bits.ADCSEL = 0;       // Select input clock source
ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
ADCCON3bits.VREFSEL = 0;      // Select AVDD and AVSS as reference source

/* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
ADCTRGMODEbits = 0;

/* Select ADC input mode */
ADCIMCON1bits.SIGN7 = 0;  // unsigned data format ADCIMCON1bits.DIFF7 = 0;  // Single ended
mode ADCIMCON1bits.SIGN8 = 0;  // unsigned data format ADCIMCON1bits.DIFF8 = 0;  // Single
ended mode ADCIMCON1bits.SIGN9 = 0;  // unsigned data format ADCIMCON1bits.DIFF9 = 0;  //
Single ended mode

/* Configure ADCGIRQENx */
ADCGIRQEN1 = 0;           // No interrupts are used
ADCGIRQEN2 = 0;

/* Configure ADCCSSx */
ADCCSS1 = 0;              // No scanning is used
ADCCSS2 = 0;

/* Configure ADCCMPCONx */
ADCCMPCON1 = 0;           // No digital comparators are used. Setting the ADCCMPCONx
ADCCMPCON2 = 0;           // register to '0' ensures that the comparator is disabled.
ADCCMPCON3 = 0;           // Other registers are “don't care”.
ADCCMPCON4 = 0;
ADCCMPCON5 = 0; ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0;             // No oversampling filters are used. ADCFLTR2 = 0;
ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;
/* Set up the trigger sources */
ADCTRGSNSbits.LVL7 = 0;   // Edge trigger ADCTRGSNSbits.LVL8 = 0;   // Edge trigger
ADCTRGSNSbits.LVL9 = 0;   // Edge trigger
ADC1TRG2bits.TRGSRC7 = 1; // Set AN7 to trigger from software
ADC2TRG3bits.TRGSRC8 = 1; // Set AN8 to trigger from software
ADC2TRG3bits.TRGSRC9 = 1; // Set AN9 to trigger from software
/* Early interrupt */
ADCEIEN1 = 0;                // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */ ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT);   // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN7 = 1;       // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY7); // Wait until ADC7 is ready

/* Enable the ADC module */ ADCCON3bits.DIGEN7 = 1;       // Enable ADC7

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;
```

```
/* Wait the conversions to complete */
while (ADCDSTAT1bits.ARDY7 == 0);
/* fetch the result */
result[0] = ADCDATA7;

while (ADCDSTAT1bits.ARDY8 == 0);
/* fetch the result */
result[1] = ADCDATA8;

while (ADCDSTAT1bits.ARDY9 == 0);
/* fetch the result */
result[2] = ADCDATA9;

/*
* Process results here
*
* Note 1: Loop time determines the sampling time since all inputs are Class 2.
* If the loop time happens is small and the next trigger happens before the
* completion of set sample time, the conversion will happen only after the
* sample time has elapsed.
*
* Note 2: Results are in fractional format
*
*/
}
return (1);
}
```

38.3.4 Selecting the Conversion Trigger Source

Class 2 inputs to the ADC module can be triggered for conversion either individually or as part of a scan sequence. Class 3 inputs can only be triggered as part of a scan sequence. Individual or scan triggers can originate from an on-board timer or output compare peripheral event, from external digital circuits connected to INT0, from external analog circuits connected to an analog comparator or through software by setting a trigger bit in an SFR.

Note: When conversion triggers for multiple Class 2 analog inputs occur simultaneously, they are prioritized according to a natural order priority scheme based on the analog input used. AN6 has the highest priority, AN7 has the next highest priority and so on.

38.3.4.1 Trigger Selection Class 2 Inputs

For each one of the Class 2 inputs, the user application can independently specify a conversion trigger source. The individual trigger source for an analog input ‘x’ is specified by the TRGSRC[4:0] bits located in registers ADCTRG1 through ADCTRG3. For example, these trigger sources may include:

- **General Purpose (GP) Timers**: When a period match occurs for the 32-bit timer, Timer3/2 or Timer5/4, or the 16-bit Timer1, Timer3 or Timer5, a special ADC trigger event signal is generated by the timer. This feature does not exist for other timers. For more information, see *Timer/Counter (TC)* from Related Links.
- **Output Compare**: The Output Compare peripherals, OC1, OC3 and OC5, can be used to generate an ADC trigger, then the output transitions from a low to high state. For more information, see *Timer/Counter (TC)* from Related Links.
- **Comparators**: The analog Comparators can be used to generate an ADC trigger when the output transitions from a low state to a high state. For more information, see *Digital Comparator* from Related Links.
- **External INT0 Pin Trigger**: In this mode, the ADC module starts a conversion on an active transition on the INT0 pin. The INT0 pin may be programmed for either a rising edge input or a falling edge input to trigger the conversion process.
- **Global Software Trigger**: The ADC module can be configured for manually triggering a conversion for all inputs that have selected this trigger option. The user can manually trigger a conversion by setting the Global Software Trigger bit, GSWTRG (ADCCON3[6]).

Related Links

38.4.1. Digital Comparator
40. Timer/Counter (TC)

38.3.4.2 Conversion Trigger Sources and Control

The following are the possible sources for each trigger signal:

- External trigger selection through the TRGSRCx[4:0] bits in the ADCTRGx registers. This capability is supported only for Class 2 analog inputs. Typically, the user specifies a particular trigger source to initiate a conversion for specific input. All of the analog inputs may select the same trigger source if desired. In such an event, the result resembles a “scanned conversion”, which has its order of completion enforced by the priority of the inputs associated with the same trigger source. The first trigger selection is 00000 (no trigger), which amounts to temporarily disabling that particular trigger and, consequently, temporarily disabling that analog input from being converted. The next two selections for trigger source (GSWTRG and GLSWTRG) are software-generated trigger sources. The second software-generated trigger selection is the Global Software Trigger (GSWTRG). This trigger links to the GSWTRG bit in the ADCCON3 register, which may be used to enable the user application to initiate a single conversion. GSWTRG is a self-clearing bit; therefore, it clears itself on the next ADC clock cycle after being set by the user application. The third software-generated trigger selection is the Global Level Software Trigger (GLSWTRG), which is linked to the GLSWTRG bit in the ADCCON3 register. This trigger may be used by the user application to initiate a burst of consecutive samples as the GLSWTRG bit is not self-clearing. The fourth trigger selection is a special selection, the Scan Trigger selection, which allows the Class 2 analog inputs to be included as members of a global scan of all inputs.
- Scanned trigger selection via the STRGSRC[4:0] bits in the ADCCON1 register and select bits in the ADCCSS1 registers. This mode is typically used to initiate the conversion of a group of analog inputs. This capability works for 2 and 3 analog inputs but is typically used for Class 3 inputs because they do not have individual associated TRGSRC bits. One of the trigger selections is the GSWTRG bit in the ADCCON3 register, which may be used to enable the user software to initiate a conversion.
- User initiated trigger via the ADINSEL[5:0] bits and the RQCNVRT bit in the ADCCON3 register. This mode enables the user application to create an individual conversion trigger request for a specified analog input. Using this mode enables the user application to trigger the conversion of an input without changing the trigger source configuration of the ADC. This is useful in handling error situations where another software module wants ADC information without disrupting the normal operation of the ADC. This is also the preferred method to generate the initial trigger to start a digital filter sequence.
- User-controlled sampling of Class 2 and Class 3 inputs via the ADINSEL[5:0] bits and the SAMP bit in the ADCCON3 register. Setting the SAMP bit causes the Class 2 and Class 3 inputs to be in Sampling mode while ignoring the selection of the SAMC[9:0] bits. This mode is also useful in software conversion of ADC with software-selectable sample time.
- External module (such as PTG) may specify an analog input for conversion via the setting of the ECRIEN bit in the ADCCON2 register. This method operates independently of the normal TRGSRC and STRGSRC methods. External modules may still use individual trigger signals and initiate conversions via the normal TRGSRC and STRGSRC methods.

38.3.4.3 User-Requested Individual Conversion Trigger (Software ADC Conversion) (Only for Class 2 and Class 3 Inputs)

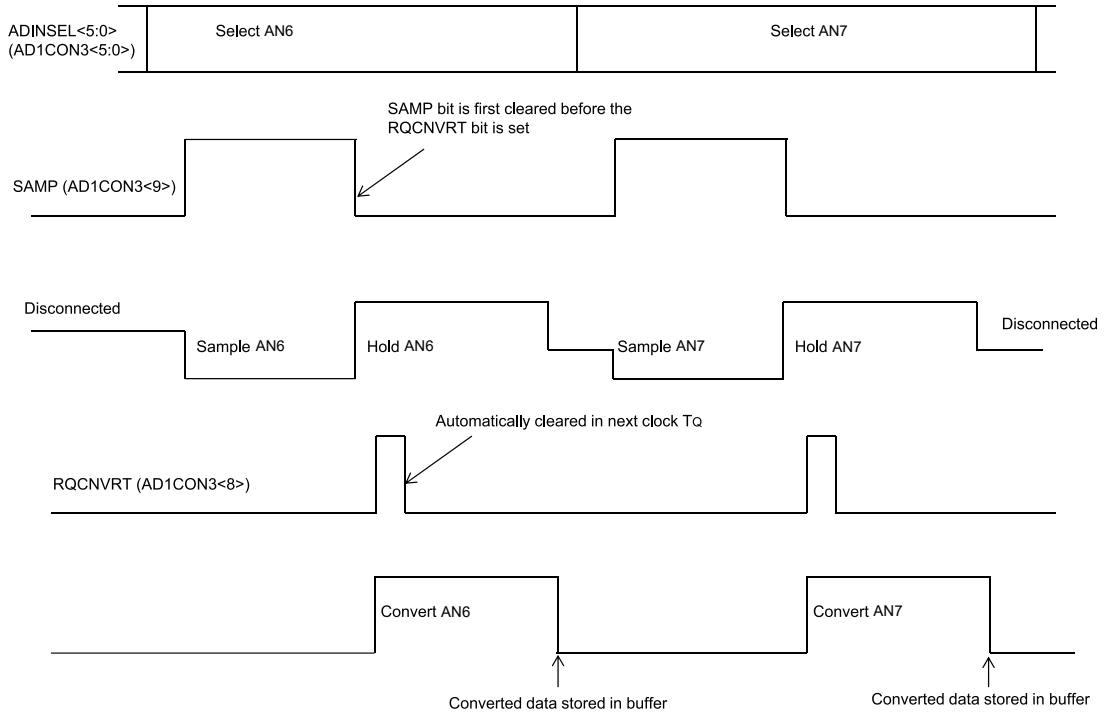
The user can explicitly request a single conversion (by software) of any selected analog input at any time during program execution without changing the trigger source configuration of the ADC.

The steps to be followed for conversion are as follows:

1. The analog input ID to be converted is specified by the ADC Input Select bits, ADINSEL[5:0] (ADCCON3[5:0]).
2. The sampling of analog input is started by setting the SAMP bit (ADCCON3[9]).
3. After the required sampling time (time delay), the SAMP bit is cleared.
4. The conversion of sampled signal is started by setting the RQCNVRT bit (ADCCON3[8]).
5. Once the conversion is complete, the ARDYx bit of the ADCDSTATx register is set. The data can be read from the ADCDATAx register.

The following figure illustrates the conversion process in graphical form.

Figure 38-6. Individual Conversion Trigger Process



38.3.5 Selecting the Voltage Reference Source

The user application can select the voltage reference for the ADC module, which can be internal or external. The Voltage Reference Input Selection bits, VREFSEL[2:0] (ADCCON3[15:13]), select the voltage reference for analog-to-digital conversions. The upper voltage reference (V_{REFH}) and the lower voltage reference (V_{REFL}) may be the internal AV_{DD} and AV_{SS} voltage rails or the band gap reference generator or the external V_{REFH}+ and V_{REF}- input pins. When the voltage reference and band gap reference are ready, the BGVRRDY (ADCCON2[31]) bit is set. If a Fault occurs in the voltage reference (such as a brown-out), the REFFLT bit (ADCCON2[30]) is set. The BGVRRDY and REFFLT bits can also generate interrupts if the BGVRIEN bit (ADCCON2[15]) and REFFLTIEN bit (ADCCON2[14]) are set, respectively.

The voltages applied to the external reference pins must comply with certain specifications. See *Electrical Characteristics* from Related Links.

The Analog Input Charge Pump Enable bit, AICPMPEN (ADCCON1[12]), must be set when the difference between the selected reference voltages (V_{REFH} - V_{REFL}) is less than 0.65 * (AV_{DD} - AV_{SS}). Setting this bit does not increase the magnitude of the reference voltage; however, setting this bit reduces the series source resistance to the sampling capacitors. This maximizes the SNR for analog-to-digital conversions using small reference voltage rails.

Related Links

43. Electrical Characteristics

38.3.6 Selecting the Scanned Inputs

All available analog inputs can be configured for scanning. Class 2 and Class 3 inputs are sampled using the shared ADC module. A single conversion trigger source is selected for all of the inputs selected for scanning using the STRGSRC[4:0] bits (ADCCON1[20:16]). On each conversion trigger, the ADC module starts converting (in the natural priority) all inputs specified in the user-specified scan list (ADCCSS1 or ADCCSS2). For Class 2 and Class 3 inputs, the trigger initiates a sequential sample/conversion process in the natural priority order.

An analog input belongs to the scan if it is:

- A Class 3 input. For Class 3 inputs, scan is the only mechanism for conversion.
- A Class 2 input that has the scan trigger selected as the trigger source by selecting the STRIG option in the TRGSRCx[4:0] bits located in the ADCTRG1 through ADCTRG8 registers.

The trigger options available for scan are identical to those available for independent triggering of Class 2 inputs. Any Class 2 inputs that are part of the scan must have the STRIG option selected as their trigger source in the TRGSRCx[4:0] bits.

Note: The end-of-scan (EOS) is generated only if the last shared input conversion has completed. Until this condition is met, the scan sequence is still in effect. Therefore, the EOS Interrupt can be used for any scan sequence with any combination of input types.

The following code is an example for ADC scanning multiple inputs.

```
int main(int argc, char** argv) {
int result[3];

/* Configure ADCCON1 */
ADCCON1 = 0;              // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
// CVD mode, Fractional mode and scan trigger source. ADCCON1bits.SELRES = 3;   // ADC7
resolution is 12 bits
ADCCON1bits.STRGSRC = 1;  // Select scan trigger.

/* Configure ADCCON2 */
ADCCON2bits.SAMC = 5;     // ADC7 sampling time = 5 * TAD7
ADCCON2bits.ADCDIV = 1;   // ADC7 clock freq is half of control clock = TAD7

/* Initialize warm up time register */ ADCANCON = 0;
ADCANCONbits.WKUPCLKCNT = 5; // Wakeup exponent = 32 * TADx

/* Clock setting */
ADCCON3bits.ADCSEL = 0;       // Select input clock source
ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
ADCCON3bits.VREFSEL = 0;      // Select AVDD and AVSS as reference source

ADC0TIMEbits.ADCDIV = 1;      // ADC0 clock frequency is half of control clock = TAD0
ADC0TIMEbits.SAMC = 5;        // ADC0 sampling time = 5 * TAD0
ADC0TIMEbits.SELRES = 3;      // ADC0 resolution is 12 bits

/* Select analog input for ADC modules, no presync trigger, not sync sampling */
ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0

/* Select ADC input mode */
ADCIMCON1bits.SIGN0 = 0;      // unsigned data format ADCIMCON1bits.DIFF0 = 0;      //
Single ended mode ADCIMCON1bits.SIGN8 = 0;      // unsigned data format ADCIMCON1bits.DIFF8
= 0;      // Single ended mode ADCIMCON1bits.SIGN40 = 0;     // unsigned data format
ADCIMCON1bits.DIFF40 = 0;     // Single ended mode

/* Configure ADCGIRQENx */
ADCGIRQEN1 = 0;               // No interrupts are used. ADCGIRQEN2 = 0;

/* Configure ADCCSSx */
ADCCSS1 = 0;                 // Clear all bits
ADCCSS2 = 0;
ADCCSS1bits.CSS0 = 1;         // AN0 (Class 1) set for scan ADCCSS1bits.CSS8 = 1;         //
AN8 (Class 2) set for scan ADCCSS2bits.CSS40 = 1;        // AN40 (Class 3) set for scan

/* Configure ADCCMPCONx */
ADCCMPCON1 = 0;               // No digital comparators are used. Setting the ADCCMPCONx
ADCCMPCON2 = 0;               // register to '0' ensures that the comparator is disabled.
ADCCMPCON3 = 0;               // Other registers are ‘don't care’.
ADCCMPCON4 = 0;
ADCCMPCON5 = 0; ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0;                // No oversampling filters are used. ADCFLTR2 = 0;
ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;
/* Set up the trigger sources */
ADCTRG1bits.TRGSRC0 = 3;      // Set AN0 (Class 1) to trigger from scan source
ADCTRG3bits.TRGSRC8 = 3;      // Set AN8 (Class 2) to trigger from scan source
// AN40 (Class 3) always uses scan trigger source

/* Early interrupt */
ADCEIEN1 = 0;                // No early interrupt
```

```
ADCEIEN2 = 0;

/* Turn the ADC on */ ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT);   // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN0 = 1;       // Enable the clock to analog bias ADC0
ADCANCONbits.ANEN7 = 1;       // Enable, ADC7

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY0);     // Wait until ADC0 is ready while(!
ADCANCONbits.WKRDY7);     // Wait until ADC7 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN0 = 1;          // Enable ADC0
ADCCON3bits.DIGEN7 = 1;          // Enable ADC7

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;

/* Wait the conversions to complete */
while (ADCDSTAT1bits.ARDY0 == 0);
/* fetch the result */
result[0] = ADCDATA0;

while (ADCDSTAT1bits.ARDY8 == 0);
/* fetch the result */
result[1] = ADCDATA8;

while (ADCDSTAT2bits.ARDY40 == 0);
/* fetch the result */
result[2] = ADCDATA40;

/*
* Process results here
*
*
*/
}
return (1);
}
```

38.3.7 Selecting the Analog-to-Digital Conversion Clock Source and Prescaler

The ADC module can use the internal Fast RC (FRC) oscillator output, system clock (SYSCLK), reference clock (REFCLK3) or peripheral bus clock (PBCLK) as the conversion clock source (T_Q). See *ADCCON3* register from Related Links.

When the ADCSEL[1:0] bits (ADCCON2[31:30]) = '01', the internal FRC oscillator is used as the ADC clock source. When using the internal FRC oscillator, the ADC module can continue to function in Sleep and Idle modes.

Note: It is recommended that applications that require precise timing of ADC acquisitions use SYSCLK as the clock source for the ADC.

For correct analog-to-digital conversions, the conversion clock limits must not be exceeded. Clock frequencies from 1 MHz to 28 MHz are supported by the ADC module.

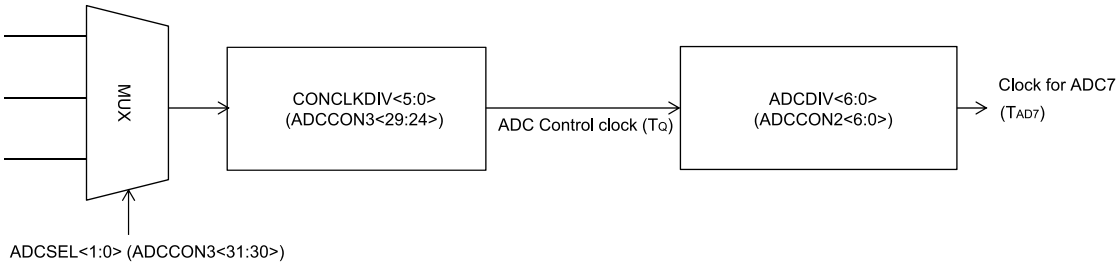
The maximum rate that analog-to-digital conversions may be completed by the ADC module (effective conversion throughput) is 2 Msps. However, the maximum rate that a single input can be converted is dependent on the sampling time requirements. In addition, the sampling time depends on the output impedance of the analog signal source. For more information on sampling time, see *ADC Sampling Requirements* from Related Links.

The input clock source for the ADC is selected using the ADCSEL[1:0] bits (ADCCON3[31:30]). The input clock is further divided by the control clock divider CONCLKDIV[5:0] bits (ADCCON3[29:24]). The output clock is called the "ADC control clock" with a time period of T_Q.

The ADC control clock is divided by the ADCDIV[6:0] bits (ADCxTIME[22:16]). This acts as the clock source for the respective dedicated ADC modules with a time period of T_{ADX}.

The ADC control clock is divided before it is used for the shared ADC by the ADCDIV[6:0] bits (ADCCON2[6:0]). The time period for this clock is denoted as T_{AD7}.

Figure 38-7. Clock Derivation for Shared ADC Modules



Equation 38-2. Sample Time for the Shared ADC Module

$t_{SAMC} = ADCCON2 < 25:16 > T_{AD}$

$t_{conversion} = 2 + ADCCON2 < 22:21 > T_{AD}$

Related Links

38.11.4. ADCCON3

38.9. ADC Sampling Requirements

38.3.8 Turning ON the ADC

Turning ON the ADC module involves the following procedure.

When the ADC module enable bit, ON (ADCCON1[15]), is set to '1', the module is in Active mode and is fully powered and functional. When the ON bit is '0', the ADC module is disabled. Once disabled, the digital and analog portions of the ADC are turned off for maximum current savings. In addition to setting the ON bit, the analog and digital circuits of ADC must be turned ON. See *Low-power Mode* from Related Links.

Note: Writing to the ADC control bits that control the ADC clock, input assignments, scanning, voltage reference selection, S&H circuit operating modes and interrupt configuration is not recommended while the ADC module is enabled.

Related Links

38.6.3. Low-Power Mode

38.3.9 ADC Status Bits

The ADC module includes the WKRDYx/WKRDY7 status bit in the ADCANCON register, which indicates the current state of ADC Analog and bias circuit. The user application must not perform any ADC operations until this bit is set.

38.4 Additional ADC Functions

This section describes some additional features of the ADC module, which includes:

- Digital comparator
- Oversampling filter

38.4.1 Digital Comparator

The ADC module features digital comparators that can be used to monitor selected analog input conversion results and generate interrupts when a conversion result is within the user-specified limits. Conversion triggers are still required to initiate conversions. The comparison occurs automatically once the conversion is complete. This feature is enabled by setting the Digital Comparator Module Enable bit, ENDCMP (ADCCMPCONx[7]).

The user application makes use of an interrupt that is generated when the analog-to-digital conversion result is higher or lower than the specified high and low limit values in the ADCCMPx register. The high and low limit values are specified in the DCMPHI[15:0] bits (ADCCMPx[31:16]) and the DCMPLO[15:0] bits (ADCCMPx[15:0]).

The CMPEx bits ('x' = 0 through 31) in the ADCCMPENx registers are used to specify which analog inputs are monitored by the digital comparator (for the first 8 analog inputs, ANx, where 'x' = 0 through 31). The ADCCMPCONx register specifies the comparison conditions that generates an interrupt, as follows:

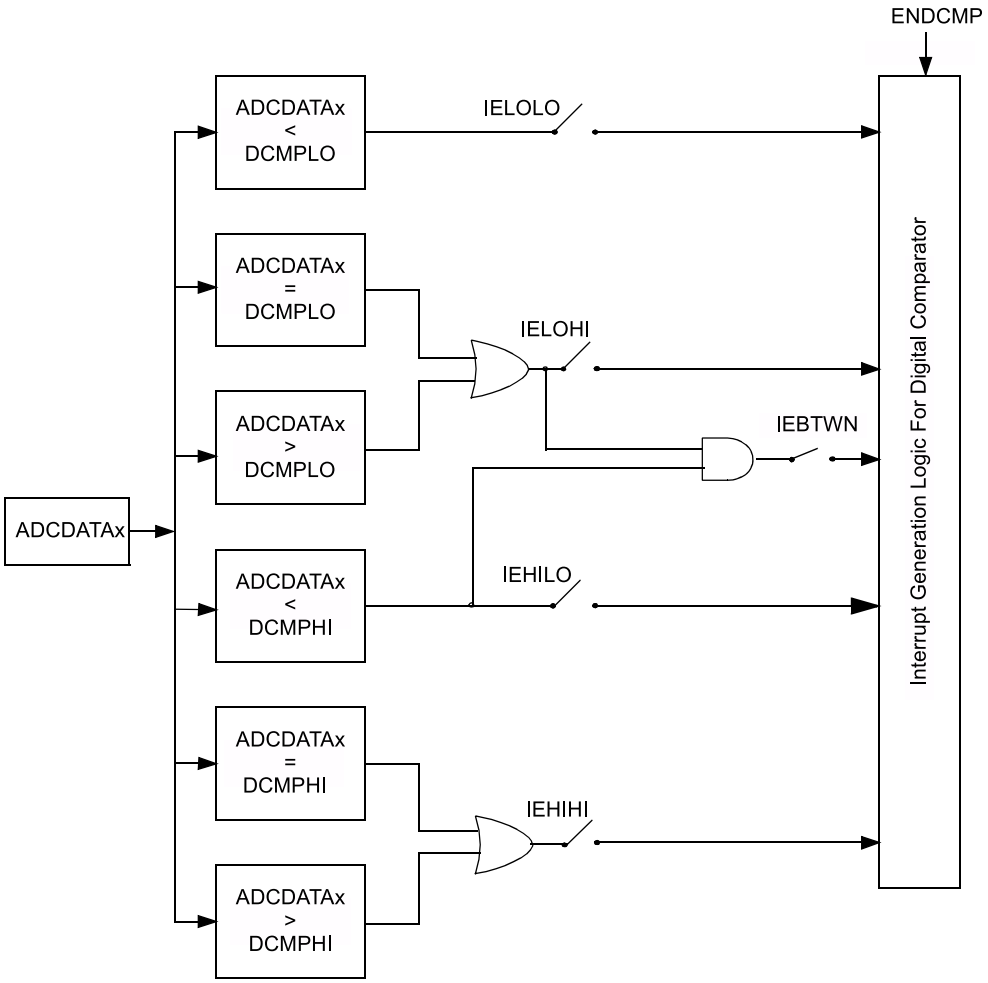
- When IEBTWN = 1, an interrupt is generated when DCMPLO ≤ ADCDATA < DCMPHI
- When IEHIHI = 1, an interrupt is generated when DCMPHI ≤ ADCDATA
- When IEHILO = 1, an interrupt is generated when ADCDATA < DCMPHI
- When IELOHI = 1, an interrupt is generated when DCMPLO ≤ ADCDATA
- When IELOLO = 1, an interrupt is generated when ADCDATA < DCMPLO

The comparator event generation is illustrated in the following figure. When the ADC module generates a conversion result, the conversion result is provided to the comparator. The comparator uses the DIFFx and SIGNx bits of the ADCIMCONx register (depending on the analog input used) to determine the data format used and to appropriately select whether the comparison must be signed or unsigned. The global ADC setting, which is specified by the FRACT bit (ADCCON1[23]), is also used to set the fractional or integer format. The digital comparator compares the ADC result with the high and low limit values (depending on the selected comparison criteria) in the ADCCMPx register.

Depending on the comparator results, a digital comparator interrupt event may be generated. If a comparator event occurs, the Digital Comparator Interrupt Event Detected status bit, DCMPED (ADCCMPCONx[5]), is set, and the Analog Input Identification (ID) bits, AINID[4:0] (ADCCMPCONx[12:8]), are automatically updated so that the user application knows which analog input generated the interrupt event.

Note: The user software must format the values contained in the ADCCMPx registers to match converted data format as either signed or unsigned, and fractional or integer.

Figure 38-8. Digital Comparator



The following code is an example for ADC digital comparator.

```
int main(int argc, char** argv) {
int result = 0, eventFlag = 0;

/* Configure ADCCON1 */
ADCCON1 = 0;                 // No ADCCON1 features are enabled including: Stop-in-Idle,
// turbo, CVD mode, Fractional mode and scan trigger source. ADCCON1bits.SELRES = 3;       //
ADC resolution is 12 bits
ADCCON1bits.STRGSRC = 0;      // No scan trigger.

/* Configure ADCCON2 */
ADCCON2bits.SAMC = 5;         // ADC7 sampling time = 5 * TAD7
ADCCON2bits.ADCDIV = 1;       // ADC7 clock freq = TAD7

/* Initialize warm up time register */ ADCANCON = 0;









/* No selection for dedicated ADC modules, no presync trigger, not sync sampling */
ADCTRGMODEbits = 0;

/* Select ADC input mode */
ADCIMCON1bits.SIGN8 = 0;      // unsigned data format
ADCIMCON1bits.DIFF8 = 0;      // Single ended mode

/* Configure ADCGIRQENx */
ADCGIRQEN1 = 0;               // No interrupts are used
ADCGIRQEN2 = 0;

/* Configure ADCCSSx */
ADCCSS1 = 0;                 // No scanning is used
ADCCSS2 = 0;

/* Configure ADCCMPCONx */
ADCCMP1 = 0;                 // Clear the register ADCCMP1bits.DCMPHI = 0xC00;  // High
limit is a 3072 result. ADCCMP1bits.DCMPLO = 0x500;  // Low limit is a 1280 result.
ADCCMPCON1bits.IEBTWN = 1;    // Create an event when the measured result is
// >= low limits and < high limit. ADCCMPEN1 = 0;                // Clear all enable bits
ADCCMPEN1bits.CMPE8 = 1;      // set the bit corresponding to AN8
ADCCMPCON1bits.ENDCMP = 1;    // enable comparator
ADCCMPCON2 = 0; ADCCMPCON3 = 0; ADCCMPCON4 = 0; ADCCMPCON5 = 0; ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0;                // No oversampling filters are used. ADCFLTR2 = 0;
ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;
/* Set up the trigger sources */
ADCTRG3bits.TRGSRC8 = 3;         // Set AN8 (Class 2) to trigger from scan source

/* Early interrupt */
ADCEIEN1 = 0;                // No early interrupt
ADCEIEN2 = 0;

/* Turn the ADC on */ ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY);     // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT);       // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN7 = 1;          // Enable the clock to analog bias

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY7);     // Wait until ADC7 is ready

/* Enable the ADC module */
ADCCON3bits.DIGEN7 = 1;          // Enable ADC7

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;
```

```
while (ADCDSTAT1bits.ARDY8 == 0);
/* fetch the result */
result = ADCDATA8;

/* Note: It is not necessary to fetch the result for the digital
* comparator to work. In this example we are triggering from
* software so we are using the ARDY8 to gate our loop. Reading the
* data clears the ARDY bit.
*/
/* See if we have a comparator event*/
if (ADCCMPCON1bits.DCMPED == 1) {
eventFlag = 1;
/*
* Process results here
*/
}
}
return (1);
}
```

38.4.2 Oversampling Digital Filter

The ADC module supports two oversampling digital filters. The oversampling digital filter consists of an accumulator and a decimator (down-sampler), which function together as a low-pass filter. By sampling an analog input at a higher-than-required sample rate, then processing the data through the oversampling digital filter, the effective resolution of the ADC module can be increased at the expense of decreased conversion throughput.

To obtain 'x' bits of extra resolution, the number of samples required (over and above the Nyquist rate) = $(2^x)^2$:

- 4x oversampling yields one extra bit of resolution (total 13 bits resolution)
- 16x oversampling yields two extra bits of resolution (total 14 bits resolution)
- 64x oversampling provides three extra bits of resolution (total 15 bits resolution)
- 256x oversampling provides four extra bits of resolution (total 16 bits resolution)

The digital filter also has an averaging mode, where it accumulates the samples and divides it by the number of samples.

Note:

1. Only Class 2 analog inputs can engage the digital filter. Therefore, the CHNLID[2:0] bits are 3 bits wide (0 to 7).
2. During the burst conversion process (repeated trigger until all required data for oversampling is obtained), in the case of filtering Class 2 input using the shared ADC module, higher priority ADC inputs may still process conversions; lower priority ADC conversion requests are held waiting until the filter burst sequence is completed.
3. If higher priority requests occur during the digital filter sequence, they delay the completion of the filtering process. This delay may affect the accuracy of the result because the multiple samples cannot be contiguous. The user must arrange the initiation trigger for the oversampling filters to occur while there are no expected interruptions from higher priority ADC conversion requests.

The user application must configure the following bits to perform an oversampling conversion:

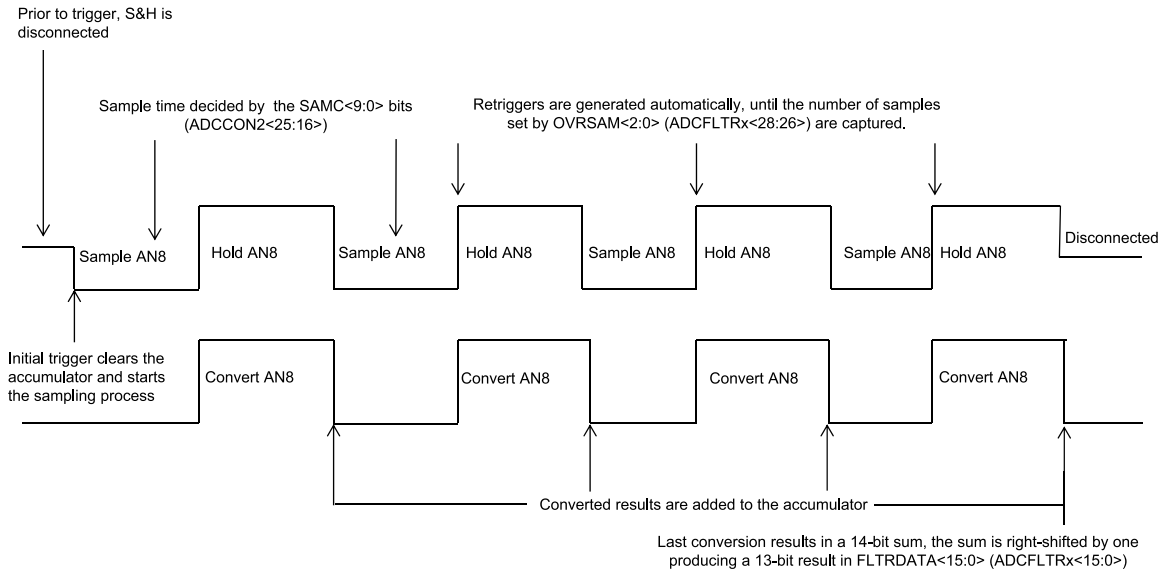
- Select the amount of oversampling through the Oversampling Filter Oversampling Ratio (OVRSAM[2:0]) bits in the ADC Filter register (ADCFLTRx[28:26]).
- Set the filter mode to either Oversampling mode or Averaging mode using the DFMODE bit(ADCFLTRx[29]).
- If the filter is set to Averaging mode and the data format is set to fractional (FRACT bit), set or clear the DATA16EN bit (ADCFLTRx[30]) to set the output resolution.
- Set the sample time for subsequent samples:
  - If using Class 2 inputs, select the sample time using the SAMC[9:0] bits (ADC- CON2[25:16]).
- Select the specific analog input to be oversampled by configuring the Analog Input ID Selection bits, CHNLID[4:0] (ADCFLTRx[20:16]).
- If needed, include the oversampling filter interrupt event in the global ADC interrupt by setting the Accumulator Filter Global Interrupt Enable bit, AFGIEN (ADCFLTRx[25]).
- Enable the oversampling filter by setting the Oversampling Filter Accumulator Enable bit, AFEN (ADCFLTRx[31]).

When the digital filter module is configured, the filter's control logic waits for an external trigger to initiate the process. The trigger signal for the analog input to be oversampled causes the accumulator to be cleared and initiates the first conversion. The trigger also forces the trigger sensitivity into level mode and forces the trigger itself to 1 as long as the filter needs to acquire the user-specified number of samples via the OVRSAM[2:0] bits (ADCFLTRx[28:26]). The time delay between each acquired sample is decided by the set sample time in the SAMC[9:0] bits in the ADCCON2 register for Class 2 and the time for conversion. When the required number set by OVRSAM[2:0] are received and processed, the data stored in the FLTRDATA[15:0] bit (ADCFLTRx[15:0]) and the AFRDY bit (ADCFLTRx[24]) is set and the interrupt is generated (if enabled).

The following figure illustrates 4x oversampling using a Class 2 input. Triggering a Class 2 input initiates sampling for the length of time defined by the SAMC[9:0] bits. Retriggers generated by the oversampling logic use the SAMC[9:0] bits to set the sample time.

Class 2 inputs use the shared S&H; therefore, oversampling blocks lower priority Class 2 and Class 3 triggers. Higher priority Class 2 triggers completely disrupt the oversampling process; therefore, they must be avoided completely. The same priority rule applies to two Class 2 inputs that use two digital filters. In such a case, the higher priority input also uses the shared ADC module in Burst mode and prevents the lower priority input from using the shared ADC. Only after all required samples are obtained by the higher priority input can the lower priority input use the shared ADC to acquire samples for its own digital filtering.

Figure 38-9. 4x Oversampling of a Class 2 Input

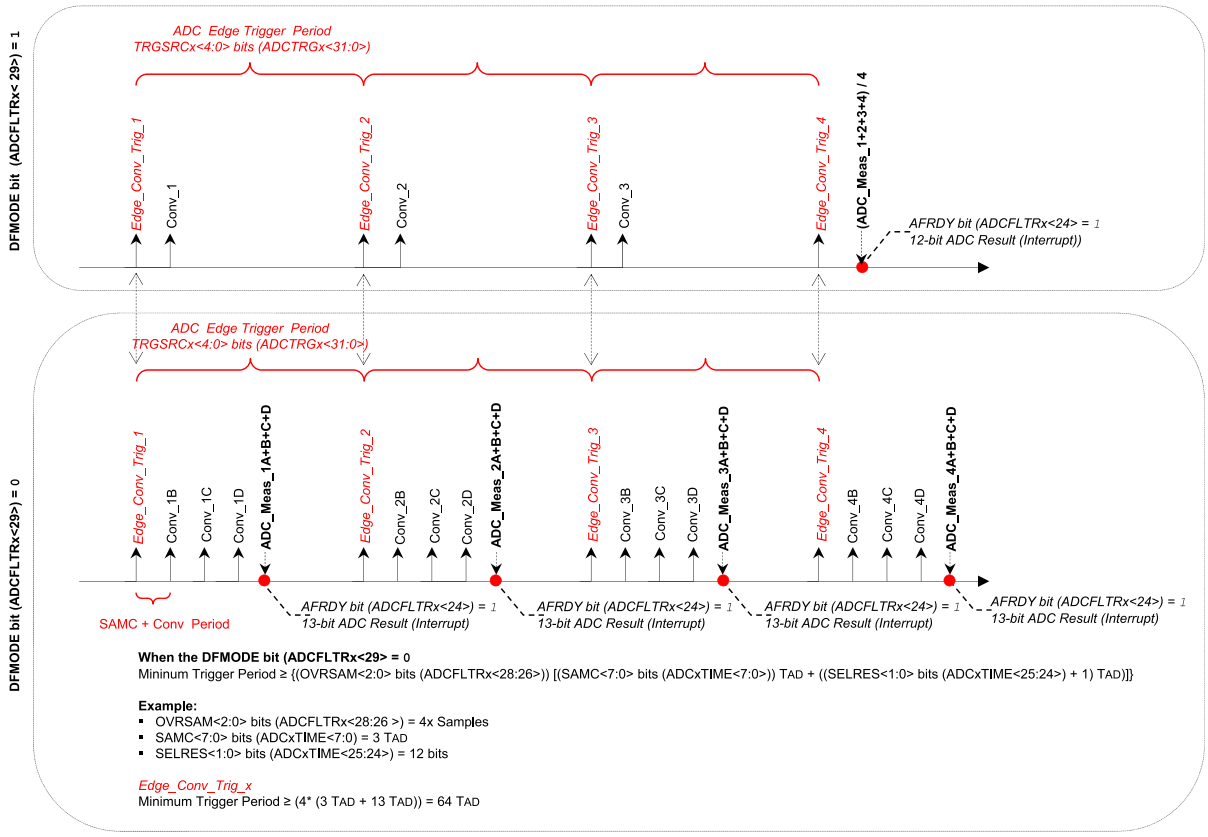


The following code is an example for ADC digital oversampling filter.

```
int main(int argc, char** argv) {
int result;

/* Configure ADCCON1 */
ADCCON1 = 0;          // No ADCCON1 features are enabled including: Stop-in-Idle, turbo,
// CVD mode, Fractional mode and scan trigger source.

/* Configure ADCCON2 */
ADCCON2 = 0;          // Since, we are using only the Class 1 inputs, no setting is
// required for ADCDIV

/* Initialize warm up time register */ ADCANCON = 0;
ADCANCONbits.WKUPCLKCNT = 5;  // Wake-up exponent = 32 * TADx

/* Clock setting */ ADCCON3 = 0;
ADCCON3bits.ADCSEL = 0;       // Select input clock source
ADCCON3bits.CONCLKDIV = 1;    // Control clock frequency is half of input clock
ADCCON3bits.VREFSEL = 0;      // Select AVDD and AVSS as reference source
```

```
ADC0TIMEbits.ADCDIV = 1;      // ADC0 clock frequency is half of control clock = TAD0
ADC0TIMEbits.SAMC = 5;        // ADC0 sampling time = 5 * TAD0
ADC0TIMEbits.SELRES = 3;      // ADC0 resolution is 12 bits

/* Select analog input for ADC modules, no presync trigger, not sync sampling */
ADCTRGMODEbits.SH0ALT = 0;    // ADC0 = AN0

/* Select ADC input mode */
ADCIMCON1bits.SIGN0 = 0;      // unsigned data format
ADCIMCON1bits.DIFF0 = 0;      // Single ended mode

/* Configure ADCGIRQENx */
ADCGIRQEN1 = 0;               // No interrupts are used
ADCGIRQEN2 = 0;

/* Configure ADCCSSx */
ADCCSS1 = 0;                  // No scanning is used
ADCCSS2 = 0;
/* Configure ADCCMPCONx */
ADCCMPCON1 = 0;               // No digital comparators are used. Setting the ADCCMPCONx
ADCCMPCON2 = 0;               // register to '0' ensures that the comparator is disabled.
ADCCMPCON3 = 0;               // Other registers are ‘don't care’.
ADCCMPCON4 = 0; ADCCMPCON5 = 0; ADCCMPCON6 = 0;

/* Configure ADCFLTRx */
ADCFLTR1 = 0;                 // Clear all bits ADCFLTR1bits.CHNLID = 0;      // Use AN0 as
the source ADCFLTR1bits.OVRSAM = 3;      // 16x oversampling ADCFLTR1bits.DFMODE = 0;      //
Oversampling mode ADCFLTR1bits.AFEN = 1;        // Enable filter 1
ADCFLTR2 = 0;                 // Clear all bits
ADCFLTR3 = 0; ADCFLTR4 = 0; ADCFLTR5 = 0; ADCFLTR6 = 0;

/* Set up the trigger sources */ ADCTGSNSbits.LVL0 = 0;        // Edge trigger
ADCTRG1bits.TRGSRC0 = 1;     // Set AN0 to trigger from software.

/* Turn the ADC on */ ADCCON1bits.ON = 1;

/* Wait for voltage reference to be stable */
while(!ADCCON2bits.BGVRRDY); // Wait until the reference voltage is ready
while(ADCCON2bits.REFFLT);   // Wait if there is a fault with the reference voltage

/* Enable clock to analog circuit */
ADCANCONbits.ANEN0 = 1;       // Enable the clock to analog bias and digital control

/* Wait for ADC to be ready */
while(!ADCANCONbits.WKRDY0); // Wait until ADC0 is ready

/* Enable the ADC module */ ADCCON3bits.DIGEN0 = 1;       // Enable ADC0

while (1) {
/* Trigger a conversion */ ADCCON3bits.GSWTRG = 1;

/* Wait for the oversampling process to complete */
while (ADCFLTR1bits.AFRDY == 0);
/* fetch the result */
result = ADCFLTR1bits.FLTRDATA;

/*
* Process result Here
*
* Note 1: Loop time determines the sampling time for the first sample.
* remaining samples sample time is determined by set sampling + conversion time.
*
* Note 2: The first 5 samples may have reduced accuracy.
*
*/
}
return (1);
}
```

Figure 38-10. ADC Filter Comparisons Example



38.5 Interrupts

The ADC module supports interrupts triggered from a variety of sources that can be processed individually or globally. An early interrupt feature is also available to compensate for interrupt servicing latency.

After an enabled interrupt is generated, the CPU jumps to the vector assigned to that interrupt. The CPU begins executing code at the vector address. The user software at this vector address must perform the required operations, such as processing the data results, clearing the interrupt flag, then exiting. See *Nested Vector Interrupt Controller (NVIC)* from Related Links for more information on interrupts and the vector address table details.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

38.5.1 Interrupt Sources

The ADC is capable of generating interrupts from the events listed in the following table.

Table 38-4. ADC Interrupt Sources

Interrupt Event	Description	Interrupt Enable Bit	Interrupt Status Bit
ANx Data Ready Event	Interrupt is generated upon a completion of a conversion from an analog input source (ANx). Each of the ARDYx bits is capable of generating a unique interrupt when set using the ADCBASE register.	AGIENx of ADCGIRQEN1	ARDYx of ADCDSTAT1 register

.....continued			
Interrupt Event	**Description**	**Interrupt Enable Bit**	**Interrupt Status Bit**
Digital Comparator Event	When an conversion's comparison criteria are met by a configured and enabled digital comparator. Each of the digital comparators is capable of generating a unique interrupt when its DCMPED bit is set.	DCMPGIEN of ADCCMPCONx register	DCMPED of ADCCMPCONx register
Oversampling Filter Data Ready Event	When an oversampling filter has completed the accumulation/decimation process and has stored the result.	AFGIEN of ADCFLTRx register	AFRDY of ADCFLTRx register
Both Band Gap Voltage and ADC Reference Voltage Ready Event	Interrupt is generated when both band gap voltage and ADC reference voltage are ready.	BGVRIEN of ADCCON2 register	BGVRRDY of ADCCON2 register
Band Gap Fault/ Reference Voltage Fault/AV_{DD} Brown-out Fault Event	Interrupt is generated when Band Gap Fault/Reference Voltage Fault/AV_{DD} Brown-out occurs.	REFFLTIEN of ADCCON2 register	REFFLT of ADCCON2 register
ADC Module Wake-up Event	Interrupt is generated when ADC wakes up after being enabled.	WKIEN0 of ADCANCON register	WKRDY0 of ADCANCON register
Update Ready Event	Interrupt is generated when ADC SFRs are ready to be (and can be safely) updated with new values.	UPDIEN of ADCCON3 register	UPDRDY of ADCCON3 register

38.5.2 ADC Base Register (ADCBASE) Usage

After conversion of ADC is complete, if the interrupt is vectored to a function that is common to all analog inputs, it takes some significant time to find the ADC input by evaluating the ARDYx bits in the ADCDSTATx. To avoid this time spent, the ADCBASE register is provided, which contains the base address of the user's ADC ISR jump table. When read, the ADCBASE register provides a sum of the contents of the ADCBASE register plus an encoding of the ARDYx bits set in the ADCDSTATx registers. This use of the ADCBASE register supports the creation of an interrupt vector address that can be used to improve the performance of an ISR.

The ARDYx bits are binary priority encoded with ARDY1 being the highest priority and ARDY8 being the lowest priority. The encoded priority result is, then, shifted left the amount specified by the number of bit positions specified by the IRQVS[2:0] bits in the ADCCON1 register, then added to the contents of the ADCBASE register. If there are no ARDYx bits set, then reading the ADCBASE register equals the value written into the ADCBASE register.

The ADCBASE register is typically loaded with the base address of a jump table that contains the address of the appropriate ISR. The k^{th} interrupt request is enabled via the AGIENx bit (1-8) in one of ADCGIRQENx SFRs ('x' = 1 or 2).

The following codes are examples for the ADCBASE register usage.

Case 1:

```
ADCBASE = 0x1234; // Set the address
ADCCON1bits.IRQVS = 2; // left shift by 2
ADCGIRQEN1bits.AGIEN0 = 1; // enable interrupt when AN0 completion is done.
```

When the ADC conversion for AN0 is complete, bit `0` of ADCDSTAT1 = ARDY0 is set.

Read value of ADCBASE = 0x1234 + (0 << 2) = 0x1234.

Therefore, the ISR must be placed at address `0x1234` for AN0.

Case 2:

```
ADCBASE = 0x1234; // Set the address
ADCCON1bits.IRQVS = 2; // left shift by 2
ADCGIRQEN1bits.AGIEN0 = 2; // enable interrupt when AN2 completion is done.
```

When the ADC conversion for AN2 is complete, bit 2 of ADCDSTAT1 = ARDY2 is set.

Read value of ADCBASE = 0x1234 + (2 << 2) = 0x123C.

Therefore, the ISR must be placed at address 0x123C for AN2.

Note: The contents of the ADCBASE register are not altered. Summation is performed when the ADCBASE register is read and the summation result is the returned read value from the ADCBASE SFR.

38.5.3 Interrupt Enabling, Priority and Vectoring

Each of the ADC events previously mentioned generates an interrupt when its associate Interrupt Enable bit, IE, is set. Each of the ADC events previously listed also has an associated interrupt vector. See *Nested Vector Interrupt Controller (NVIC)* from Related Links for more information on the vector location and control/status bits associated with each individual interrupt.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

38.5.4 Individual and Global Interrupts

The use of the individual interrupts previously listed can significantly optimize the servicing of multiple ADC events by keeping each ISR focused on efficiently handling a specific event. In addition, different ISRs can be easily segregated according to the tasks performed, thereby making user software easier to implement and maintain. There may be cases where it is desirable to have a single ISR service multiple interrupt events. To facilitate this, each ADC event can be logically “ORed” to create a single global ADC interrupt. When an ADC event is enabled for a global interrupt, it vectors to a single interrupt routine. It is the responsibility of this single global ISR to determine the source of the interrupt through polling and process it accordingly.

Use of the Global Interrupt requires configuration of its own unique IE, IF, IP and IS bits as well as configuration of its interrupt vector as described in Interrupt Enabling, Priority and Vectoring. See *Interrupt Enabling, Priority and Vectoring* from Related Links.

Interrupts for the ADC can be configured as individual or global, or utilized as both where some are processed individually and others in the global ISR.

Related Links

38.5.3. Interrupt Enabling, Priority and Vectoring

38.6 Power-Saving Modes of Operation

The Power-Saving, Sleep and Idle modes are useful for reducing the conversion noise by minimizing the digital activity of the CPU, buses and other peripherals.

38.6.1 Sleep Mode

When the device enters Sleep mode, the system clock (SYCCLK) is halted. If an ADC module selects SYSCLK as its clock source or selects REFCLK3 as its clock source (REFCLK3 is generated from SYSCLK), the ADC enters the Sleep mode.

When the SYSCLK is the source (directly or indirectly) and Sleep mode occurs during a conversion, the conversion is aborted. The converter cannot resume a partially completed conversion on exiting from Sleep mode. The ADC register contents are not affected by the device entering or leaving Sleep mode. The ADC module can operate during Sleep mode if the ADC clock source is derived from a source other than SYSCLK that is active during Sleep mode. The FRC clock source is a logical choice for operation during Sleep; however, the REFCLK3 clock source can also be used, provided it has an input clock that is operational during Sleep mode.

ADC operation during Sleep mode reduces the digital switching noise from the conversion. When the conversion is completed, the ARDYx status bit for that analog input is set and the result is loaded into the corresponding ADC Result register (ADCDATAx).

If any of the ADC interrupts are enabled, the device is woken up from Sleep mode when the ADC interrupt occurs. The program execution resumes at the ADC ISR if the ADC interrupt is greater than the current CPU priority. Otherwise, execution continues from the instruction after the WAIT instruction that placed the device in Sleep mode.

To minimize the effects of digital noise on the ADC module operation, the user must select a conversion trigger source that ensures that the analog-to-digital conversion take places in Sleep mode. For example, the external interrupt pin (INT0) conversion trigger option (TRGSRC[4:0] = `00100`) can be used for performing sampling and conversion while the device is in Sleep mode.

Note: For the ADC module to operate in Sleep mode, the ADC clock source must be set to Internal FRC (ADCSEL[1:0] bits (ADCCON2[31:30]) = `01`). Alternately, the REFCLK3 source can be used; however, the clock source used for REFCLK3 must operate during Sleep mode. Any changes to the ADC clock configuration require that the ADC be disabled.

38.6.2 Operation During Idle Mode

For the ADC, the stop in Idle Mode bit, SIDL (ADCCON1[13]), specifies whether the ADC module stops on Idle or continues on Idle. If SIDL = `0`, the ADC module continues normal operation when the device enters Idle mode. If any of the ADC interrupts are enabled, the device wakes up from Idle mode when the ADC interrupt occurs. The program execution resumes at the ADC ISR if the ADC interrupt is greater than the current CPU priority. Otherwise, execution continues from the instruction after the WAIT instruction that placed the device in Idle mode.

If SIDL = `1`, the ADC module stops in Idle mode. If the device enters Idle mode during a conversion, the conversion is aborted. The converter cannot resume a partially completed conversion on exiting from Idle mode.

38.6.3 Low-Power Mode

The ADC module can be placed in a low-power state by disabling the digital circuit for individual ADC modules that are not running. This is possible by clearing the DIGENx bits and the DIGEN7 bit in the ADCCON3 register. (See *ADCCON3* register from Related Links.)

An even lower power state is possible by disabling the analog and bias circuit for individual ADC modules that are not running. This is possible by clearing the ANENx bits and the ANEN7 bit in the ADCANCON register. (See *ADCANCON* register from Related Links.) Disabling the digital circuit to achieve Low-Power mode provides a significantly faster module restart compared to disabling and re-enabling the analog and bias circuit of the ADC module. This is because disabling and re-enabling the analog and bias circuit using the ANENx bits and the ANEN7 bit requires a wake-up time (typical minimum wake-up time of 20 µs) for the ADC module before it can be used. Refer to the Electrical Specifications in the specific device data sheet for more information on the stabilization time.

When the analog and bias circuit for an ADC module is enabled, the wake-up must be polled (or through an interrupt) using the wake-up ready bits, WKRDY6:WKRDY0 and WKRDY7, which must be equal to '`1`'.

Related Links

38.11.4. ADCCON3
38.11.24. ADCANCON
43. Electrical Characteristics

38.7 Effects of Reset

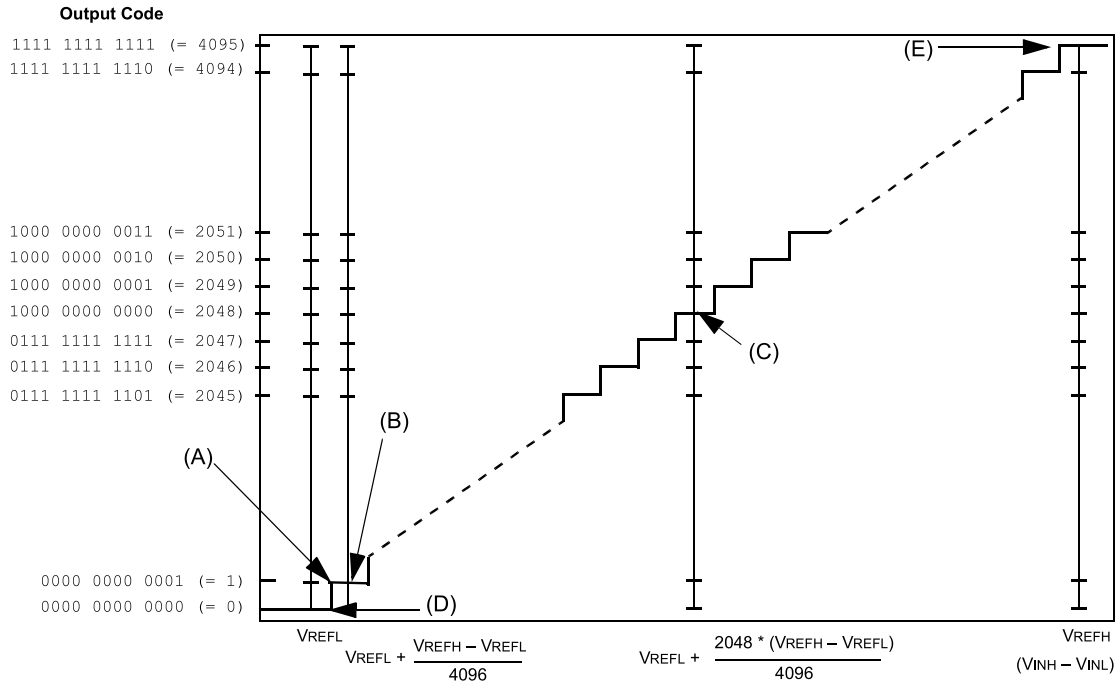
Following any Reset event, all the ADC control and status registers are reset to their default values with control bits in a non-active state. This disables the ADC module and sets the analog input pins to Analog Input mode. Any conversion that was in progress terminates, and the result cannot be written to the result buffer. The values in the ADCDATAx registers are initialized to `0x00000000` during a device Reset. The bias circuits are also turned off, so the ADC resuming operations wait for the bias circuits to stabilize by polling (or requesting to be interrupted by) the BGVRRDY bit (ADCCON2 register).

38.8 Transfer Function

A typical transfer function of the 12-bit ADC is illustrated in the following figure. The difference of the input voltages (V_{INH} - V_{INL}) is compared with the reference (V_{REFH} - V_{REFL}).

- The first code transition (A) occurs when the input voltage is (V_{REFH} - V_{REFL}/8192) or 0.5 LSb.
- The `0000 0000 0001` code is centered at (V_{REFH} - V_{REFL}/4096) or 1.0 LSb (B).
- The `1000 0000 0000` code is centered at (2048 * (V_{REFH} - V_{REFL})/4096) (C).
- An input voltage less than (1 * (V_{REFH} - V_{REFL})/8192) converts as `0000 0000 0000` (D).
- An input greater than (8192 * (V_{REFH} - V_{REFL})/8192) converts as `1111 1111 1111` (E).

Figure 38-11. Analog-to-Digital Transfer Function



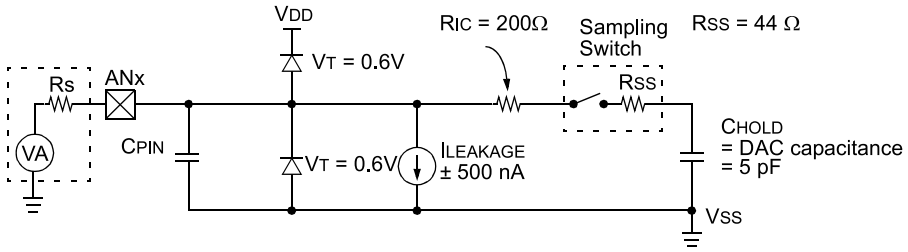
38.9 ADC Sampling Requirements

The analog input model of the 12-bit ADC is illustrated in the following figure. The total acquisition time for the analog-to-digital conversion is a function of the internal circuit settling time and the holding capacitor charge time.

For the ADC module to meet its specified accuracy, the charge holding capacitor (C_{HOLD}) must be allowed to fully charge to the voltage level on the analog input pin. The analog output source impedance (R_S), the interconnect impedance (R_{IC}) and the internal sampling switch (R_{SS}) impedance combine to directly affect the time required to charge the C_{HOLD}. The combined impedance of the analog sources must, therefore, be small enough to fully charge (to within one-fourth LSB of the desired voltage) the holding capacitor within the selected sample time. The internal holding capacitor is in the discharged state prior to each sample operation.

At least 1 T_{AD} time period must be allowed between conversions for the acquisition time. Refer to the *Electrical Characteristics* from the Related Links.

Figure 38-12. 12-bit ADC Analog Input Model



Note: The C_{PIN} value depends on the device package and is not tested. The effect of the C_{PIN} is negligible if Rs 5 k.

Legend:

- C_{PIN} = Input capacitance
- R_{SS} = Sampling switch resistance
- R_S = Source resistance
- $I_{LEAKAGE}$ = Leakage current at the pin due to various junctions
- V_T = Threshold voltage
- R_{IC} = Interconnect resistance
- C_{HOLD} = Sample/hold capacitance

Related Links

43. Electrical Characteristics

38.10 Connection Considerations

Because the analog inputs employ Electrostatic Discharge (ESD) protection, they have diodes to V_{DD} and V_{SS}; therefore, the analog input must be between V_{DD} and V_{SS}. If the input voltage exceeds this range by greater than 0.3V (either direction), one of the diodes becomes forward biased, and it may damage the device if the input current specification is exceeded.

An external RC filter is sometimes added for antialiasing of the input signal. The R (resistive) component must be selected to ensure that the acquisition time is met. Any external components connected (through high-impedance) to an analog input pin (capacitor, Zener diode and so on) must have very little leakage current at the pin.

38.11 Register Description

Notes: The following conventions are used in the following registers:

- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- 1= Bit is set0= Bit is cleared
- x = Bit is unknown
- -n = Value at POR
- HS = Hardware Set
- HC = Hardware Cleared

Note: CLR/SET/INV registers for each register are located at offset <register offset> + 0x04, 0x08, 0x0C, respectively.

38.11.1 Register Summary

The PIC32CX-BZ2 12-bit High Speed SAR ADC module has the following Special Function Registers (SFRs):

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00 ... 0x13FF	Reserved									
0x1400	ADCCON1	7:0		IRQVS[2:0]			STRGLVL	DMABL[2:0]		
		15:8	ON	FRZ	SIDL			FSYDMA	FSYUPB	SCANEN
		23:16	FRACT	SELRES[1:0]		STRGSRC[4:0]				
		31:24								
0x1404 ... 0x140F	Reserved									
0x1410	ADCCON2	7:0		ADCDIV[6:0]						
		15:8	BGVRIEN	REFFLTIEN	EOSIEN		ENXCNVRT			
		23:16	SAMC[7:0]							
		31:24	BGVRRDY	REFFLT	EOSRDY				SAMC[9:8]	
0x1414 ... 0x141F	Reserved									
0x1420	ADCCON3	7:0	GLSWTRG	GSWTRG	ADINSEL[5:0]					
		15:8	VREFSEL[2:0]			TRGSUSP	UPDIEN	UPDRDY	SAMP	RQCNVRT
		23:16	CHN_EN_SHR							
		31:24	ADCSEL[1:0]		CONCLKDIV[5:0]					
0x1424 ... 0x143F	Reserved									
0x1440	ADCIMCON1	7:0	DIFF3	SIGN3	DIFF2	SIGN2	DIFF1	SIGN1	DIFF0	SIGN0
		15:8	DIFF7	SIGN7	DIFF6	SIGN6	DIFF5	SIGN5	DIFF4	SIGN4
		23:16	DIFF11	SIGN11	DIFF10	SIGN10	DIFF9	SIGN9	DIFF8	SIGN8
		31:24								
0x1444 ... 0x147F	Reserved									
0x1480	ADCGIRQEN1	7:0	AGIEN7	AGIEN6	AGIEN5	AGIEN4	AGIEN3	AGIEN2	AGIEN1	AGIEN0
		15:8					AGIEN11	AGIEN10	AGIEN9	AGIEN8
		23:16								
		31:24								
0x1484 ... 0x149F	Reserved									
0x14A0	ADCCSS1	7:0	CSS7	CSS6	CSS5	CSS4	CSS3	CSS2	CSS1	CSS0
		15:8					CSS11	CSS10	CSS9	CSS8
		23:16								
		31:24								
0x14A4 ... 0x14BF	Reserved									
0x14C0	ADCDSTAT1	7:0	ARDY7	ARDY6	ARDY5	ARDY4	ARDY3	ARDY2	ARDY1	ARDY0
		15:8					ARDY11	ARDY10	ARDY9	ARDY8
		23:16								
		31:24								
0x14C4 ... 0x14DF	Reserved									
0x14E0	ADCCMPEN1	7:0	CMPEx[7:0]							
		15:8								
		23:16								
		31:24								

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x14E4 ... 0x14EF	Reserved									
0x14F0	ADCCMP1	7:0	DCMPLO[7:0]							
		15:8	DCMPLO[15:8]							
		23:16	DCMPHI[7:0]							
		31:24	DCMPHI[15:8]							
0x14F4 ... 0x14FF	Reserved									
0x1500	ADCCMPEN2	7:0	CMPEx[7:0]							
		15:8								
		23:16								
		31:24								
0x1504 ... 0x150F	Reserved									
0x1510	ADCCMP2	7:0	DCMPLO[7:0]							
		15:8	DCMPLO[15:8]							
		23:16	DCMPHI[7:0]							
		31:24	DCMPHI[15:8]							
0x1514 ... 0x159F	Reserved									
0x15A0	ADCFLTR1	7:0	FLTRDATA[7:0]							
		15:8	FLTRDATA[15:8]							
		23:16				CHNLID[4:0]				
		31:24	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
0x15A4 ... 0x15AF	Reserved									
0x15B0	ADCFLTR2	7:0	FLTRDATA[7:0]							
		15:8	FLTRDATA[15:8]							
		23:16				CHNLID[4:0]				
		31:24	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
0x15B4 ... 0x15FF	Reserved									
0x1600	ADCTRG1	7:0				TRGSRC0[4:0]				
		15:8				TRGSRC1[4:0]				
		23:16				TRGSRC2[4:0]				
		31:24				TRGSRC3[4:0]				
0x1604 ... 0x160F	Reserved									
0x1610	ADCTRG2	7:0				TRGSRC4[4:0]				
		15:8				TRGSRC5[4:0]				
		23:16				TRGSRC6[4:0]				
		31:24				TRGSRC7[4:0]				
0x1614 ... 0x167F	Reserved									
0x1680	ADCCMPCON1	7:0	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
		15:8			AINID[5:0]					
		23:16								
		31:24								
0x1684 ... 0x168F	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1690	ADCCMPCON2	7:0	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
		15:8				AINID[4:0]				
		23:16								
		31:24								
0x1694 ... 0x16FF	Reserved									
0x1700	ADCBASE	7:0	ADCBASE[7:0]							
		15:8	ADCBASE[15:8]							
		23:16								
		31:24								
0x1704 ... 0x170F	Reserved									
0x1710	ADCDMASTAT	7:0								RAF0
		15:8	DMACNTEN							RAF0IEN
		23:16	WROVRERR							RBF0
		31:24	DMAEN							RBF0IEN
0x1714 ... 0x171F	Reserved									
0x1720	ADCCNTB	7:0	ADCCNTB[7:0]							
		15:8	ADCCNTB[15:8]							
		23:16	ADCCNTB[23:16]							
		31:24	ADCCNTB[31:24]							
0x1724 ... 0x172F	Reserved									
0x1730	ADCDMAB	7:0	ADDMAB[7:0]							
		15:8	ADDMAB[15:8]							
		23:16	ADDMAB[23:16]							
		31:24	ADDMAB[31:24]							
0x1734 ... 0x173F	Reserved									
0x1740	ADCTRGSNS	7:0	LVL7	LVL6	LVL5	LVL4	LVL3	LVL2	LVL1	LVL0
		15:8								
		23:16								
		31:24								
0x1744 ... 0x17FF	Reserved									
0x1800	ADCANCON	7:0	ANEN7							ANEN0
		15:8	WKRDY7							WKRDY0
		23:16	WKIEN7							WKIEN0
		31:24					WKUPCLKCNT[3:0]			
0x1804 ... 0x1AFF	Reserved									
0x1B00	ADCSYSCFG0	7:0	AN[7:0]							
		15:8	AN[15:8]							
		23:16					AN[19:16]			
		31:24								
0x1B04 ... 0x1DFF	Reserved									

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E00	ADCDATAx	7:0	DATA[7:0]							
		15:8	DATA[15:8]							
		23:16	DATA[23:16]							
		31:24	DATA[31:24]							

38.11.2 ADCCON1 – ADC Control Register 1

Name: ADCCON1
Offset: 0x1400
Reset: 0x00601000
Property: -

This register controls the basic operation of the ADC module, including behavior in Sleep and Idle modes, and data formatting. This register also specifies the vector shift amounts for the Interrupt Controller. Additional ADCCON1 functions include the RAM buffer length in DMA mode.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	FRACT	SELRES[1:0]		STRGSRC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON	FRZ	SIDL			FSYDMA	FSYUPB	SCANEN
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit	7	6	5	4	3	2	1	0
		IRQVS[2:0]			STRGLVL	DMABL[2:0]		
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 23 – FRACT Fractional Data Output Format bit

Value	Description
0	Integer
1	Fractional

Bits 22:21 – SELRES[1:0] Shared ADC (ADC2) Resolution bits

Note: Changing the resolution of the ADC does not shift the result in the corresponding ADCDATAx register. The result occupies 12 bits, with the corresponding lower unused bits set to '0'. For example, a resolution of 6 bits results in ADCDATAx[5:0] being set to '0' and ADCDATAx[11:6] holding the result.

Value	Description
11	12 bits (default)
10	10 bits
01	8 bits
00	6 bits

Bits 20:16 – STRGSRC[4:0] ScanTrigger Source Select bits

Value	Description
10001 - 11111	Reserved
10000	EVSYS_47
01111	EVSYS_46
01110	EVSYS_45
01101	EVSYS_44
01100	EVSYS_43
01011	EVSYS_42
01010	EVSYS_41

Value	Description
01001	EVSYS_40
01000	EVSYS_39
00111	EVSYS_38
00110	EVSYS_37
00101	EVSYS_36
00100	INT0 External interrupt
00011	Reserved
00010	Global level software trigger (GLSWTRG)
00001	Global software edge trigger (GSWTRG)
00000	No Trigger

Bit 15 – ON ADC Module Enable bit

Note: The ON bit must be set only after the ADC module is configured.

Value	Description
0	ADC module is disabled
1	ADC module is enabled

Bit 14 – FRZ Freeze in Debug Mode

Value	Description
0	Do not freeze in Debug mode
1	Freeze in Debug mode

Bit 13 – SIDL Stop in Idle Mode bit

Value	Description
0	Continue module operation in Idle mode
1	Discontinue module operation when device enters Idle mode

Bit 10 – FSYDMA Fast Synchronous DMA System Clock bit

Value	Description
0	Fast synchronous DMA system clock is disabled
1	Fast synchronous DMA system clock is enabled

Bit 9 – FSYUPB Fast Synchronous UPB Clock bit

Value	Description
0	Fast synchronous UPB clock is disabled
1	Fast synchronous UPB clock is enabled

Bit 8 – SCANEN SCAN Enable bit

Bits 6:4 – IRQVS[2:0] Interrupt Vector Shift bits

To determine the interrupt vector address, this bit specifies the amount of left-shift done to the ARDYx status bits in the ADCDSTAT1 and ADCDSTAT2 registers prior to adding with the ADCBASE register.
Interrupt Vector Address = Read Value of ADCBASE, and Read Value of ADCBASE = Value written to ADCBASE + x << IRQVS[2:0], where ‘x’ is the smallest active input ID from the ADCDSTAT1 or ADCDSTAT2 registers (which has highest priority).

Value	Description
111	Shift x left 7 bit position
110	Shift x left 6 bit position
101	Shift x left 5 bit position
100	Shift x left 4 bit position
011	Shift x left 3 bit position
010	Shift x left 2 bit position
001	Shift x left 1 bit position
000	Shift x left 0 bit position

Bit 3 – STRGLVL ScanTrigger High Level/Positive Edge Sensitivity bit

Value	Description
0	Scan trigger is positive edge sensitive. Once STRIG mode is selected (TRGSRCx[4:0] in the ADCTRGx register), only a single scan trigger is generated, which completes the scan of all selected analog inputs.
1	Scan trigger is high level sensitive. Once STRIG mode is selected (TRGSRCx[4:0] in the ADCTRGx register), the scan trigger continues for all selected analog inputs, until the STRIG option is removed.

Bits 2:0 – DMABL[2:0] DMA to System RAM Buffer Length Size

Defines the number of locations in system memory allocated per analog input for DMA interface use. As each output data is 16-bit wide, one location consists of 2 bytes. Therefore, the actual size reserved in the system RAM follows the formula: RAM Buffer Length in bytes = $2_{(DMABL+1)}$.

38.11.3 ADCCON2 – ADC Control Register 2

Name: ADCCON2
Offset: 0x1410
Reset: 0x00000000
Property: -

This register controls the reference selection for the ADC module, the sample time for the shared ADC module, interrupt enable for reference, early interrupt selection and clock division selection for the shared ADC.

Bit	31	30	29	28	27	26	25	24
	BGVRRDY	REFFLT	EOSRDY				SAMC[9:8]	
Access	R/HS/HC	R/HS/HC	R/HS/HC				R/W	R/W
Reset	0	0	0				0	0

Bit	23	22	21	20	19	18	17	16
	SAMC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BGVRIEN	REFFLTIEN	EOSIEN		ENXCNVRT			
Access	R/W	R/W	R/W		R/W			
Reset	0	0	0		0			

Bit	7	6	5	4	3	2	1	0
		ADCDIV[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 31 – BGVRRDY Band Gap Voltage/ADC Reference Voltage Status bit
Data processing is valid only after BGVRRDY is set by hardware, so the application code must check that the BGVRRDY bit is set to ensure data validity. This bit set to '0' when ON (ADCCON1[15]) = 0.

Value	Description
0	Either or both band gap voltage and ADC reference voltages (V_{REF}) are not ready
1	Both band gap voltage and ADC reference voltages (V_{REF}) are ready

Bit 30 – REFFLT Band Gap/V_{REF}/A_{VDD} BOR Fault Status bit
This bit is cleared when the ON bit (ADCCON1[15]) = 0 and the BGVRRDY bit = 1.

Value	Description
0	Band gap and V_{REF} voltage are working properly
1	Fault in band gap or the V_{REF} voltage while the ON bit (ADCCON1[15]) was set. Most likely a band gap or V_{REF} fault is caused by a BOR of the analog V_{DD}supply.

Bit 29 – EOSRDY End of Scan Interrupt Status bit
This bit is cleared when ADCCON2[31:24] are read in software.

Value	Description
0	Scanning has not completed
1	All analog inputs are considered for scanning through the scan trigger (all analog inputs specified in the ADCCSS1 and ADCCSS2 registers) have completed scanning

Bits 25:16 – SAMC[9:0] SampleTime for the Shared ADC (ADC2) bits
Where T_{AD7} = Period of the ADC conversion clock for the Shared ADC (ADC2) controlled by the ADCDIV[6:0] bits.

Value	Description
1111111111	1025 T_{AD7}
...	

Value	Description
00000000 01	3 T_{AD7}
00000000 00	2 T_{AD7}

Bit 15 – BGVRIEN Band Gap/V_{REF} Voltage Ready Interrupt Enable bit

Value	Description
0	No interrupt is generated when the BGVRRDY bit is set
1	Interrupt is generated when the BGVRDDY bit is set

Bit 14 – REFFLTIEN Band Gap/V_{REF} Voltage Fault Interrupt Enable bit

Value	Description
0	No interrupt is generated when the REFFLT bit is set
1	Interrupt is generated when the REFFLT bit is set

Bit 13 – EOSIEN End of Scan Interrupt Enable bit

Value	Description
0	No interrupt is generated when the EOSRDY bit is set
1	Interrupt is generated when the EOSRDY bit is set

Bit 11 – ENXCNVRT Enable External Conversion Request Interface

Setting this bit enables another module (such as the PTG) to specify and request conversion of an ADC input.

Note: The external module (such as the PTG) is responsible for asserting only the proper trigger signals. This ADC module has no method to block specific trigger requests from the external module.

Bits 6:0 – ADCDIV[6:0] Division Ratio for the Shared SAR ADC Core Clock bits

The ADCDIV[6:0] bits divide the ADC control clock (T_Q) to generate the clock for the shared SAR ADC.

Value	Description
1111111	254 * T_Q = T_{AD2}
...	
0000011	6 * T_Q = T_{AD2}
0000010	4 * T_Q = T_{AD2}
0000001	2 * T_Q = T_{AD2}
0000000	Reserved

38.11.4 ADCCON3 – ADC Control Register 3

Name: ADCCON3
Offset: 0x1420
Reset: 0x00000000
Property: -

This register enables ADC clock selection, enables/disables the digital feature for the shared ADC module and controls the manual (software) sampling and conversion.

Bit	31	30	29	28	27	26	25	24
	ADCSEL[1:0]		CONCLKDIV[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CHN_EN_SHR							
Access	R/W							
Reset	0							

Bit	15	14	13	12	11	10	9	8
	VREFSEL[2:0]			TRGSUSP	UPDIEN	UPDRDY	SAMP	RQCNVRT
Access	R/W	R/W	R/W	R/W	R/W	R/HS/HC	R/W	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	GLSWTRG	GSWTRG	ADINSEL[5:0]					
Access	R/W	R/W, HC	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:30 – ADCSEL[1:0] Analog-to-Digital Clock Source (T_{CLK}) bits

Value	Description
00	Peripheral Bus Clock
01	FRC Clock
10	REFO3 Clock Output
11	System Clock (SYS_CLK)

Bits 29:24 – CONCLKDIV[5:0] Analog-to-Digital Control Clock (T_Q) Divider bits

Value	Description
111111	64 * T_{CLK}= T_Q
...	
000011	4 * T_{CLK}= T_Q
000010	3 * T_{CLK}= T_Q
000001	2 * T_{CLK}= T_Q
000000	T_{CLK}= T_Q

Bit 23 – CHN_EN_SHR Shared ADC Digital Enable bit

Value	Description
1	ADC is digital enabled
0	ADC is digital disabled

Bits 15:13 – VREFSEL[2:0] Voltage Reference (V_{REF}) Input Selection bits

Table 38-5.

VREFSEL[2:0]	AD_{REF+}	AD_{REF-}
000	AV_{DD}	AV_{SS}
001-111	RESERVED FOR FUTURE USE	

Bit 12 – TRGSUSP Trigger Suspend bit

Value	Description
1	Triggers are blocked from starting a new analog-to-digital conversion, but the ADC module is not disabled
0	Triggers are not blocked

Bit 11 – UPDIEN Update Ready Interrupt Enable bit

Value	Description
1	Interrupt is generated when the UPDRDY bit is set by hardware
0	No interrupt is generated

Bit 10 – UPDRDY ADC Update Ready Status bit

Note: This bit is only active while the TRGSUSP bit is set and there are no more running conversions of any ADC modules.

Value	Description
1	ADC SFRs can be updated
0	ADC SFRs cannot be updated

Bit 9 – SAMP Class 2 and Class 3 Analog Input Sampling Enable bit(1,2,3,4)

Value	Description
1	The ADC S&H amplifier is sampling
0	The ADC S&H amplifier is holding

Bit 8 – RQCNVRT Individual ADC Input Conversion Request bit

This bit and its associated ADINSEL[5:0] bits enable the user to individually request an analog-to-digital conversion of an analog input through software.

Note: This bit is automatically cleared in the next ADC clock cycle.

Value	Description
1	Trigger the conversion of the selected ADC input as specified by the ADINSEL[5:0] bits
0	Do not trigger the conversion

Bit 7 – GLSWTRG Global Level Software Trigger bit

Value	Description
1	Trigger conversion for ADC inputs that have selected the GLSWTRG bit as the trigger signal, either through the associated TRGSRC[4:0] bits in the ADCTRGx registers or through the STRGSRC[4:0]bits in the ADCCON1 register
0	Do not trigger an analog-to-digital conversion

Bit 6 – GSWTRG Global Software Trigger bit

This bit is automatically cleared in the next ADC clock cycle.

Value	Description
0	Trigger conversion for ADC inputs that have selected the GSWTRG bit as the trigger signal, either through the associated TRGSRC[4:0] bits in the ADCTRGx registers or through the STRGSRC[4:0]bits in the ADCCON1 register
1	Do not trigger an analog-to-digital conversion

Bits 5:0 – ADINSEL[5:0] Analog Input Select bits

These bits select the analog input to be converted when the RQCNVRT bit is set.

Note:

1. The SAMP bit has the highest priority and setting this bit keeps the S&H circuit in Sample mode until the bit is cleared. Also, usage of the SAMP bit causes settings of SAMC[9:0] bits (ADCCON2[25:16]) to be ignored.
2. The SAMP bit only connects Class 2 and Class 3 analog inputs to the shared ADC.
3. The SAMP bit is not a self-clearing bit and it is the responsibility of application software to first clear this bit and, only after setting the RQCNVRT bit, to start the analog-to-digital conversion.
4. Normally, when the SAMP and RQCNVRT bits are used by software routines, all TRGSRCx[4:0] bits and STRGSRC[4:0] bits must be set to '00000' to disable all external hardware triggers and prevent them from interfering with the software-controlled sampling command signal SAMP and with the software-controlled trigger RQCNVRT.

Value	Description
111111	Reserved
...	
001011	PMU Test Output
001010	VddCore (Internal)
001001	CP_Test_1.2V (Internal)
001000	BandGap Reference (Internal)
000111	AN7 is being monitored
...	
000001	AN1 is being monitored
000000	AN0 is being monitored

38.11.5 ADCIMCON1 – ADC Input Mode Control Register 1

Name: ADCIMCON1
Offset: 0x1440
Reset: 0x00000000
Property: -

This register enables the user to select between single-ended and differential operation as well as select between signed and unsigned data format.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DIFF11	SIGN11	DIFF10	SIGN10	DIFF9	SIGN9	DIFF8	SIGN8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIFF7	SIGN7	DIFF6	SIGN6	DIFF5	SIGN5	DIFF4	SIGN4
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DIFF3	SIGN3	DIFF2	SIGN2	DIFF1	SIGN1	DIFF0	SIGN0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 23 – DIFF11 AN11 Mode bit

Value	Description
1	AN11 is using Differential mode
0	AN11 is using Single-ended mode

Bit 22 – SIGN11 AN11 Signed Data Mode bit

Value	Description
1	AN11 is using Signed Data mode
0	AN11 is using Unsigned Data mode

Bit 21 – DIFF10 AN10 Mode bit

Value	Description
1	AN10 is using Differential mode
0	AN10 is using Single-ended mode

Bit 20 – SIGN10 AN10 Signed Data Mode bit

Value	Description
1	AN10 is using Signed Data mode
0	AN10 is using Unsigned Data mode

Bit 19 – DIFF9 AN9 Mode bit

Value	Description
1	AN9 is using Differential mode
0	AN9 is using Single-ended mode

Bit 18 – SIGN9 AN9 Signed Data Mode bit

Value	Description
1	AN9 is using Signed Data mode
0	AN9 is using Unsigned Data mode

Bit 17 – DIFF8 AN8 Mode bit

Value	Description
1	AN8 is using Differential mode
0	AN8 is using Single-ended mode

Bit 16 – SIGN8 AN8 Signed Data Mode bit

Value	Description
1	AN8 is using Signed Data mode
0	AN8 is using Unsigned Data mode

Bit 15 – DIFF7 AN7 Mode bit

Value	Description
1	AN7 is using Differential mode
0	AN7 is using Single-ended mode

Bit 14 – SIGN7 AN7 Signed Data Mode bit

Value	Description
1	AN7 is using Signed Data mode
0	AN7 is using Unsigned Data mode

Bit 13 – DIFF6 AN6 Mode bit

Value	Description
1	AN6 is using Differential mode
0	AN6 is using Single-ended mode

Bit 12 – SIGN6 AN6 Signed Data Mode bit

Value	Description
1	AN6 is using Signed Data mode
0	AN6 is using Unsigned Data mode

Bit 11 – DIFF5 AN5 Mode bit

Value	Description
1	AN5 is using Differential mode
0	AN5 is using Single-ended mode

Bit 10 – SIGN5 AN5 Signed Data Mode bit

Value	Description
1	AN5 is using Signed Data mode
0	AN5 is using Unsigned Data mode

Bit 9 – DIFF4 AN4 Mode bit

Value	Description
1	AN4 is using Differential mode
0	AN4 is using Single-ended mode

Bit 8 – SIGN4 AN4 Signed Data Mode bit

Value	Description
1	AN4 is using Signed Data mode
0	AN4 is using Unsigned Data mode

Bit 7 – DIFF3 AN3 Mode bit

Value	Description
1	AN3 is using Differential mode
0	AN3 is using Single-ended mode

Bit 6 – SIGN3 AN3 Signed Data Mode bit

Value	Description
1	AN3 is using Signed Data mode
0	AN3 is using Unsigned Data mode

Bit 5 – DIFF2 AN2 Mode bit

Value	Description
1	AN2 is using Differential mode
0	AN2 is using Single-ended mode

Bit 4 – SIGN2 AN2 Signed Data Mode bit

Value	Description
1	AN2 is using Signed Data mode
0	AN2 is using Unsigned Data mode

Bit 3 – DIFF1 AN1 Mode bit

Value	Description
1	AN1 is using Differential mode
0	AN1 is using Single-ended mode

Bit 2 – SIGN1 AN1 Signed Data Mode bit

Value	Description
1	AN1 is using Signed Data mode
0	AN1 is using Unsigned Data mode

Bit 1 – DIFF0 AN0 Mode bit

Value	Description
1	AN0 is using Differential mode
0	AN0 is using Single-ended mode

Bit 0 – SIGN0 AN0 Signed Data Mode bit

Value	Description
1	AN0 is using Signed Data mode
0	AN0 is using Unsigned Data mode

38.11.6 ADCGIRQEN1 – ADC Global Interrupt Enable Register 1

Name: ADCGIRQEN1
Offset: 0x1480
Reset: 0x00000000
Property: -

This register specifies which of the individual input conversion interrupts can generate the global ADC interrupt.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					AGIEN11	AGIEN10	AGIEN9	AGIEN8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	AGIEN7	AGIEN6	AGIEN5	AGIEN4	AGIEN3	AGIEN2	AGIEN1	AGIEN0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – AGIEN ADC Global Interrupt Enable bits

Value	Description
1	Interrupts are enabled for the selected analog input. The interrupt is generated after the converted data is ready (indicated by the ARDYx bit (‘x’ = 8-1) of the ADCDSTAT1 register)
0	Interrupts are disabled

38.11.7 ADCCSS1 – ADC Common Scan Select Register 1

Name: ADCCSS1
Offset: 0x14A0
Reset: 0x00000000
Property: -

This register specifies the analog inputs to be scanned by the common scan trigger.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					CSS11	CSS10	CSS9	CSS8
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CSS7	CSS6	CSS5	CSS4	CSS3	CSS2	CSS1	CSS0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – CSS Analog Common Scan Select bits

Notes:

1. In addition to setting the appropriate bits in this register, Class 2 analog inputs must select the STRIG input as the trigger source if they are to be scanned through the CSSx bits. Refer to the bit descriptions in the ADCTRGx registers for selecting the STRIG option.
2. If a Class 2 input is included in the scan by setting the CSSx bit to '1' and by setting the TRGSRCx[4:0] bits to STRIG mode (0b11), the user application must ensure that no other triggers are generated for that input using the RQCNVRT bit in the ADCCON3 register or the hardware input or any digital filter. Otherwise, the scan behavior is unpredictable.

Value	Description
1	Select ANx for input scan
0	Skip ANx for input scan

38.11.8 ADCDSTAT1 – ADC Data Ready Status Register 1

Name: ADCDSTAT1
Offset: 0x14C0
Reset: 0x00000000
Property: -

This register contains the interrupt status of the individual analog input conversions. Each bit represents the data-ready status for its associated conversion result.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					ARDY11	ARDY10	ARDY9	ARDY8
Access					R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ARDY7	ARDY6	ARDY5	ARDY4	ARDY3	ARDY2	ARDY1	ARDY0
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – ARDY Conversion Data Ready for Corresponding Analog Input Ready bits

Value	Description
1	This bit is set when converted data is ready in the data register
0	This bit is cleared when the associated data register is read

38.11.9 ADCCMPEN1 – ADC Digital Comparator 1 Enable Register

Name: ADCCMPEN1
Offset: 0x14E0
Reset: 0x00000000
Property: -

These registers select which analog input conversion results is processed by the digital comparator.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CMPEx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CMPEx[7:0] ADC Digital Comparator ‘x’ Enable bits
Note: CMPEx = where "x" stands for bit value from 0 to 7.

These bits enable conversion results corresponding to the analog input to be processed by the digital comparator. CMPE0 enables AN0, CMPE1 enables AN1 and so on.
Notes:

1. CMPEx = ANx, where ‘x’ = 0-31 (Digital Comparator inputs are limited to AN0 through AN31).
2. Changing the bits in this register while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.

38.11.10 ADCCMPEN2 – ADC Digital Comparator 2 Enable Register

Name: ADCCMPEN2
Offset: 0x1500
Reset: 0x00000000
Property: -

These registers select which analog input conversion results is processed by the digital comparator.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CMPEx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CMPEx[7:0] ADC Digital Comparator ‘x’ Enable bits
Note: CMPEx = where ‘x’ stands for bit value from 0 to 7.

These bits enable conversion results corresponding to the analog input to be processed by the digital comparator. CMPE0 enables AN0, CMPE1 enables AN1 and so on.
Notes:

1. CMPEx = ANx, where ‘x’ = 0-31 (Digital Comparator inputs are limited to AN0 through AN31).
2. Changing the bits in this register while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.

38.11.11 ADCCMP1 – ADC Digital Comparator 1 Limit Value Register

Name: ADCCMP1
Offset: 0x14F0
Reset: 0x00000000
Property: -

These registers contain the high and low digital comparison values for use by the digital comparator.

Notes:

1. Changing theses bits while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.
2. The format of the limit values must match the format of the ADC converted value in terms of sign and fractional settings.
3. For Digital Comparator 0 used in CVD mode, the DCMPHI[15:0] and DCMPLO[15:0] bits must always be specified in signed format as the CVD output data is differential and is always signed.

Bit	31	30	29	28	27	26	25	24
	DCMPHI[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCMPHI[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DCMPLO[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCMPLO[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – DCMPHI[15:0] Digital Comparator ‘x’ High Limit Value bits(1,2,3)
These bits store the high limit value, which is used by digital comparator for comparisons with ADC converted data.

Bits 15:0 – DCMPLO[15:0] Digital Comparator ‘x’ Low Limit Value bits(1,2,3)
These bits store the low limit value, which is used by digital comparator for comparisons with ADC converted data.

38.11.12 ADCCMP2 – ADC Digital Comparator 2 Limit Value Register

Name: ADCCMP2
Offset: 0x1510
Reset: 0x00000000
Property: -

These registers contain the high and low digital comparison values for use by the digital comparator.

Notes:

1. Changing theses bits while the Digital Comparator is enabled (ENDCMP = 1) can result in unpredictable behavior.
2. The format of the limit values must match the format of the ADC converted value in terms of sign and fractional settings.
3. For Digital Comparator 0 used in CVD mode, the DCMPHI[15:0] and DCMPLO[15:0] bits must always be specified in signed format as the CVD output data is differential and is always signed.

Bit	31	30	29	28	27	26	25	24
	DCMPHI[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCMPHI[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DCMPLO[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCMPLO[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – DCMPHI[15:0] Digital Comparator ‘x’ High Limit Value bits(1,2,3)
These bits store the high limit value, which is used by digital comparator for comparisons with ADC converted data.

Bits 15:0 – DCMPLO[15:0] Digital Comparator ‘x’ Low Limit Value bits(1,2,3)
These bits store the low limit value, which is used by digital comparator for comparisons with ADC converted data.

38.11.13 ADCFLTR1 – ADC Digital Filter 1 Register

Name: ADCFLTR1
Offset: 0x15A0
Reset: 0x00000000
Property: -

These registers provide control and status bits for the oversampling filter accumulator, and also includes the 16-bit filter output data.

Bit	31	30	29	28	27	26	25	24
	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				CHNLID[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FLTRDATA[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLTRDATA[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit 31 – AFEN Digital Filter ‘x’ Enable bit

Value	Description
1	Digital filter is enabled
0	Digital filter is disabled and the AFRDY status bit is cleared

Bit 30 – DATA16EN Filter Significant Data Length bit

Note: This bit is significant only if DFMODE = 1 (Averaging Mode) and FRACT (ADCCON1[23]) = 1 (Fractional Output Mode).

Value	Description
1	All 16 bits of the filter output data are significant
0	Only the first 12 bits are significant, followed by four zeros

Bit 29 – DFMODE ADC Filter Mode bit

Value	Description
1	Filter ‘x’ works in Averaging mode
0	Filter ‘x’ works in Oversampling Filter mode (default)

Bits 28:26 – OVRSAM[2:0] Oversampling Filter Ratio bits

Value	Description
	If DFMODE is ‘0’
111	128 samples (shift sum 3 bits to right, output data is in 15.1 format)
110	32 samples (shift sum 2 bits to right, output data is in 14.1 format)
101	8 samples (shift sum 1 bit to right, output data is in 13.1 format)
100	2 samples (shift sum 0 bits to right, output data is in 12.1 format)
011	256 samples (shift sum 4 bits to right, output data is 16 bits)
010	64 samples (shift sum 3 bits to right, output data is 15 bits)

Value	Description
001	16 samples (shift sum 2 bits to right, output data is 14 bits)
000	4 samples (shift sum 1 bit to right, output data is 13 bits)
	If DFMODE is '1'
111	256 samples (256 samples to be averaged)
110	128 samples (128 samples to be averaged)
101	64 samples (64 samples to be averaged)
100	32 samples (32 samples to be averaged)
011	16 samples (16 samples to be averaged)
010	8 samples (8 samples to be averaged)
001	4 samples (4 samples to be averaged)
000	2 samples (2 samples to be averaged)

Bit 25 – AFGIEN Digital Filter 'x' Interrupt Enable bit

Value	Description
1	Digital filter interrupt is enabled and is generated by the AFRDY status bit
0	Digital filter is disabled

Bit 24 – AFRDY Digital Filter 'x' Data Ready Status bit

Note: This bit is cleared by reading the FLTRDATA[15:0] bits or by disabling the Digital Filter module (by setting AFEN to '0').

Value	Description
1	Data is ready in the FLTRDATA[15:0] bits
0	Data is not ready

Bits 20:16 – CHNLID[4:0] Digital Filter Analog Input Selection bits

Note: Only the first 12 analog inputs, Class 2 (AN0 -AN11), can use a digital filter.

These bits specify the analog input to be used as the oversampling filter data source.

Value	Description
11111	Reserved
...	
...	
...	
01100	Reserved
01011	AN11
...	
...	
...	
00010	AN2
00001	AN1
00000	AN0

Bits 15:0 – FLTRDATA[15:0] Digital Filter 'x' Data Output Value bits

The filter output data is as per the fractional format set in the FRACT bit (ADCCON1[23]). The FRACT bit must not be changed while the filter is enabled. Changing the state of the FRACT bit after the operation of the filter ended must not update the value of the FLTRDATA[15:0] bits to reflect the new format.

38.11.14 ADCFLTR2 – ADC Digital Filter 2 Register

Name: ADCFLTR2
Offset: 0x15B0
Reset: 0x00000000
Property: -

These registers provide control and status bits for the oversampling filter accumulator, and also include the 16-bit filter output data.

Bit	31	30	29	28	27	26	25	24
	AFEN	DATA16EN	DFMODE	OVRSAM[2:0]			AFGIEN	AFRDY
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				CHNLID[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FLTRDATA[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLTRDATA[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit 31 – AFEN Digital Filter ‘x’ Enable bit

Value	Description
1	Digital filter is enabled
0	Digital filter is disabled and the AFRDY status bit is cleared

Bit 30 – DATA16EN Filter Significant Data Length bit

Note: This bit is significant only if DFMODE = 1 (Averaging Mode) and FRACT (ADCCON1[23]) = 1 (Fractional Output Mode).

Value	Description
1	All 16 bits of the filter output data are significant
0	Only the first 12 bits are significant, followed by four zeros

Bit 29 – DFMODE ADC Filter Mode bit

Value	Description
1	Filter ‘x’ works in Averaging mode
0	Filter ‘x’ works in Oversampling Filter mode (default)

Bits 28:26 – OVRSAM[2:0] Oversampling Filter Ratio bits

Value	Description
	If DFMODE is ‘0’
111	128 samples (shift sum 3 bits to right, output data is in 15.1 format)
110	32 samples (shift sum 2 bits to right, output data is in 14.1 format)
101	8 samples (shift sum 1 bit to right, output data is in 13.1 format)
100	2 samples (shift sum 0 bits to right, output data is in 12.1 format)
011	256 samples (shift sum 4 bits to right, output data is 16 bits)
010	64 samples (shift sum 3 bits to right, output data is 15 bits)

Value	Description
001	16 samples (shift sum 2 bits to right, output data is 14 bits)
000	4 samples (shift sum 1 bit to right, output data is 13 bits)
	If DFMODE is '1'
111	256 samples (256 samples to be averaged)
110	128 samples (128 samples to be averaged)
101	64 samples (64 samples to be averaged)
100	32 samples (32 samples to be averaged)
011	16 samples (16 samples to be averaged)
010	8 samples (8 samples to be averaged)
001	4 samples (4 samples to be averaged)
000	2 samples (2 samples to be averaged)

Bit 25 – AFGIEN Digital Filter 'x' Interrupt Enable bit

Value	Description
1	Digital filter interrupt is enabled and is generated by the AFRDY status bit
0	Digital filter is disabled

Bit 24 – AFRDY Digital Filter 'x' Data Ready Status bit

Note: This bit is cleared by reading the FLTRDATA[15:0] bits or by disabling the Digital Filter module (by setting AFEN to '0').

Value	Description
1	Data is ready in the FLTRDATA[15:0] bits
0	Data is not ready

Bits 20:16 – CHNLID[4:0] Digital Filter Analog Input Selection bits

Note: Only the first 12 analog inputs, Class 2 (AN0-AN11), can use a digital filter.

These bits specify the analog input to be used as the oversampling filter data source.

Value	Description
11111	Reserved
...	
...	
...	
01100	Reserved
01011	AN11
...	
...	
...	
00010	AN2
00001	AN1
00000	AN0

Bits 15:0 – FLTRDATA[15:0] Digital Filter 'x' Data Output Value bits

The filter output data is as per the fractional format set in the FRACT bit (ADCCON1[23]). The FRACT bit must not be changed while the filter is enabled. Changing the state of the FRACT bit after the operation of the filter ended must not update the value of the FLTRDATA[15:0] bits to reflect the new format.

38.11.15 ADCTRG1 – ADC Trigger Source 1 Register

Name: ADCTRG1
Offset: 0x1600
Reset: 0x00000000
Property: -

This register controls the trigger source selection for AN0 through AN3 analog inputs.

Bit	31	30	29	28	27	26	25	24
				TRGSRC3[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				TRGSRC2[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				TRGSRC1[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				TRGSRC0[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – TRGSRC3[4:0] Trigger Source for Conversion of Analog Input AN3 Select bits
Note: For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC[4:0] bits (ADCCON1[20:16]) to select the trigger source, and requires the appropriate CSS bits to be set in the ADCCSSx registers.

Value	Description
10001 - 11111	Reserved
10000	EVSYS_47
01111	EVSYS_46
01110	EVSYS_45
01101	EVSYS_44
01100	EVSYS_43
01011	EVSYS_42
01010	EVSYS_41
01001	EVSYS_40
01000	EVSYS_39
00111	EVSYS_38
00110	EVSYS_37
00101	EVSYS_36
00100	INT0 External interrupt
00011	STRIG
00010	Global level software trigger (GLSWTRG)
00001	Global software edge trigger (GSWTRG)
00000	No Trigger

Bits 20:16 – TRGSRC2[4:0] Trigger Source for Conversion of Analog Input AN2 Select bits
Note: See bits 28-24 for bit value definitions.

Bits 12:8 – TRGSRC1[4:0] Trigger Source for Conversion of Analog Input AN1 Select bits
Note: See bits 28-24 for bit value definitions.

Bits 4:0 – TRGSRC0[4:0] Trigger Source for Conversion of Analog Input AN0 Select bits
Note: See bits 28-24 for bit value definitions.

38.11.16 ADCTRG2 – ADC Trigger Source 2 Register

Name: ADCTRG2
Offset: 0x1610
Reset: 0x00000000
Property: -

This register controls the trigger source selection for AN4 through AN7 analog inputs.

Bit	31	30	29	28	27	26	25	24
				TRGSRC7[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				TRGSRC6[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				TRGSRC5[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				TRGSRC4[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – TRGSRC7[4:0] Trigger Source for Conversion of Analog Input AN7 Select bits
Note: For STRIG, in addition to setting the trigger, it also requires programming of the STRGSRC[4:0] bits (ADCCON1[20:16]) to select the trigger source, and requires the appropriate CSS bits to be set in the ADCCSSx registers.

Value	Description
10001 - 11111	Reserved
10000	EVSYS_47
01111	EVSYS_46
01110	EVSYS_45
01101	EVSYS_44
01100	EVSYS_43
01011	EVSYS_42
01010	EVSYS_41
01001	EVSYS_40
01000	EVSYS_39
00111	EVSYS_38
00110	EVSYS_37
00101	EVSYS_36
00100	INT0 External interrupt
00011	STRIG
00010	Global level software trigger (GLSWTRG)
00001	Global software edge trigger (GSWTRG)
00000	No Trigger

Bits 20:16 – TRGSRC6[4:0] Trigger Source for Conversion of Analog Input AN6 Select bits
Note: See bits 28-24 for bit value definitions.

Bits 12:8 – TRGSRC5[4:0] Trigger Source for Conversion of Analog Input AN5 Select bits
Note: See bits 28-24 for bit value definitions.

Bits 4:0 – TRGSRC4[4:0] Trigger Source for Conversion of Analog Input AN4 Select bits
Note: See bits 28-24 for bit value definitions.

38.11.17 ADCCMPCON1 – ADC Digital Comparator 1 Control Register

Name: ADCCMPCON1
Offset: 0x1680
Reset: 0x00000000
Property: -

This register controls the operation of Digital Comparator 1, including the generation of interrupts, comparison criteria to be used and provides status when a comparator event occurs.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			AINID[5:0]					
Access			R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
Access	R/W	R/W	R/HS/HC	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 13:8 – AINID[5:0] Digital Comparator 1 Analog Input Identification (ID) bits
When a digital comparator event occurs (DCMPED = 1), these bits identify the analog input being monitored by digital comparator 1.
Note: In normal ADC mode, only analog inputs [8:1] can be processed by the digital comparator 1.

Value	Description
111111	Reserved
...	
...	
...	
101101	Reserved
101100	Reserved
101011	Reserved
000111	AN7 is being monitored
...	
000001	AN1 is being monitored
000000	AN0 is being monitored

Bit 7 – ENDCMP Digital Comparator 1 Enable bit

Value	Description
1	Digital comparator 1 is enabled
0	Digital comparator 1 is not enabled, and the DCMPED status bit (ADCCMP0CON[5]) is cleared

Bit 6 – DCMPGIEN Digital Comparator 1 Global Interrupt Enable bit

Value	Description
1	A Digital comparator 1 interrupt is generated when the DCMPED status bit (ADCCMP0CON[5]) is set
0	A Digital comparator 1 interrupt is disabled

Bit 5 – DCMPED Digital Comparator 1 “Output True” Event Status bit

The logical conditions under which the digital comparator becomes “True” are defined by the IEBTWN, IEHIHI, IEHILO, IELOHI and IELOLO bits.

Note: This bit is cleared by reading the AINID[5:0] bits or by disabling the Digital Comparator module (by setting ENDCMP to ‘0’).

Value	Description
1	Digital comparator 1 output true event has occurred (output of comparator is ‘1’)
0	Digital comparator 1 output is false (output of comparator is ‘0’)

Bit 4 – IEBTWN Between Low/High Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator event when DATA[31:0] is less than DCMPHI[15:0] AND greater than DCMPLO[15:0]
0	Do not generate a digital comparator event

Bit 3 – IEHIHI High/High Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DCMPHI[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

Bit 2 – IEHILO High/Low Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DATA[31:0] bits are less than DCMPHI[15:0] bits
0	Do not generate an event

Bit 1 – IELOHI Low/High Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DCMPLO[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

Bit 0 – IELOLO Low/Low Digital Comparator 1 Event bit

Value	Description
1	Generate a digital comparator 1 event when DATA[31:0] bits are less than DCMPLO[15:0] bits
0	Do not generate an event

38.11.18 ADCCMPCON2 – ADC Digital Comparator 2 Control Register

Name: ADCCMPCON2
Offset: 0x1690
Reset: 0x00000000
Property: -

These registers control the operation of Digital Comparator 2, including the generation of interrupts and the comparison criteria to be used. This register also provides the status when a comparator event occurs.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				AINID[4:0]				
Access				R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ENDCMP	DCMPGIEN	DCMPED	IEBTWN	IEHIHI	IEHILO	IELOHI	IELOLO
Access	R/W	R/W	R/HS/HC	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 12:8 – AINID[4:0] Digital Comparator 2 Analog Input Identification (ID) bits
When a digital comparator event occurs (DCMPED = 1), these bits identify the analog input being monitored by the digital comparator.
Note: Only analog inputs [8:1] can be processed by the Digital Comparator module 'x' ('x' = 1-2).

Value	Description
11111	Reserved
11110	Reserved
...	
...	
...	
00011	Reserved
000111	AN7 is being monitored
...	
00001	AN1 is being monitored
00000	AN0 is being monitored

Bit 7 – ENDCMP Digital Comparator 2 Enable bit

Value	Description
1	Digital comparator 2 is enabled
0	Digital comparator 2 is not enabled, and the DCMPED status bit (ADCCMP0CON[5]) is cleared

Bit 6 – DCMPGIEN Digital Comparator 2 Global Interrupt Enable bit

Value	Description
1	Digital comparator 2 interrupt is generated when the DCMPED status bit (ADCCMP0CON[5]) is set
0	Digital comparator 2 interrupt is disabled

Bit 5 – DCMPED Digital Comparator 2 “Output True” Event Status bit

The logical conditions where the digital comparator gets “True” are defined by the IEBTWN, IEHIHI, IEHILO, IELOHI and IELOLO bits.

Note: This bit is cleared by reading the AINID[5:0] bits (ADCCMP0CON[13:8]) or by disabling the Digital Comparator module (by setting ENDCMP to ‘0’).

Value	Description
1	Digital comparator 2 output true event has occurred (output of comparator is ‘1’)
0	Digital comparator 2 output is false (output of comparator is ‘0’)

Bit 4 – IEBTWN Between Low/High Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator event when DCMPLO[15:0] bits DATA[31:0] bits [DCMPHI[15:0] bits
0	Do not generate a digital comparator event

Bit 3 – IEHIHI High/High Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DCMPHI[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

Bit 2 – IEHILO High/Low Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DATA[31:0] bits are less than DCMPHI[15:0] bits
0	Do not generate an event

Bit 1 – IELOHI Low/High Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DCMPLO[15:0] bits are less than or equal to DATA[31:0] bits
0	Do not generate an event

Bit 0 – IELOLO Low/Low Digital Comparator 2 Event bit

Value	Description
1	Generate a digital comparator 2 event when DATA[31:0] bits are less than DCMPLO[15:0] bits
0	Do not generate an event

38.11.19 ADCBASE – ADC Base Register

Name: ADCBASE
Offset: 0x1700
Reset: 0x00000000
Property: -

This register specifies the base address of the user ADC Interrupt Service Routine (ISR) jump table.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ADCBASE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADCBASE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – ADCBASE[15:0] ADCISR Base Address bits
This register, when read, contains the base address of the user's ADC ISR jump table. The interrupt vector address is determined by the IRQVS[2:0] bits of the ADCCON1 register specifying the amount of left shift done to the ARDYx status bits in the ADCDSTAT1 register, prior to adding with ADCBASE register.
Interrupt vector address = Read value of ADCBASE
Read value of ADCBASE = Value written to ADCBASE + x << IRQVS[2:0], where ‘x’ is the smallest active analog input ID from the ADCDSTAT1 register (which has the highest priority).

38.11.20 ADCDMASTAT – ADC DMA Status Register

Name: ADCDMASTAT
Offset: 0x1710
Reset: 0x00000000
Property: -

This register contains the DMA status bits.

Bit	31	30	29	28	27	26	25	24
	DMAEN							RBF0IEN
Access	R/W							R/W
Reset	0							0

Bit	23	22	21	20	19	18	17	16
	WROVRERR							RBF0
Access	R/HS/HC							R/HS/HC
Reset	0							0

Bit	15	14	13	12	11	10	9	8
	DMACNTEN							RAF0IEN
Access	R/W							R/W
Reset	0							0

Bit	7	6	5	4	3	2	1	0
								RAF0
Access								R/HS/HC
Reset								0

Bit 31 – DMAEN DMA Interface Enable bit
When DMAEN = 0, no data is being saved into the DMA FIFO, no SRAM writes occur, and the DMA interface logic is being kept in Reset state.

Value	Description
1	DMA interface is enabled
0	DMA interface is disabled

Bit 24 – RBF0IEN RAM Buffer B Full Interrupt Enable for channel 0

Value	Description
1	Interrupts are enabled and generated when the RBFx Status bit is set
0	Interrupts are disabled

Bit 23 – WROVRERR Write Overflow Error in the DMA FIFO
Set by hardware, cleared by hardware after a software read of the ADDMAST register.
Note: The write always occurs and the old data is replaced with new data because the software missed reading the old data on time.

Bit 16 – RBF0 RAM Buffer B FULL status bit for channel 0
This bit is self-clearing upon being read by software.

Bit 15 – DMACNTEN DMA Buffer Sample Count Enable bit
The DMA interface saves the current sample count for each buffer in the table starting at the ADCCNTB address after each sample write into the corresponding buffer in the SRAM.

Bit 8 – RAF0IEN RAM Buffer A FULL Interrupt Enable for channel 0

Value	Description
1	Interrupts are enabled and generated when the RAFx Status bit is set
0	Interrupts are disabled

Bit 0 – RAF0 RAM Buffer A FULL status bit for channel 0
This bit is self-clearing upon being read by software.

38.11.21 ADCCNTB – ADC Channel Sample Count Base Address Register

Name: ADCCNTB
Offset: 0x1720
Reset: 0x00000000
Property: -

This register contains the base address of the sample count in RAM. In addition to storing the converted data of the ADC module in RAM, DMA also stores the converted sample count.

Bit	31	30	29	28	27	26	25	24
	ADCCNTB[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADCCNTB[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADCCNTB[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADCCNTB[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ADCCNTB[31:0] ADC Channel Count Base Address
SRAM address for the DMA interface at which to save the first class channel buffer A sample count values into the System RAM. If first class channel x, x=0...6, is ready with a new available sample data and the DMA interface is currently saving data for channel x to RAM Buffer z (where z==0means BufferA and z==1means Buffer B, z depending on x), then the DMA interface will increment (+1) the 1 byte count value stored at the System RAM address (ADCCNTB + 2*x + z).
ADCCNTB works in conjunction with ADDMAB. The DMA interface uses ADCCNTB to save the buffer sample counts only if ADDMAST.DMA_CNT_EN is set to 1.

38.11.22 ADCDMAB – ADC DMA Base Address Register

Name: ADCDMAB
Offset: 0x1730
Reset: 0x00000000
Property: -

This register contains the base address of RAM for the DMA engine.

Bit	31	30	29	28	27	26	25	24
	ADDMAB[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDMAB[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDMAB[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDMAB[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ADDMAB[31:0] Base Address for the DMA Interface
BASE ADDRESS for the DMA interface at which to save first class channel data into the System RAM. If first class channel x, x == 0...6, is ready with new available sample data and the DMA interface is currently saving data for channel x to RAM Buffer z (where z ==`0` means Buffer A and z == `1` means Buffer B, z depending on x) and the current DMA x-counter value is y (y depending on x), then the DMA interface stores the 2-byte output data value at System RAM address (ADDMAB + (2*x + z)*2(DMABL+1) +2*y. Also, if ADDMAST.DMA_CNT_EN is set to `1`, the DMA interface stores (without delay) the value y itself at the System RAM address (ADCCNTB + 2*x + z).

38.11.23 ADCTRGSNS – ADC Trigger Level/Edge Sensitivity Register

Name: ADCTRGSNS
Offset: 0x1740
Reset: 0x00000000
Property: -

This register contains the setting for trigger level for each ADC analog input.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	LVL7	LVL6	LVL5	LVL4	LVL3	LVL2	LVL1	LVL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – LVL Trigger Level and Edge Sensitivity bits

Notes:

1. This register specifies the trigger level for analog inputs 0 to 7.
2. The higher analog input ID belongs to Class 3, and, therefore, is only scan triggered. All Class 3 analog inputs use the scan trigger, for which the level/edge is defined by the STRGLVL bit (ADCCON1[3]).

Value	Description
1	Analog input is sensitive to the high level of its trigger (level sensitivity implies retriggering as long as the trigger signal remains high)
0	Analog input is sensitive to the positive edge of its trigger (this is the value after a reset)

38.11.24 ADCANCON – ADC Analog Warm-up Control Register

Name: ADCANCON
Offset: 0x1800
Reset: 0x00000000
Property: -

This register contains the warm-up control settings for the analog and bias circuit of the ADC module.

Bit	31	30	29	28	27	26	25	24
					WKUPCLKCNT[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WKIEN7							WKIEN0
Access	R/W							R/W
Reset	0							0

Bit	15	14	13	12	11	10	9	8
	WKRDY7							WKRDY0
Access	R/HS/HC							R/HS/HC
Reset	0							0

Bit	7	6	5	4	3	2	1	0
	ANEN7							ANEN0
Access	R/W							R/W
Reset	0							0

Bits 27:24 – WKUPCLKCNT[3:0] Wake-up Clock Count bits
These bits represent the number of ADC clocks required to warm-up the ADC module before it can perform conversion. Although the clocks are specific to each ADC, the WKUPCLKCNT bit is common to all ADC modules.

Value	Description
1111	2^{15}= 32,768 clocks
...	
...	
...	
0110	2^{6}= 64 clocks
0101	2^{5}= 32 clocks
0100	2^{4}= 16 clocks
0011	2^{4}= 16 clocks
0010	2^{4}= 16 clocks
0001	2^{4}= 16 clocks
0000	2^{4}= 16 clocks

Bit 23 – WKIEN7 Shared ADC (ADC2) Wake-up Interrupt Enable bit

Value	Description
1	Enable interrupt and generate interrupt when the WKRDY2 status bit is set
0	Disable interrupt

Bit 16 – WKIEN0 ADC1Wake-up Interrupt Enable bit

Value	Description
1	Enable interrupt and generate interrupt when the WKRDYx status bit is set
0	Disable interrupt

Bit 15 – WKRDY7 Shared ADC (ADC2) Wake-up Status bit
Note: This bit is cleared by hardware when the ANEN2 bit is cleared.

Value	Description
1	ADC2 Analog and bias circuitry ready after the wake-up count number $2^{WKUPEXP}$ clocks after setting ANEN2 to '1'
0	ADC2 Analog and bias circuitry is not ready

Bit 8 – WKRDY0 ADC1 Wake-up Status bit

Note: This bit is cleared by hardware when the ANENx bit is cleared.

Value	Description
1	ADC1 Analog and bias circuitry ready after the wake-up count number $2^{WKUPEXP}$ clocks after setting ANEN1 to '1'
0	ADC1 Analog and bias circuitry is not ready

Bit 7 – ANEN7 Shared ADC (ADC2) Analog and Bias Circuitry Enable bit

Value	Description
1	Analog and bias circuitry enabled. Once the analog and bias circuit is enabled, the ADC module needs a warm-up time, as defined by the WKUPCLKCNT[3:0] bits.
0	Analog and bias circuitry disabled

Bit 0 – ANEN0 ADC1 Analog and Bias Circuitry Enable bits

Value	Description
1	Analog and bias circuitry enabled. Once the analog and bias circuit is enabled, the ADC module needs a warm-up time, as defined by the WKUPCLKCNT[3:0] bits.
0	Analog and bias circuitry disabled

38.11.25 ADCSYSCFG0 – ADC System Configuration Register 0

Name: ADCSYSCFG0
Offset: 0x1B00
Reset: 0x00000000
Property: -

This register contains read-only bits corresponding to the analog input.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					AN[19:16]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AN[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	AN[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bits 19:0 – AN[19:0] ADC Analog Input bits
These bits reflect the system configuration and are updated during boot-up time. By reading these read-only bits, the user application can determine whether or not an analog input in the device is available.

38.11.26 ADCDATAx – ADC Output Data Register ('x' = 0 to 11)

Name: ADCDATAx
Offset: 0x1Exx
Reset: 0x00000000
Property: -

These registers are the analog-to-digital conversion output data registers. The ADCDATAx register is associated with each external analog input, 0-7, plus internal analog inputs 8-11.

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA[31:0] ADC Converted Data Output bits

Notes:

1. When an alternate input is used as the input source for a dedicated ADC module, the data output is still read from the Primary input Data Output register.
2. Reading the ADCDATAx register value after changing the FRACT bit converts the data into the format specified by the FRACT bit.

39. Analog Comparators (AC)

39.1 Overview

The Analog Comparator (AC) supports individual comparators: one shared (with built-in supply voltage monitor (MVREF)) AC_CMP1 and one dedicated AC_CMP0.

A continuous mode of operation on a shared comparator is supported only if MVREF is disabled: PMU.CLKCTRL.MVREFFSMEN bit = 0 (Disabled). Each comparator (COMP) compares the voltage levels on two inputs and provides a digital output based on the comparison. Each comparator may be configured to generate interrupt requests and/or peripheral events upon several different combinations of input change.

The input selection includes up to four shared analog port pins and several internal signals. Each Comparator Output state can also be output on a pin for use by external devices.

The comparators are grouped in pairs on each port. The AC peripheral implements of comparators. These are called Comparator 0 (COMP0) and Comparator 1 (COMP1) They have identical behaviors but separate Control registers. pair can be set in Window mode to compare a signal to a voltage range instead of a single voltage level.

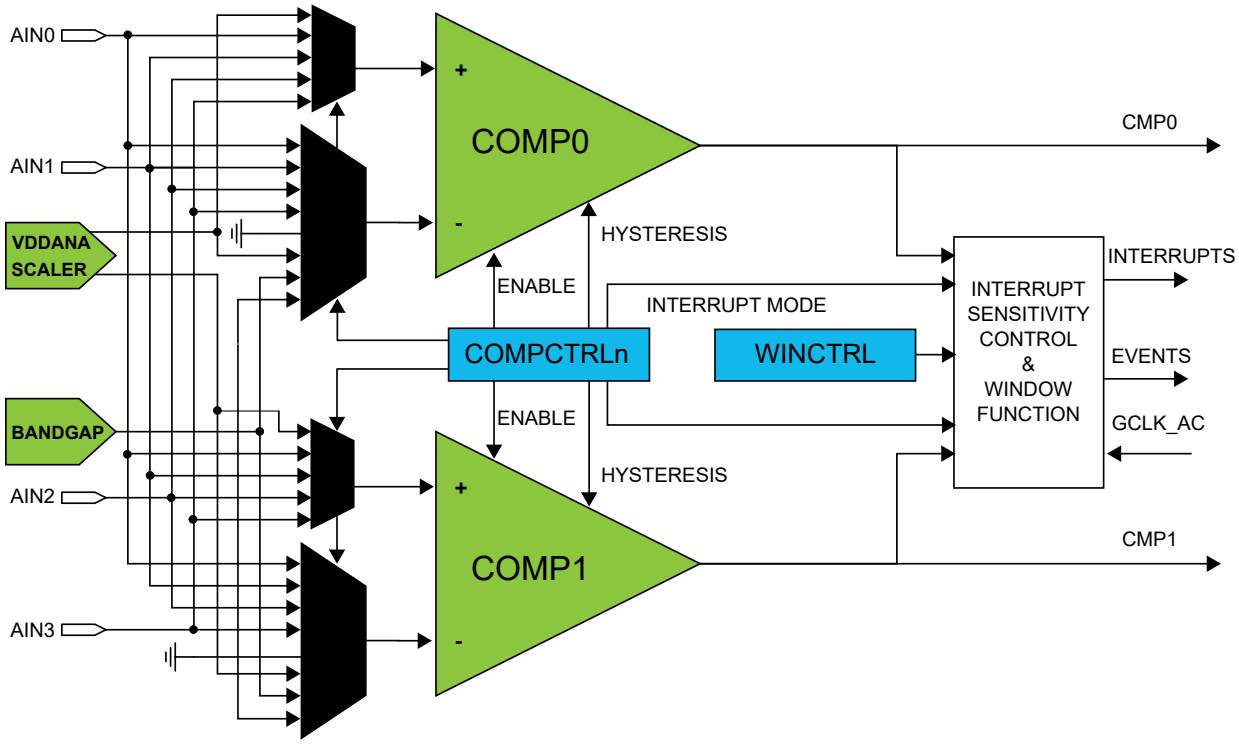
Note: MVREF will not be able to monitor the supply voltage for 275 µs from the time $\overline{\text{MCLR}}$ is released.

39.2 Features

- Two individual comparators (Single pair configuration)
- Analog comparator outputs available on pins
  - Asynchronous or synchronous
- Flexible input selection:
  - Up to four pins selectable for positive or negative inputs
  - Ground (for zero crossing)
  - Bandgap reference voltage
  - Programmable VDD scaler for AC_CMP1 (shared with Programmable Low Voltage Detector (PLVD)) and fixed VDD/2 for AC_CMP0
- Interrupt generation on:
  - Rising or falling edge
  - Toggle
  - End of comparison
- Window function interrupt generation on:
  - Signal above window
  - Signal inside window
  - Signal below window
  - Signal outside window
- Event generation on:
  - Comparator output
  - Window function inside/outside window
- Optional digital filter on comparator output

39.3 Block Diagram

Figure 39-1. Analog Comparator Block Diagram



39.4 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

39.4.1 I/O Lines

Using the AC's I/O lines requires the I/O pins to be configured as analog pins for AC_AINx inputs. See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.

Table 39-1. I/O LINES

Signal	Peripheral Function
AC_AIN0	—
AC_AIN1	—
AC_AIN2	—
AC_AIN3	—
AC_CMP0	CFGCON1.CMP0_OE
AC_CMP1	CFGCON1.CMP1_OE

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

39.4.2 Power Management

The AC will continue to operate in any Sleep mode where the selected source clock is running. The AC's interrupts can be used to wake-up the device from Standby Sleep mode. Events connected to the Event System can trigger other operations in the system without exiting Standby Sleep mode.

39.4.3 Clocks

The AC bus clock (PB2_CLK) can be enabled and disabled in the Main Clock module, CRU (see *Clock and Reset Unit (CRU)* from Related Links), and the Analog Comparator module can be enabled or disabled via the PMD1 register. See *Peripheral Module Disable Register (PMD)* from Related Links.

A generic clock (GCLK_AC) is required to clock the AC. This clock must be configured and enabled in the generic clock controller before using the AC.

This generic clock is asynchronous to the bus clock (PB2_CLK). Due to this asynchronicity, writes to certain registers will require synchronization between the clock domains. See *Synchronization* from Related Links.

Related Links

13. Clock and Reset Unit (CRU)
20. Peripheral Module Disable Register (PMD)
39.5.13. Synchronization

39.4.4 DMA

Not applicable.

39.4.5 Interrupts

The interrupt request lines are connected to the interrupt controller. Using the AC interrupts requires the interrupt controller to be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

39.4.6 Events

The events are connected to the Event System. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.

Related Links

28. Event System (EVSYS)

39.4.7 Debug Operation

When the CPU is halted in debug mode, the AC will halt normal operation after any ongoing comparison is completed. The AC can be forced to continue normal operation during debugging. See *DBGCTRL* register from Related Links. If the AC is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

Related Links

39.7.9. DBGCTRL

39.4.8 Register Access Protection

All registers with write access can be write-protected optionally by the Peripheral Access Controller (PAC), except for the following registers:

- Control B register (CTRLB)
- Interrupt Flag register (INTFLAG)

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

PAC write protection does not apply to accesses through an external debugger.

39.4.9 Analog Connections

Each comparator has up to four I/O pins that can be used as analog inputs. Each pair of comparators shares the same four pins. These pins must be configured for analog operation before using them as comparator inputs.

Any internal reference source, such as a bandgap voltage reference, must be configured and enabled prior to its use as a comparator input.

39.5 Functional Description

39.5.1 Principle of Operation

Each comparator has one positive input and one negative input. Each positive input may be chosen from a selection of analog input pins. Each negative input may be chosen from a selection of both analog input pins and internal inputs, such as bandgap voltage reference.

The digital output from the comparator is ‘1’ when the difference between the positive and the negative input voltage is positive, and ‘0’otherwise.

The individual comparators can be used independently (Normal mode) or paired to form a window comparison (Window mode).

39.5.2 Basic Operation

39.5.2.1 Initialization

Some registers are enable-protected, meaning they can only be written when the module is disabled.

The following register is enable-protected:

- Event Control register (EVCTRL)

Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

39.5.2.2 Enabling, Disabling and Resetting

The AC is enabled by writing a ‘1’ to the Enable bit in the Control A register (CTRLA.ENABLE). The AC is disabled by writing a ‘0’ to CTRLA.ENABLE.

The AC is reset by writing a ‘1’ to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the AC will be reset to their initial state, and the AC will be disabled. See *CTRLA* register from Related Links.

Related Links

39.7.1. CTRLA

39.5.2.3 Comparator Configuration

Each individual comparator must be configured by its respective Comparator Control register (COMPCTRLx) before that comparator is enabled. These settings cannot be changed while the comparator is enabled.

- Select the desired measurement mode with COMPCTRLx.SINGLE. See *Starting a Comparison* from Related Links.
- Fixed hysteresis does not support programmable Hysteresis.
- Fixed speed of operation
- Select the interrupt source with COMPCTRLx.INTSEL.
- Select the positive and negative input sources with the COMPCTRLx.MUXPOS and COMPCTRLx.MUXNEG bits. See *Selecting Comparator Inputs* from Related Links.
- Select the filtering option with COMPCTRLx.FLEN.
- Select the standby operation with the Run in Standby bit (COMPCTRLx.RUNSTDBY).

The individual comparators are enabled by writing a ‘1’ to the Enable bit in the Comparator x Control registers (COMPCTRLx.ENABLE). The individual comparators are disabled by writing a ‘0’ to COMPCTRLx.ENABLE. Writing a ‘0’ to CTRLA.ENABLE will also disable all the comparators but will not clear their COMPCTRLx.ENABLE bits.

Related Links

39.5.2.4. Starting a Comparison

39.5.3. Selecting Comparator Inputs

39.5.2.4 Starting a Comparison

Each comparator channel can be in one of two different measurement modes, which is determined by the COMPCTRLx.SINGLE bit:

- Continuous measurement
- Single-shot

After being enabled, a start-up delay is required before the result of the comparison is ready. This start-up time is measured automatically to account for environmental changes, such as temperature or voltage supply level, and is specified in the Electrical Specifications. During the start-up time, the COMP output is not available.

The comparator can be configured to generate interrupts when the output toggles, when the output changes from '0' to '1' (rising edge), when the output changes from '1' to '0' (falling edge) or at the end of the comparison. An end-of-comparison interrupt can be used with the Single-Shot mode to chain further events in the system, regardless of the state of the comparator outputs. The Interrupt mode is set by the Interrupt Selection bit group in the Comparator Control register (COMPCTRLx.INTSEL). Events are generated using the comparator output state regardless of whether the interrupt is enabled or not.

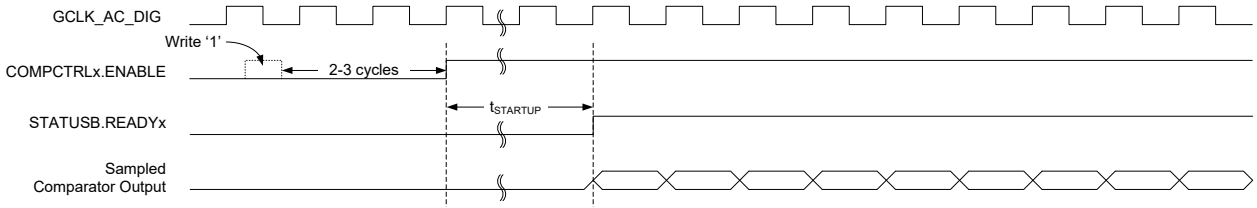
39.5.2.4.1 Continuous Measurement

Continuous measurement is selected by writing COMPCTRLx.SINGLE to zero. In continuous mode, the comparator is continuously enabled and performing comparisons. This ensures that the result of the latest comparison is always available in the Current State bit in the Status A register (STATUSA.STATEx).

After the start-up time has passed, a comparison is done and STATUSA is updated. The Comparator x Ready bit in the Status B register (STATUSB.READYx) is set, and the appropriate peripheral events and interrupts are also generated. New comparisons are performed continuously until the COMPCTRLx.ENABLE bit is written to zero. The start-up time applies only to the first comparison.

In the continuous operation, edge detection of the comparator output for interrupts is done by comparing the current and previous sample. The sampling rate is the GCLK_AC frequency. An example of continuous measurement is shown in the following figure.

Figure 39-2. Continuous Measurement Example



For low-power operation, comparisons can be performed during sleep mode without a clock. The comparator is enabled continuously, and changes of the comparator state are detected asynchronously. When a toggle occurs, the CRU will start GCLK_AC to register the appropriate peripheral events and interrupts. The GCLK_AC clock is, then, disabled again automatically unless configured to wake up the system from sleep.

39.5.2.4.2 Single-Shot

Single-shot operation is selected by writing COMPCTRLx.SINGLE to '1'. During single-shot operation, the comparator is normally idle. The user starts a single comparison by writing '1' to the respective Start Comparison bit in the write-only Control B register (CTRLB.STARTx). The comparator is enabled and, after the start-up time has passed, a single comparison is done and STATUSA is updated. Appropriate peripheral events and interrupts are also generated. No new comparisons will be performed.

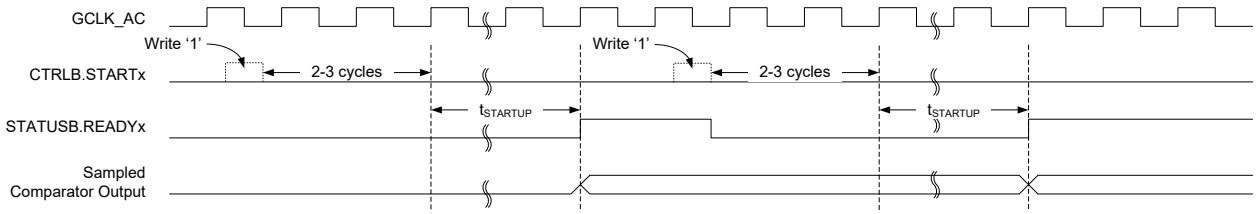
Writing '1' to CTRLB.STARTx also clears the Comparator x Ready bit in the Status B register (STATUSB.READYx). STATUSB.READYx is set automatically by hardware when the single comparison has completed.

A single-shot measurement can also be triggered by the Event System. Setting the Comparator x Event Input bit in the Event Control Register (EVCTRL.COMPEIx) enables triggering on incoming peripheral events. Each comparator can be triggered independently by separate events. Event-triggered operation is similar to user-triggered operation;

the difference is that a peripheral event from another hardware module causes the hardware to automatically start the comparison and will not clear STATUSB.READYx.

To detect an edge of the comparator output in single-shot operation for the purpose of interrupts, the result of the current measurement is compared with the result of the previous measurement (one sampling period earlier). An example of single-shot operation is shown in the following figure.

Figure 39-3. Single-Shot Example



For low-power operation, event-triggered measurements can be performed during sleep modes. When the event occurs, the CRU will start GCLK_AC. The comparator is enabled and, after the start-up time has passed, a comparison is done and appropriate peripheral events and interrupts are also generated. The comparator and GCLK_AC are, then, disabled again automatically unless configured to wake up the system from sleep.

39.5.3 Selecting Comparator Inputs

Each comparator has one positive and one negative input. The positive input is one of the external input pins (AINx). The negative input can be fed either from an external input pin (AINx) or from one of the internal reference voltage sources common to all comparators. The user selects the input source as follows:

- The positive input is selected by the Positive Input MUX Select bit group in the Comparator Control register (COMPCTRLx.MUXPOS).
- The negative input is selected by the Negative Input MUX Select bit group in the Comparator Control register (COMPCTRLx.MUXNEG).

In the case of using an external I/O pin, the selected pin must be configured for analog use in the GPIO by disabling the digital input and output. The switching of the analog input multiplexers is controlled to minimize crosstalk between the channels. The input selection must be changed only while the individual comparator is disabled.

Note: For internal use of the comparison results by the CCL module (see *Configurable Custom Logic (CCL)* from Related Links), COMPCTRLx.OUT must be `0x1` or `0x2`.

Related Links

34. Configurable Custom Logic (CCL)

39.5.4 Window Operation

Each comparator pair can be configured to work together in Window mode. In this mode, a voltage range is defined, and the comparators give information about whether an input signal is within this range or not. Window mode is enabled by the Window Enable x bit in the Window Control register (WINCTRL.WENx). Both comparators in a pair must have the same measurement mode setting in their respective Comparator Control Registers (COMPCTRLx.SINGLE).

Notes:

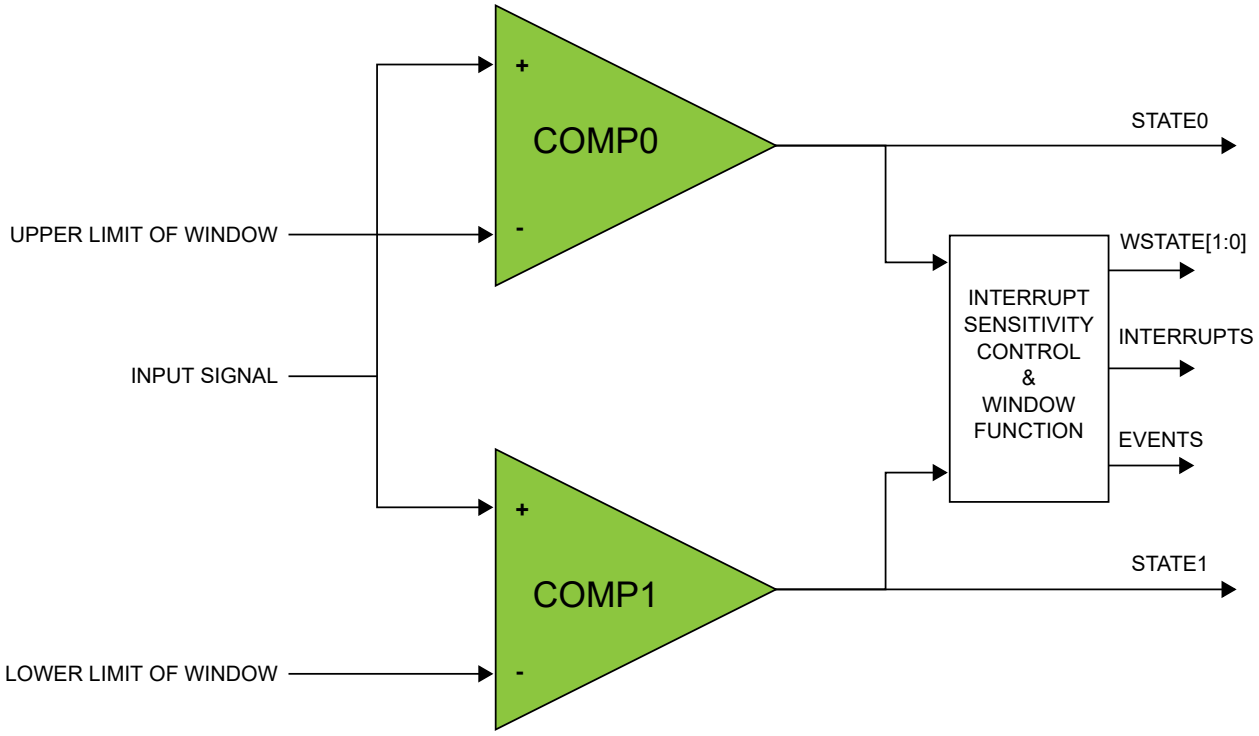
1. Both comparators must be enabled (COMPCTRLx.ENABLE = 1) before the Window mode is enabled (WINCTRL.WEN0 = 1).
2. Window mode must be first disabled (WINCTRL.WEN0 = 0) before both comparators are disabled (COMPCTRLx.ENABLE = 0).
3. The ready bits for both comparators must be checked (STATUSB.READYx = 1) before reading the measurement results.

To physically configure the pair of comparators for Window mode, the same I/O pin must be chosen as positive input for each comparator, providing a shared input signal. The negative inputs define the range for the window. In the figure below, COMP0 defines the upper limit and COMP1 defines the lower limit of the window, as shown but the window will also work in the opposite configuration with COMP0 lower and COMP1 higher. The current state of the window function is available in the Window x State bit group of the Status register (STATUS.WSTATEx).

Window mode can be configured to generate interrupts when the input voltage changes to below the window, when the input voltage changes to above the window, when the input voltage changes into the window or when the input voltage changes outside the window. The interrupt selections are set by the Window Interrupt Selection bit field in the Window Control register (WINCTRL.WINTSEL). Events are generated using the inside/outside state of the window, regardless of whether the interrupt is enabled or not. Note that the individual comparator outputs, interrupts and events continue to function normally during Window mode.

When the comparators are configured for Window mode and Single-shot mode, measurements are performed simultaneously on both comparators. Writing '1' to either Start Comparison bit in the Control B register (CTRLB.STARTx) will start a measurement. Likewise either peripheral event can start a measurement.

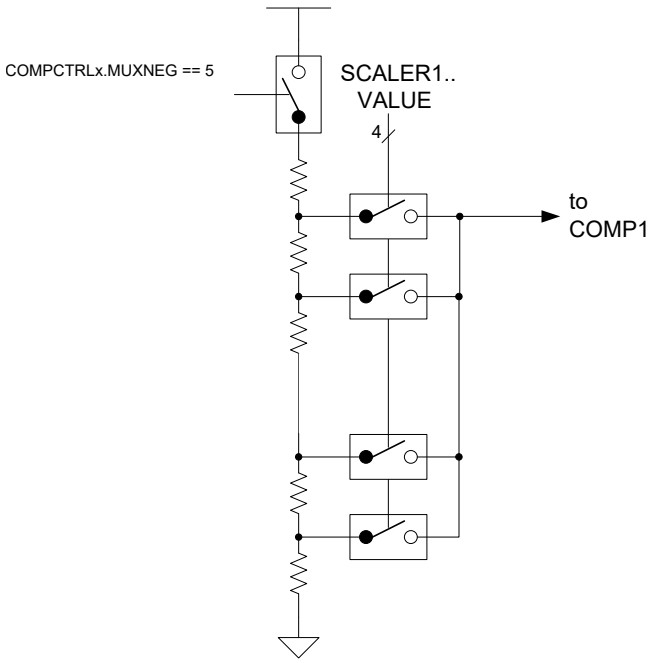
Figure 39-4. Comparators in Window Mode



39.5.5 VDD Scaler

The VDD scaler generates a reference voltage that is a fraction of the device's supply voltage with 64 levels. The programmable VDD scaler (PLVD Scaler) is available for AC_CMP1 only. AC_CMP0 uses a fixed VDD/2 reference. The scaler of a comparator is enabled when the Negative Input Mux bit field or the Positive Input Mux in the respective Comparator Control register (COMPCTRLx.MUXNEG) is set to `0x5` and the comparator is enabled. The voltage of each channel is selected by the Value bit field in the SCALER1 registers (SCALER1.VALUE) using the VDD resistor ladder.

Figure 39-5. PLVD Scaler



39.5.6 Filtering

The output of the comparators can be filtered digitally to reduce noise. The filtering is determined by the Filter Length bits in the Comparator Control x register (COMPCTRLx.FLEN), and is independent for each comparator. Filtering is selectable from none, 3-bit majority (N=3) or 5-bit majority (N=5) functions. Any change in the comparator output is considered valid only if N/2+1 out of the last N samples agree. The filter sampling rate is the GCLK_AC frequency.

Note that filtering creates an additional delay of N-1 sampling cycles from when a comparison is started until the comparator output is validated. For Continuous mode, the first valid output will occur when the required number of filter samples is taken. Subsequent outputs will be generated every cycle based on the current sample plus the previous N-1 samples, as shown in Figure 39-6. For Single-shot mode, the comparison completes after the Nth filter sample, as shown in Figure 39-7.

Figure 39-6. Continuous Mode Filtering

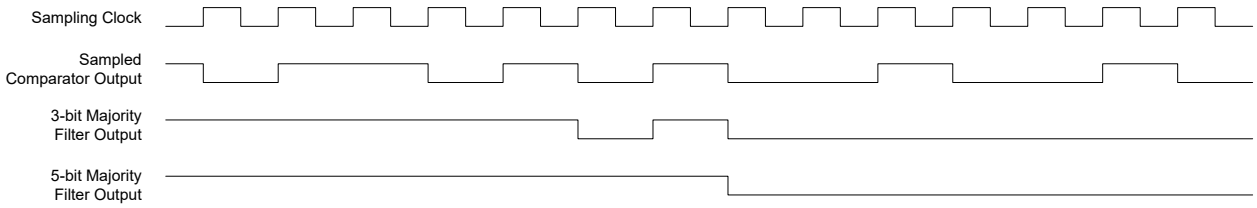
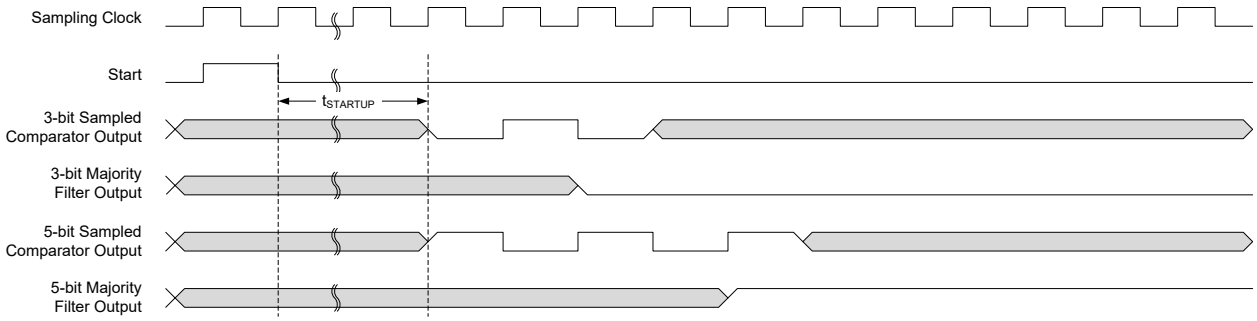


Figure 39-7. Single-Shot Filtering



During Sleep modes, filtering is supported only for single-shot measurements. Filtering must be disabled if continuous measurements will be done during Sleep modes, or the resulting interrupt/event may be generated incorrectly.

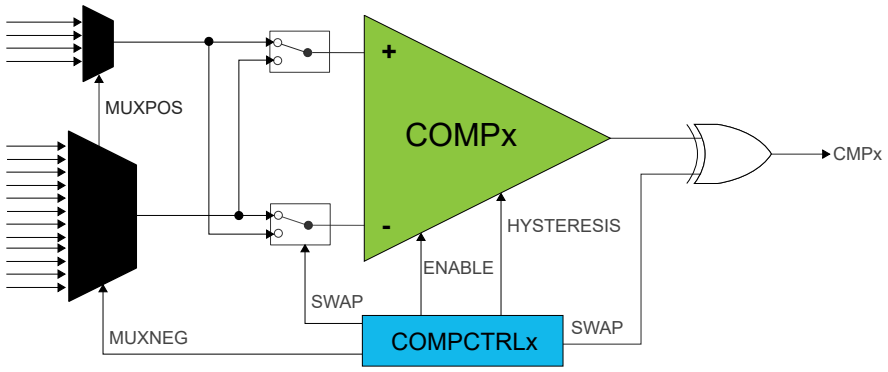
39.5.7 Comparator Output

The output of each comparator can be routed to an I/O pin by setting the Output bit group in the Comparator Control x register (COMPCTRLx.OUT). To get the analog comparator output on the I/O line, CFGCON1.CMP0_OE/ CFGCON1.CMP1_OE also needs to be set or enabled. This allows the comparator to be used by external circuitry. Either the raw, non-synchronized output of the comparator or the GCLK_AC-synchronized version, including filtering, can be used as the I/O signal source. The output appears on the corresponding AC_CMPx pin. The AC_CMP1 can be output on an alternate pin by configuring the CFGCON0.ACCMP1_ALTEN configuration.

39.5.8 Offset Compensation

The Swap bit in the Comparator Control registers (COMPCTRLx.SWAP) controls switching of the input signals to a comparator's positive and negative terminals. When the comparator terminals are swapped, the output signal from the comparator is also inverted, as shown in Figure 39-8. This allows the user to measure or compensate for the comparator input offset voltage. As part of the input selection, COMPCTRLx.SWAP can be changed only while the comparator is disabled.

Figure 39-8. Input Swapping for Offset Compensation



39.5.9 DMA Operation

Not applicable.

39.5.10 Interrupts

The AC has the following interrupt sources:

- Comparator (COMP0, COMP1): Indicates a change in comparator status
- Window (WIN0): Indicates a change in the window status

Comparator interrupts are generated based on the conditions selected by the Interrupt Selection bit group in the Comparator Control registers (COMPCTRLx.INTSEL). Window interrupts are generated based on the conditions selected by the Window Interrupt Selection bit group in the Window Control register (WINCTRL.WINTSEL[1:0]).

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the interrupt condition occurs. Each interrupt can be individually enabled by writing a one to the corresponding bit in the Interrupt Enable Set (INTENSET) register and disabled by writing a one to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register.

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the interrupt flag is cleared, the interrupt is disabled or the AC is Reset. See *INTFLAG* register from Related Links for details on how to clear interrupt flags. All interrupt requests from the peripheral are OR'ed together on the system level to generate one combined interrupt request to the NVIC. The user must read the INTFLAG register to determine which interrupt condition is present. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Note: Interrupts must be globally enabled for interrupt requests to be generated.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

39.7.6. INTFLAG

39.5.11 Events

The AC can generate the following output events:

- Comparator (COMP0, COMP1): Generated as a copy of the comparator status
- Window (WIN0): Generated as a copy of the window inside/outside status

Writing a one to an Event Output bit in the Event Control Register (EVCTRL.xxEO) enables the corresponding output event. Writing a zero to this bit disables the corresponding output event. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.

The AC can take the following action on an input event:

- Start comparison (START0, START1): Start a comparison

Writing a one to an Event Input bit into the Event Control register (EVCTRL.COMPEIx) enables the corresponding action on an input event. Writing a zero to this bit disables the corresponding action on an input event. Note that if several events are connected to the AC, the enabled action will be taken on any of the incoming events. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.

When EVCTRL.COMPEIx is one, the event will start a comparison on COMPx after the start-up time delay. In normal mode, each comparator responds to its corresponding input event independently. For a pair of comparators in window mode, either comparator event will trigger a comparison on both comparators simultaneously.

Related Links

28. Event System (EVSYS)

39.5.12 Sleep Mode Operation

The Run in Standby bits in the Comparator x Control registers (COMPCTRLx.RUNSTDBY) control the behavior of the AC during standby sleep mode. Each RUNSTDBY bit controls one comparator. When the bit is zero, the comparator is disabled during sleep, but maintains its current configuration. When the bit is one, the comparator continues to operate during sleep. Note that when RUNSTDBY is zero, the analog blocks are powered off for the lowest power consumption. This necessitates a start-up time delay when the system returns from sleep.

For Window Mode operation, both comparators in a pair must have the same RUNSTDBY configuration.

When RUNSTDBY is one, any enabled AC interrupt source can wake up the CPU. The AC can also be used during sleep modes where the clock used by the AC is disabled, provided that the AC is still powered (not in shutdown). In this case, the behavior is slightly different and depends on the measurement mode, as listed in Table 39-2.

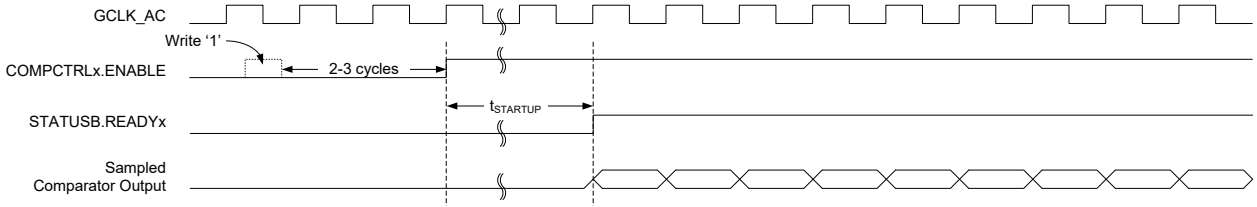
Table 39-2. Sleep Mode Operation

COMPCTRLx.MODE	RUNSTDBY=0	RUNSTDBY=1
0 (Continuous)	COMPx disabled	GCLK_AC stopped, COMPx enabled
1 (Single-shot)	COMPx disabled	GCLK_AC stopped, COMPx enabled only when triggered by an input event

39.5.12.1 Continuous Measurement during Sleep

When a comparator is enabled in continuous measurement mode and GCLK_AC is disabled during sleep, the comparator will remain continuously enabled and will function asynchronously. The current state of the comparator is asynchronously monitored for changes. If an edge matching the interrupt condition is found, GCLK_AC is started to register the interrupt condition and generate events. If the interrupt is enabled in the Interrupt Enable registers (INTENCLR/SET), the AC can wake up the device; otherwise GCLK_AC is disabled until the next edge detection. Filtering is not possible with this configuration.

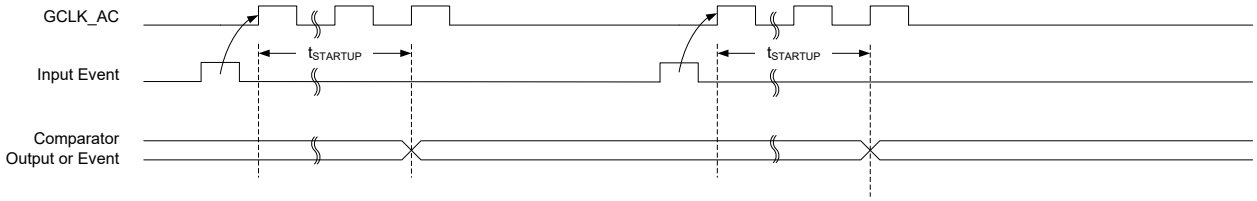
Figure 39-9. Continuous Mode SleepWalking



39.5.12.2 Single-Shot Measurement during Sleep

For low-power operation, event-triggered measurements can be performed during sleep modes. When the event occurs, the CRU will start GCLK_AC. The comparator is enabled and, after the start-up time has passed, a comparison is done, with filtering if desired, and the appropriate peripheral events and interrupts are also generated as shown in the following figure. The comparator and GCLK_AC are, then, disabled again automatically unless configured to wake the system from sleep. Filtering is allowed with this configuration.

Figure 39-10. Single-Shot SleepWalking



39.5.13 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset bit in Control register (CTRLA.SWRST)
- Enable bit in Control register (CTRLA.ENABLE)
- Enable bit in Comparator Control register (COMPCTRLn.ENABLE)

The following registers are synchronized when written:

- Window Control register (WINCTRL)

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

39.6 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0							ENABLE	SWRST
0x01	CTRLB	7:0								
0x02	EVCTRL	7:0								
		15:8								
0x04	INTENCLR	7:0								
0x05	INTENSET	7:0								
0x06	INTFLAG	7:0								
0x07	STATUSA	7:0			WSTATE0[1:0]					
0x08	STATUSB	7:0								
0x09	DBGCTRL	7:0								DBGRUN
0x0A	WINCTRL	7:0						WINTSEL0[1:0]		WEN0
0x0B ... 0x0C	Reserved									
0x0D	SCALER1	7:0					VALUE[3:0]			
0x0E ... 0x0F	Reserved									
0x10	COMPCTRL0	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
		15:8	SWAP	MUXPOS[2:0]				MUXNEG[2:0]		
		23:16								
		31:24			OUT[1:0]			FLEN[2:0]		
0x14	COMPCTRL1	7:0		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
		15:8	SWAP	MUXPOS[2:0]				MUXNEG[2:0]		
		23:16								
		31:24			OUT[1:0]			FLEN[2:0]		
0x18 ... 0x1F	Reserved									
0x20	SYNCBUSY	7:0						WINCTRL	ENABLE	SWRST
		15:8								
		23:16								
		31:24								

39.7 Register Description

Registers can be 8, 16, or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write-protection is denoted by the “PAC Write-Protection” property in each individual register description. See *Register Access Protection* from Related Links.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description. See *Synchronization* from Related Links.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable-protection is denoted by the "Enable-Protected" property in each individual register description.

Related Links

39.4.8. Register Access Protection

39.5.13. Synchronization

39.7.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
							ENABLE	SWRST
Access							R/W	W
Reset							0	0

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from updating the register until the peripheral is enabled/disabled. The value written to CTRL.ENABLE will read back immediately and the corresponding bit in the Synchronization Busy register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE is cleared when the peripheral is enabled/disabled.

Value	Description
0	The AC is disabled.
1	The AC is enabled. Each comparator must also be enabled individually by the Enable bit in the Comparator Control register (COMPCTRLn.ENABLE).

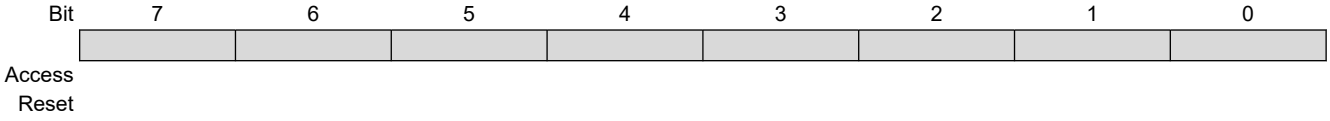
Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the AC to their initial state, and the AC will be disabled.
Writing a '1' to CTRLA.SWRST will always take precedence, meaning that all other writes in the same write-operation will be discarded.
Due to synchronization, there is a delay from writing CTRLA.SWRST until the reset is complete. CTRLA.SWRST and SYNCBUSY.SWRST will both be cleared when the reset is complete.

Value	Description
0	There is no reset operation ongoing.
1	The reset operation is ongoing.

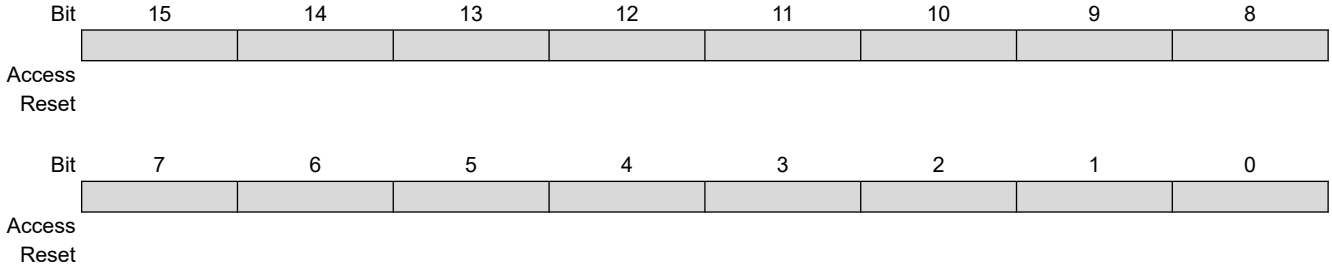
39.7.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -



39.7.3 Event Control

Name: EVCTRL
Offset: 0x02
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected



39.7.4 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

39.7.5 Interrupt Enable Set

Name: INTENSET
Offset: 0x05
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

39.7.6 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x06
Reset: 0x00
Property: –

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

39.7.7 Status A

Name: STATUSA
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			WSTATE0[1:0]					
Access			R	R				
Reset			0	0				

Bits 5:4 – WSTATE0[1:0] Window 0 Current State
These bits show the current state of the signal if the window 0 mode is enabled.
These values may change in during startup and measurement cycles. When polling for sample completion use both STATUSB.READYx bits to signal completion.

Value	Name	Description
0x0	ABOVE	Signal is above window
0x1	INSIDE	Signal is inside window
0x2	BELOW	Signal is below window
0x3	-	Reserved

39.7.8 Status B

Name: STATUSB
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

39.7.9 Debug Control

Name: DBGCTRL
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Debug Run

This bit is not reset by a software reset.

This bits controls the functionality when the CPU is halted by an external debugger.

Value	Description
0	The AC is halted when the CPU is halted by an external debugger. Any on-going comparison will complete.
1	The AC continues normal operation when the CPU is halted by an external debugger.

39.7.10 Window Control

Name: WINCTRL
Offset: 0x0A
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

Bit	7	6	5	4	3	2	1	0
						WINTSEL0[1:0]		WEN0
Access						R/W	R/W	R/W
Reset						0	0	0

Bits 2:1 – WINTSEL0[1:0] Window 0 Interrupt Selection
These bits configure the interrupt mode for the comparator window 0 mode.

Value	Name	Description
0x0	ABOVE	Interrupt on signal above window
0x1	INSIDE	Interrupt on signal inside window
0x2	BELOW	Interrupt on signal below window
0x3	OUTSIDE	Interrupt on signal outside window

Bit 0 – WEN0 Window 0 Mode Enable

Value	Description
0	Window mode is disabled for comparators 0 and 1.
1	Window mode is enabled for comparators 0 and 1.

39.7.11 Scaler 1

Name: SCALER1
Offset: 0x0D
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
					VALUE[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – VALUE[3:0] Scaler Value
These bits define the scaling factor for channel 1 of the VDD voltage scaler. The output voltage, V_{SCALE}, is:

$$V_{\mathrm{SCALE}} = V_{DD} \times \left(\frac{R\_Bottom}{R\_Total}\right)$$

Where, R_Total = 900.
Refer to the following table for R_Bottom for different VALUE[3:0]
For example, V_{SCALE} for VALUE[3:0] = 0x02 at VDD = 3.3V

$$V_{\mathrm{SCALE}} = 3.3 \times \left(\frac{598.5}{900}\right) = 2.1945V$$

Table 39-3. Scaler Value

Value[3:0]	R_Bottom
0x0	Reserved
0x1	Reserved
0x2	598.5
0x3	634.5
0x4	306
0x5	324
0x6	328.5
0x7	360
0x8	387
0x9	400.5
0xA	432
0xB	450
0xC	468
0xD	490.5
0xE	481.5
0xF	External Reference on LVDIN pin

39.7.12 Comparator Control n

Name: COMPCTRL
Offset: 0x10 + n*0x04 [n=0..1]
Reset: 0x00000000
Property: PAC Write-Protection

Bit	31	30	29	28	27	26	25	24
			OUT[1:0]			FLEN[2:0]		
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SWAP	MUXPOS[2:0]				MUXNEG[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY		INTSEL[1:0]		SINGLE	ENABLE	
Access		R/W		R/W	R/W	R/W	R/W	
Reset		0		0	0	0	0	

Bits 29:28 – OUT[1:0] Output
These bits configure the output selection for comparator n. COMPCTRLn.OUT can be written only while COMPCTRLn.ENABLE is zero.
Note: For internal use of the comparison results by the CCL, this must be 0x1 or 0x2.

These bits are not synchronized.

Value	Name	Description
0x0	OFF	The output of COMPn is not routed to the COMPn I/O port
0x1	ASYNC	The asynchronous output of COMPn is routed to the COMPn I/O port
0x2	SYNC	The synchronous output (including filtering) of COMPn is routed to the COMPn I/O port
0x3	N/A	Reserved

Bits 26:24 – FLEN[2:0] Filter Length
These bits configure the filtering for comparator n. COMPCTRLn.FLEN can only be written while COMPCTRLn.ENABLE is zero.
These bits are not synchronized.

Value	Name	Description
0x0	OFF	No filtering
0x1	MAJ3	3-bit majority function (2 of 3)
0x2	MAJ5	5-bit majority function (3 of 5)
0x3-0x7	N/A	Reserved

Bit 15 – SWAP Swap Inputs and Invert
This bit swaps the positive and negative inputs to COMPn and inverts the output. This function can be used for offset cancellation. COMPCTRLn.SWAP can be written only while COMPCTRLn.ENABLE is zero.
These bits are not synchronized.

Value	Description
0	The output of MUXPOS connects to the positive input, and the output of MUXNEG connects to the negative input.

Value	Description
1	The output of MUXNEG connects to the positive input, and the output of MUXPOS connects to the negative input.

Bits 14:12 – MUXPOS[2:0] Positive Input Mux Selection

These bits select which input will be connected to the positive input of comparator n. COMPCTRLn.MUXPOS can be written only while COMPCTRLn.ENABLE is zero.
These bits are not synchronized.

Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3
0x4	VSCALE	VDD scaler
0x5-0x7	-	Reserved

Bits 10:8 – MUXNEG[2:0] Negative Input Mux Selection

These bits select which input will be connected to the negative input of comparator n. COMPCTRLn.MUXNEG can only be written while COMPCTRLn.ENABLE is zero.
These bits are not synchronized.

Value	Name	Description
0x0	PIN0	I/O pin 0
0x1	PIN1	I/O pin 1
0x2	PIN2	I/O pin 2
0x3	PIN3	I/O pin 3
0x4	GND	Ground
0x5	VSCALE	VDD scaler
0x6	BANDGAP	Internal bandgap voltage

Bit 6 – RUNSTDBY Run in Standby

This bit controls the behavior of the comparator during standby sleep mode.
This bit is not synchronized

Value	Description
0	The comparator is disabled during sleep.
1	The comparator continues to operate during sleep.

Bits 4:3 – INTSEL[1:0] Interrupt Selection

These bits select the condition for comparator n to generate an interrupt or event. COMPCTRLn.INTSEL can be written only while COMPCTRLn.ENABLE is zero.
These bits are not synchronized.

Value	Name	Description
0x0	TOGGLE	Interrupt on comparator output toggle
0x1	RISING	Interrupt on comparator output rising
0x2	FALLING	Interrupt on comparator output falling
0x3	EOC	Interrupt on end of comparison (single-shot mode only)

Bit 2 – SINGLE Single-Shot Mode

This bit determines the operation of comparator n. COMPCTRLn.SINGLE can be written only while COMPCTRLn.ENABLE is zero.
These bits are not synchronized.

Value	Description
0	Comparator n operates in continuous measurement mode.
1	Comparator n operates in single-shot mode.

Bit 1 – ENABLE Enable

Writing a zero to this bit disables comparator n.
Writing a one to this bit enables comparator n.

Due to synchronization, there is a delay from updating the register until the comparator is enabled/disabled. The value written to COMPCTRLn.ENABLE will read back immediately after being written. SYNCBUSY.COMPCTRLn is set. SYNCBUSY.COMPCTRLn is cleared when the peripheral is enabled/disabled.
Writing a one to COMPCTRLn.ENABLE will prevent further changes to the other bits in COMPCTRLn. These bits remain protected until COMPCTRLn.ENABLE is written to zero and the write is synchronized.

39.7.13 Synchronization Busy

Name: SYNCBUSY
Offset: 0x20
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						WINCTRL	ENABLE	SWRST
Access						R	R	R
Reset						0	0	0

Bit 2 – WINCTRL WINCTRL Synchronization Busy
This bit is cleared when the synchronization of the WINCTRL register between the clock domains is complete.
This bit is set when the synchronization of the WINCTRL register between clock domains is started.

Bit 1 – ENABLE Enable Synchronization Busy
This bit is cleared when the synchronization of the CTRLA.ENABLE bit between the clock domains is complete.
This bit is set when the synchronization of the CTRLA.ENABLE bit between clock domains is started.

Bit 0 – SWRST Software Reset Synchronization Busy
This bit is cleared when the synchronization of the CTRLA.SWRST bit between the clock domains is complete.
This bit is set when the synchronization of the CTRLA.SWRST bit between clock domains is started.

40. Timer/Counter (TC)

40.1 Overview

There are up to TC peripheral instances.

Each TC consists of a counter, a prescaler, compare/capture channels and control logic. The counter can be set to count events, or clock pulses. The counter, together with the compare/capture channels, can be configured to time stamp input events or IO pin edges, allowing for capturing of frequency and/or pulse width.

A TC can also perform waveform generation, such as frequency generation and pulse-width modulation.

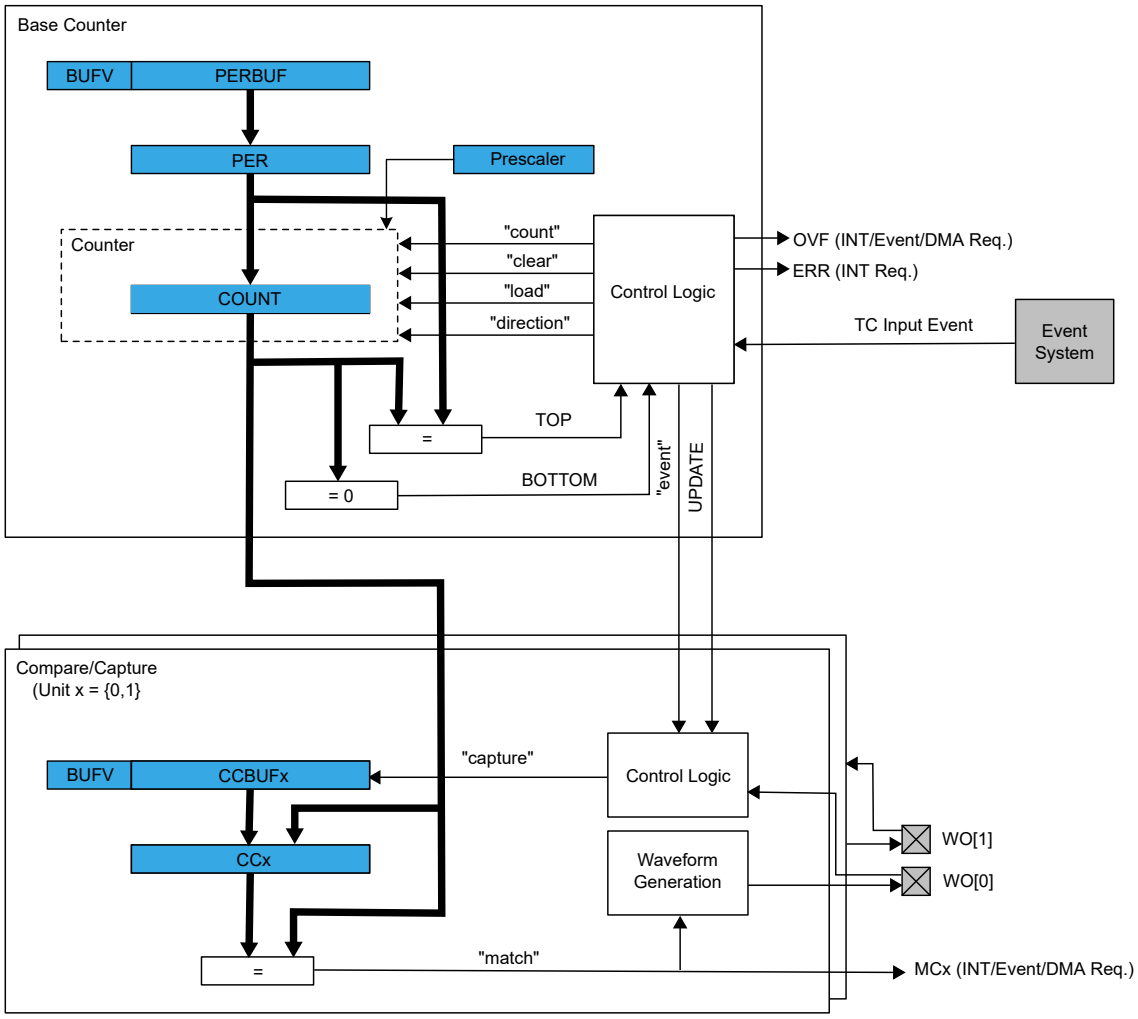
Note: Traditional Timer/Counter (TC) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

40.2 Features

- Selectable Configuration
  - 8-, 16- or 32-bit TC operation, with compare/capture channels
- 2 Compare/Capture Channels (CC) with:
  - Double buffered timer period setting (in 8-bit mode only)
  - Double buffered compare channel
- Waveform Generation
  - Frequency generation
  - Single-slope pulse-width modulation
- Input Capture
  - Event / IO pin edge capture
  - Frequency capture
  - Pulse-width capture
  - Time-stamp capture
  - Minimum and maximum capture
- One Input Event
- Interrupts/Output Events ON:
  - Counter overflow/underflow
  - Compare match or capture
- Internal Prescaler
- DMA Support

40.3 Block Diagram

Figure 40-1. Timer/Counter Block Diagram



40.4 Signal Description

Table 40-1. Signal Description for TC

Signal Name	Type	Description
WO[1:0]	Digital output	Waveform output
	Digital input	Capture input

See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

40.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

40.5.1 I/O Lines

In order to use the I/O lines of this peripheral, the I/O pins must be configured using the I/O Peripheral Pin Select (PPS).

40.5.2 Power Management

This peripheral can continue to operate in any Sleep mode where its source clock is running. The interrupts can wake up the device from Sleep modes. Events connected to the event system can trigger other operations in the system without exiting Sleep modes.

40.5.3 Clocks

The TC bus clocks (PB1_CLK) can be enabled and disabled in the PMD Registers. For more details and default status of this clock, see *Peripheral Module Disable Register (PMD)* from Related Links.

The generic clocks (GCLK_TCx) are asynchronous to the user interface clock (PB1_CLK). Due to this asynchronicity, accessing certain registers will require synchronization between the clock domains.

Note: Two instances of the TC may share a peripheral clock channel. In this case, they cannot be set to different clock frequencies. See *System and Peripheral Clock Generation (CLKGEN)* from Related Links to identify shared peripheral clocks.

Related Links

13.4. System and Peripheral Clock Generation (CLKGEN)

20. Peripheral Module Disable Register (PMD)

40.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). In order to use DMA requests with this peripheral, the DMAC must be configured first (see *Direct Memory Access Controller (DMAC)* from Related Links).

Related Links

22. Direct Memory Access Controller (DMAC)

40.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

40.5.6 Events

The events of this peripheral are connected to the Event System.

Related Links

28. Event System (EVSYS)

40.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will halt normal operation. This peripheral can be forced to continue operation during debugging. For more details, see *DBGCTRL* from Related Links.

Related Links

40.7.2.11. DBGCTRL

40.5.8 Register Access Protection

Registers with write access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag Status and Clear register (INTFLAG)
- Status register (STATUS)
- Period and Period Buffer registers (PER, PERBUF)
- Compare/Capture Value registers and Compare/Capture Value Buffer registers (CCx, CCBUFx)

Note: Optional write protection is indicated by the "PAC Write Protection" property in the register description.

Write protection does not apply for accesses through an external debugger.

40.5.9 Analog Connections

Not applicable.

40.6 Functional Description

40.6.1 Principle of Operation

The following definitions are used throughout the documentation:

Table 40-2. Timer/Counter Definitions

Name	Description
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be the same as Period (PER) or the Compare Channel 0 (CC0) register value depending on the waveform generator mode in Waveform Output Operations. See *Waveform Output Operations* from Related Links.
ZERO	The counter is ZERO when it contains all zeros.
MAX	The counter reaches MAX when it contains all ones.
UPDATE	The timer/counter signals an update when it reaches ZERO or TOP, depending on the direction settings.
Timer	The timer/counter clock control is handled by an internal source.
Counter	The clock control is handled externally (e.g., counting external events).
CC	For compare operations, the CC are referred to as “compare channels.” For capture operations, the CC are referred to as “capture channels.”

Each TC instance has up to two compare/capture channels (CC0 and CC1).

The counter in the TC can either count events from the Event System or clock ticks of the GCLK_TCx clock, which may be divided by the prescaler.

The counter value is passed to the CCx where it can be either compared to user-defined values or captured.

For optimized timing, the CCx and CCBUFx registers share a common resource. When writing into CCBUFx, lock the access to the corresponding CCx register (SYNCBUSY.CCX = 1) until the CCBUFx register value is not loaded into the CCx register (BUFVx == 1). Each buffer register has a buffer valid (BUFV) flag that indicates when the buffer contains a new value.

The Counter register (COUNT) and the Compare and Capture registers with buffers (CCx and CCBUFx) can be configured as 8-, 16- or 32-bit registers, with corresponding MAX values. Mode settings (CTRLA.MODE) determine the maximum range of the Counter register.

In 8-bit mode, a Period Value (PER) register and its Period Buffer Value (PERBUF) register are also available. The counter range and the operating frequency determine the maximum time resolution achievable with the TC peripheral.

The TC can be set to count up or down. Under normal operation, the counter value is continuously compared to the TOP or ZERO value to determine whether the counter has reached that value. On a comparison match, the TC can request DMA transactions, or generate interrupts or events for the Event System.

In a compare operation, the counter value is continuously compared to the values in the CCx registers. In the case of a match, the TC can request DMA transactions, or generate interrupts or events for the Event System. In waveform generator mode, these comparisons are used to set the waveform period or pulse width.

Capture operation can be enabled to perform input signal period and pulse width measurements, or to capture selectable edges from an IO pin or internal event from Event System.

Related Links

40.6.2.6.1. Waveform Output Operations

40.6.2 Basic Operation

40.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the TC is disabled (CTRLA.ENABLE=0):

- Control A register (CTRLA), except the Enable (ENABLE) and Software Reset (SWRST) bits
- Drive Control register (DRVCTRL)
- Wave register (WAVE)
- Event Control register (EVCTRL)

Writing to enable-protected bits and setting the CTRLA.ENABLE bit can be performed in a single 32-bit access of the CTRLA register. Writing to enable-protected bits and clearing the CTRLA.ENABLE bit cannot be performed in a single 32-bit access.

Before enabling the TC, the peripheral must be configured by the following steps:

1. Enable the TC bus clock if not already enabled by default (PB1_CLK).
2. Select 8-, 16- or 32-bit counter mode via the TC Mode bit group in the Control A register (CTRLA.MODE). The default mode is 16-bit.
3. Select one wave generation operation in the Waveform Generation Operation bit group in the WAVE register (WAVE.WAVEGEN).
4. If desired, the GCLK_TCx clock can be prescaled via the Prescaler bit group in the Control A register (CTRLA.PRESCALER).
   - If the prescaler is used, select a prescaler synchronization operation via the Prescaler and Counter Synchronization bit group in the Control A register (CTRLA.PRESYNC).
5. If desired, select one-shot operation by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT).
6. If desired, configure the counting direction 'down' (starting from the TOP value) by writing a '1' to the Counter Direction bit in the Control B register (CTRLBSET.DIR).
7. For capture operation, enable the individual channels to capture in the Capture Channel x Enable bit group in the Control A register (CTRLA.CAPTEN).
8. If desired, enable inversion of the waveform output or IO pin input signal for individual channels via the Invert Enable bit group in the Drive Control register (DRVCTRL.INVEN).

40.6.2.2 Enabling, Disabling, and Resetting

The TC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The TC is disabled by writing a zero to CTRLA.ENABLE.

The TC is reset by writing a '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the TC, except DBGCTRL, will be reset to their initial state. See *CTRLA* from Related Links.

The TC must be disabled before the TC is reset in order to avoid undefined behavior.

Related Links
40.7.2.1. CTRLA

40.6.2.3 Prescaler Selection

The GCLK_TCx is fed into the internal prescaler.

The prescaler consists of a counter that counts up to the selected prescaler value, whereupon the output of the prescaler toggles.

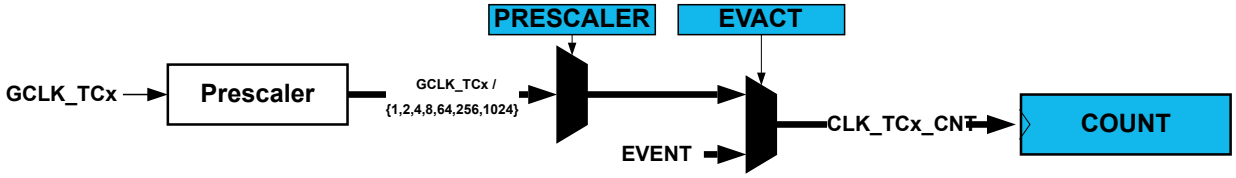
If the prescaler value is higher than one, the Counter Update condition can be optionally executed on the next GCLK_TCx clock pulse or the next prescaled clock pulse. For further details, refer to Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) description.

Prescaler outputs from 1 to 1/1024 are available. For a complete list of available prescaler outputs, see the register description for the Prescaler bit group in the Control A register (CTRLA.PRESCALER).

Note: When counting events, the prescaler is bypassed.

The joint stream of prescaler ticks and event action ticks is called CLK_TCx_CNT.

Figure 40-2. Prescaler



40.6.2.4 Counter Mode

The counter mode is selected by the Mode bit group in the Control A register (CTRLA.MODE). By default, the counter is enabled in the 16-bit counter resolution. Three counter resolutions are available:

- COUNT8: The 8-bit TC has its own Period Value and Period Buffer Value registers (PER and PERBUF).
- COUNT16: 16-bit is the default counter mode. There is no dedicated period register in this mode.
- COUNT32: 32-bit mode is achieved by pairing two 16-bit TC peripherals. TC(2n) is paired with TC(n+1). When paired, the TC peripherals are configured using the registers of the even-numbered TC (TC0 or TC2, respectively).

  The TC bus clocks (PB1_CLK) for both host and client TCs need to be enabled.

  The odd-numbered partner (TC1 or TC3, respectively) will act as a client, and the Client bit in the Status register (STATUS.SLAVE) will be set. The register values of a client will not reflect the registers of the 32-bit counter. Writing to any of the client registers will not affect the 32-bit counter. Normal access to the client COUNT and CCx registers is not allowed.

40.6.2.5 Counter Operations

Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each TC clock input (CLK_TCx_CNT). A counter clear or reload marks the end of the current counter cycle and the start of a new one.

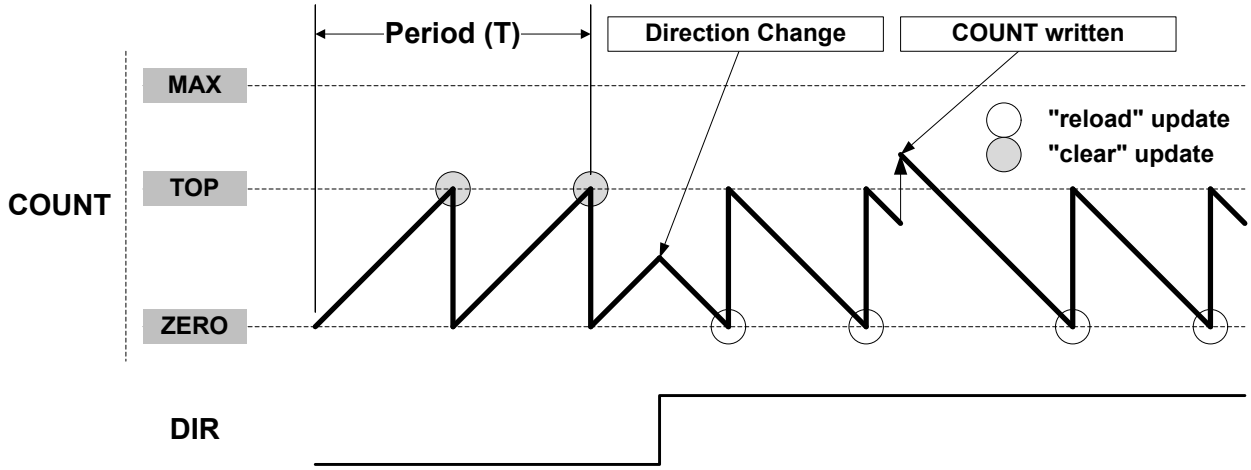
The counting direction is set by the Direction bit in the Control B register (CTRLB.DIR). If this bit is zero the counter is counting up, and counting down if CTRLB.DIR=1. The counter will count up or down for each tick (clock or event) until it reaches TOP or ZERO. When it is counting up and TOP is reached, the counter will be set to zero at the next tick (overflow) and the Overflow Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF) will be set. When it is counting down, the counter is reloaded with the TOP value when ZERO is reached (underflow), and INTFLAG.OVF is set.

INTFLAG.OVF can be used to trigger an interrupt, a DMA request, or an event. An overflow/underflow occurrence (i.e., a compare match with TOP/ZERO) will stop counting if the One-Shot bit in the Control B register is set (CTRLBSET.ONESHOT).

It is possible to change the counter value (by writing directly in the COUNT register) even when the counter is running. When starting the TC, the COUNT value will be either ZERO or TOP (depending on the counting direction set by CTRLBSET.DIR or CTRLBCLR.DIR), unless a different value has been written to it, or the TC has been

stopped at a value other than ZERO. The write access has higher priority than count, clear, or reload. The direction of the counter can also be changed when the counter is running. See also the following figure.

Figure 40-3. Counter Operation



Due to asynchronous clock domains, the internal counter settings are written when the synchronization is complete. Normal operation must be used when using the counter as timer base for the capture channels.

40.6.2.5.1 Stop Command and Event Action

A Stop command can be issued from software by using Command bits in the Control B Set register (CTRLBSET.CMD = 0x2, STOP). When a Stop is detected while the counter is running, the counter will not retain its current value. All waveforms are cleared and the Stop bit in the Status register is set (STATUS.STOP).

40.6.2.5.2 Re-Trigger Command and Event Action

A re-trigger command can be issued from software by writing the Command bits in the Control B Set register (CTRLBSET.CMD = 0x1, RETRIGGER), or from event when a re-trigger event action is configured in the Event Control register (EVCTRL.EVACT = 0x1, RETRIGGER).

When the command is detected during counting operation, the counter will be reloaded or cleared, depending on the counting direction (CTRLBSET.DIR or CTRLBCLR.DIR). When the re-trigger command is detected while the counter is stopped, the counter will resume counting from the current value in the COUNT register.

Note: When a re-trigger event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACT=0x1, RETRIGGER), enabling the counter will not start the counter. The counter will start on the next incoming event and restart on corresponding following event.

40.6.2.5.3 Count Event Action

The TC can count events. When an event is received, the counter increases or decreases the value, depending on direction settings (CTRLBSET.DIR or CTRLBCLR.DIR). The count event action can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT=0x2, COUNT).

Note: If this operation mode is selected, PWM generation is not supported.

40.6.2.5.4 Start Event Action

The TC can start counting operation on an event when previously stopped. In this configuration, the event has no effect if the counter is already counting. When the peripheral is enabled, the counter operation starts when the event is received or when a re-trigger software command is applied.

The Start TC on Event action can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT=0x3, START).

40.6.2.6 Compare Operations

By default, the Compare/Capture channel is configured for compare operations.

When using the TC and the Compare/Capture Value registers (CCx) for compare operations, the counter value is continuously compared to the values in the CCx registers. This can be used for timer or for waveform operation.

The Channel x Compare Buffer (CCBUFx) registers provide double buffer capability. The double buffering synchronizes the update of the CCx register with the buffer value at the UPDATE condition or a forced update command (CTRLBSET.CMD=UPDATE). See *Double Buffering* from Related Links. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses and ensures glitch-free output.

Related Links

40.6.2.7. Double Buffering

40.6.2.6.1 Waveform Output Operations

The compare channels can be used for waveform generation on output port pins. To make the waveform available on the connected pin, the following requirements must be fulfilled:

1. Choose a Waveform Generation mode in the Waveform Generation Operation bit in Waveform register (WAVE.WAVEGEN).
2. Optionally invert the waveform output WO[x] by writing the corresponding Output Waveform x Invert Enable bit in the Driver Control register (DRVCTRL.INVENx).
3. Configure the pins with the I/O Peripheral Pin Select (PPS). See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.
   Note: Event must not be used when the compare channel is set in waveform output operating mode.

The counter value is continuously compared with each CCx value. On a comparison match, the Match or Capture Channel x bit in the Interrupt Flag Status and Clear register (INTFLAG.MCx) will be set on the next zero-to-one transition of CLK_TC_CNT (see Normal Frequency Operation). An interrupt/and or event can be generated on comparison match if enabled. The same condition generates a DMA request.

There are four waveform configurations for the Waveform Generation Operation bit group in the Waveform register (WAVE.WAVEGEN). This will influence how the waveform is generated and impose restrictions on the top value. The configurations are:

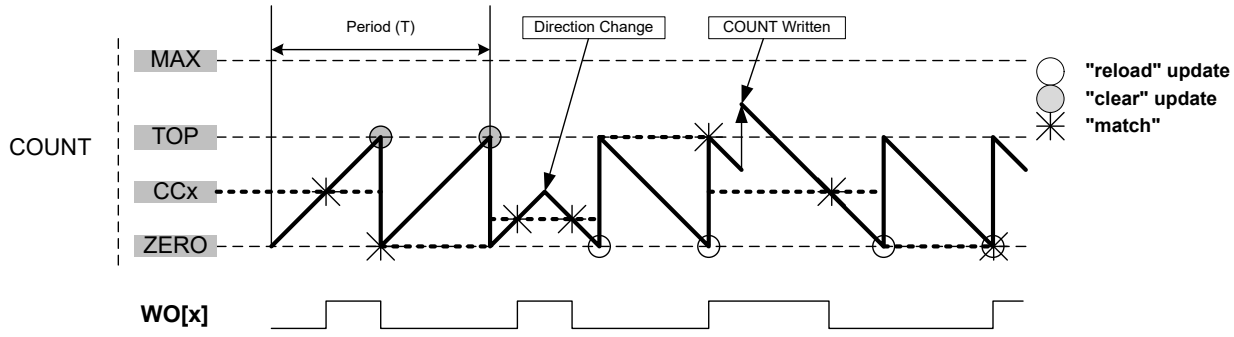
- Normal frequency (NFRQ)
- Match frequency (MFRQ)
- Normal pulse-width modulation (NPWM)
- Match pulse-width modulation (MPWM)

When using NPWM or NFRQ configuration, the TOP will be determined by the counter resolution. In 8-bit Counter mode, the Period register (PER) is used as TOP, and the TOP can be changed by writing to the PER register. In 16- and 32-bit Counter mode, TOP is fixed to the maximum (MAX) value of the counter.

Normal Frequency Generation (NFRQ)

For Normal Frequency Generation, the period time (T) is controlled by the period register (PER) for 8-bit Counter mode and MAX for 16- and 32-bit mode. The waveform generation output (WO[x]) is toggled on each compare match between COUNT and CCx, and the corresponding Match or Capture Channel x Interrupt Flag (INTFLAG.MCx) will be set.

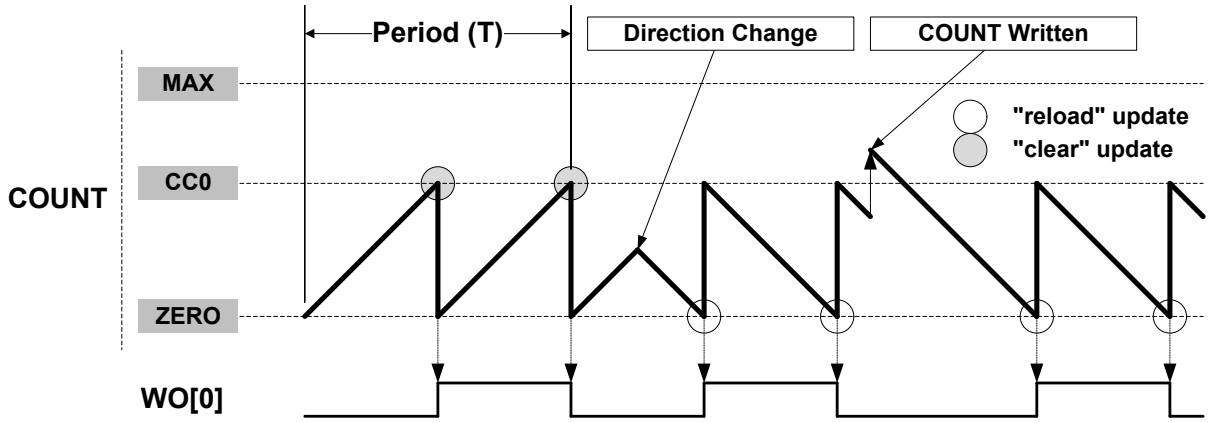
Figure 40-4. Normal Frequency Operation



Match Frequency Generation (MFRQ)

For Match Frequency Generation, the period time (T) is controlled by the CC0 register instead of PER or MAX. WO[0] toggles on each Update condition.

Figure 40-5. Match Frequency Operation



Normal Pulse-Width Modulation Operation (NPWM)
NPWM uses single-slope PWM generation.

For single-slope PWM generation, the period time (T) is controlled by the TOP value, and CCx controls the duty cycle of the generated waveform output. When up-counting, the WO[x] is set at start or compare match between the COUNT and TOP values, and cleared on compare match between COUNT and CCx register values. When down-counting, the WO[x] is cleared at start or compare match between the COUNT and ZERO values, and set on compare match between COUNT and CCx register values.

The following equation calculates the exact resolution for a single-slope PWM (R_{PWM_SS}) waveform:

$$R_{PWM\_SS} = \frac{\log(TOP+1)}{\log(2)}$$

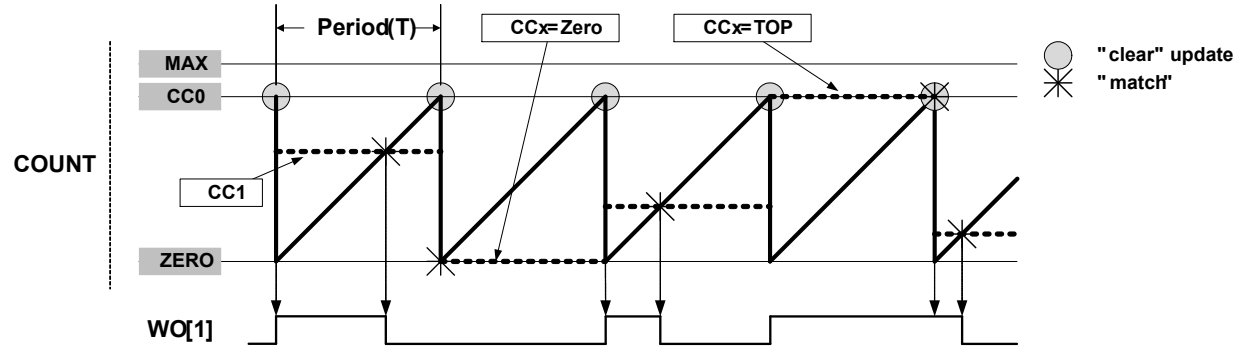
The PWM frequency (f_{PWM_SS}) depends on TOP value and the peripheral clock frequency (f_{GCLK_TC}), and can be calculated by the following equation:

$$f_{PWM\_SS} = \frac{f_{GCLK\_TC}}{N(TOP+1)}$$

Where N represents the prescaler divider used (1, 2, 4, 8, 16, 64, 256, 1024).

Match Pulse-Width Modulation Operation (MPWM)
In MPWM, the output of WO[1] is depending on CC1 as shown in the figure below. On every overflow/underflow, a one-TC-clock-cycle negative pulse is put out on WO[0] (not shown in the figure).

Figure 40-6. Match PWM Operation



The following table shows the Update Counter and Overflow Event/Interrupt Generation conditions in different operation modes.

Table 40-3. Counter Update and Overflow Event/interrupt Conditions in TC

Name	Operation	TOP	Update	Output Waveform		OVFIF/Event	
				On Match	On Update	Up	Down
NFRQ	Normal Frequency	PER	TOP/ ZERO	Toggle	Stable	TOP	ZERO
MFRQ	Match Frequency	CC0	TOP/ ZERO	Toggle	Stable	TOP	ZERO
NPWM	Single-slope PWM	PER	TOP/ ZERO	See description above.		TOP	ZERO
MPWM	Single-slope PWM	CC0	TOP/ ZERO	Toggle	Toggle	TOP	ZERO

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

40.6.2.7 Double Buffering

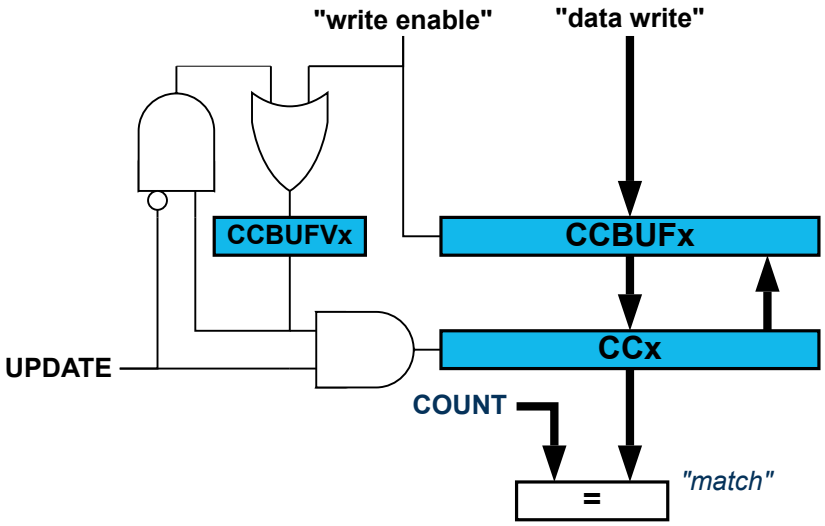
The Compare Channels (CCx) registers, and the Period (PER) register in 8-bit mode are double buffered. Each buffer register has a buffer valid bit (CCBUFVx or PERBUFV) in the STATUS register, which indicates that the buffer register contains a new valid value that can be copied into the corresponding register. As long as the respective buffer valid status flag (PERBUFV or CCBUFVx) are set to '1', related syncbusy bits are set (SYNCBUSY.PER or SYNCBUSY.CCx), a write to the respective PER/PERBUF or CCx/CCBUFx registers will generate a PAC error, and access to the respective PER or CCx register is invalid.

When the buffer valid flag bit in the STATUS register is '1' and the Lock Update bit in the CTRLB register is set to '0', (writing CTRLBCLR.LUPD to '1'), double buffering is enabled: the data from buffer registers will be copied into the corresponding register under hardware UPDATE conditions, then the buffer valid flags bit in the STATUS register are automatically cleared by hardware.

Note: The software update command (CTRLBSET.CMD=0x3) is acting independently of the LUPD value.

A compare register is double buffered as in the following figure.

Figure 40-7. Compare Channel Double Buffering



Both the registers (PER/CCx) and corresponding buffer registers (PERBUF/CCBUFx) are available in the I/O register map, and the double buffering feature is not mandatory. The double buffering is disabled by writing a '1' to CTRLBSET.LUPD.

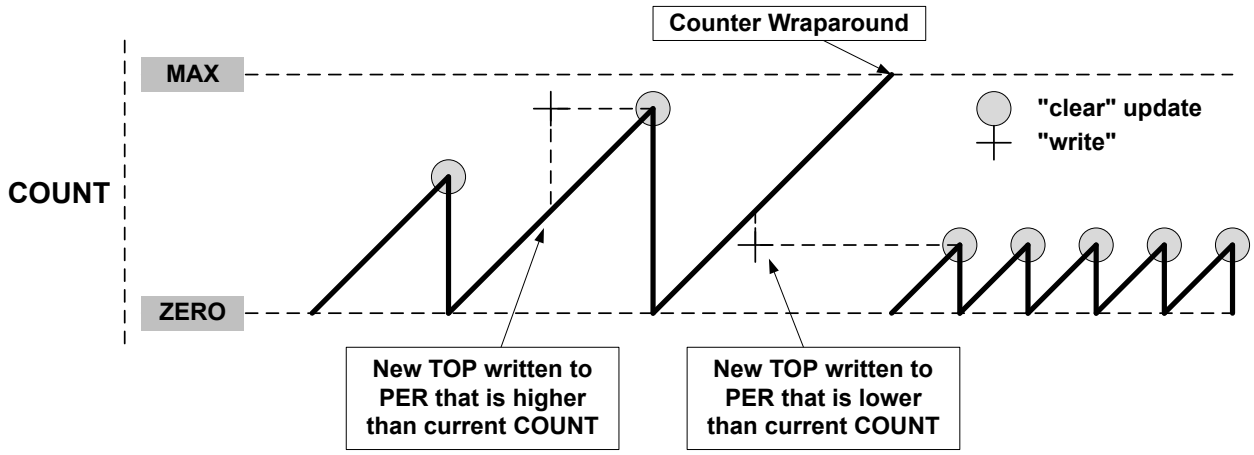
Note: In NFRQ, MFRQ or PWM, down-counting counter mode (CTRLBSET.DIR=1), when double buffering is enabled (CTRLBCLR.LUPD=1), PERBUF register is continuously copied into the PER independently of update conditions.

Changing the Period

The counter period can be changed by writing a new TOP value to the Period register (PER or CC0, depending on the waveform generation mode), which is available in 8-bit mode. Any period update on registers (PER or CCx) is effective after the synchronization delay.

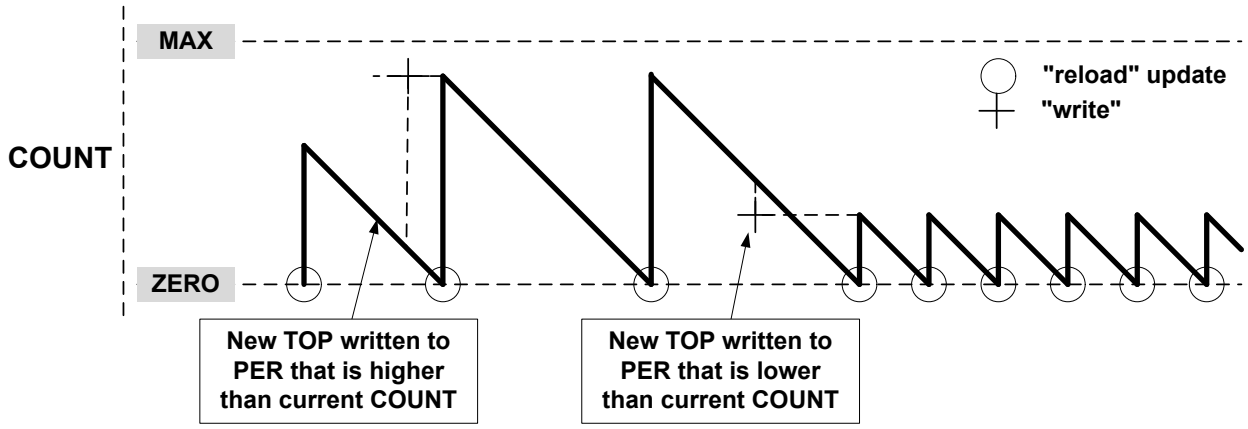
A counter wraparound can occur in any operation mode when up-counting without buffering (see the following figure).

Figure 40-8. Unbuffered Single-Slope Up-Counting Operation



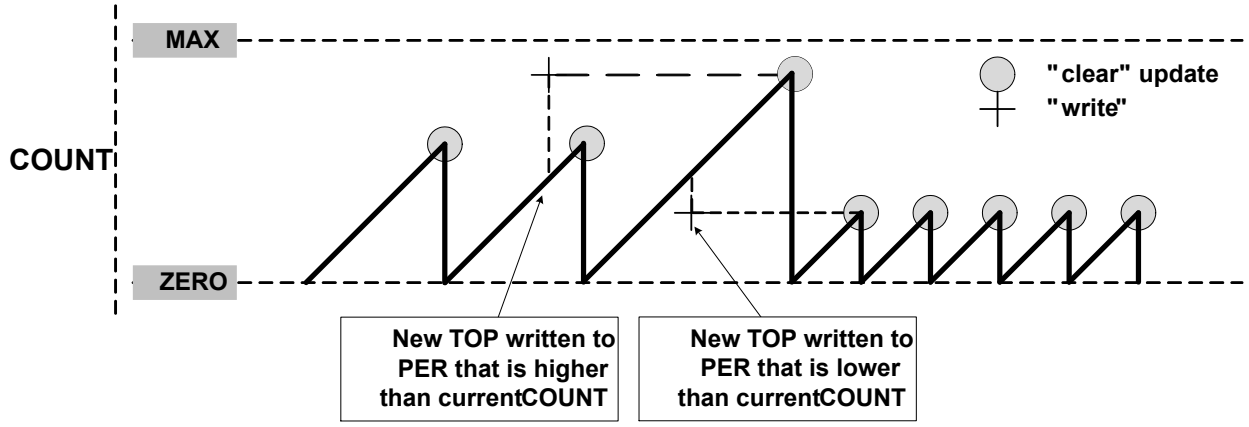
COUNT and TOP are continuously compared, so when a new TOP value that is lower than current COUNT is written to TOP, COUNT will wrap before a compare match.

Figure 40-9. Unbuffered Single-Slope Down-Counting Operation



When double buffering is used, the buffer can be written at any time and the counter will still maintain correct operation. The period register is always updated on the update condition, as shown in the following figure. This prevents wraparound and the generation of odd waveforms.

Figure 40-10. Changing the Period Using Buffering



40.6.2.8 Capture Operations

To enable and use capture operations, the corresponding Capture Channel x Enable bit in the Control A register (CTRLA.CAPTENx) must be written to '1'.

A capture trigger can be provided by input event line TC_EV or by asynchronous I/O pin WO[x] for each capture channel or by a TC event. To enable the capture from input event line, Event Input Enable bit in the Event Control register (EVCTRL.TCEI) must be written to '1'. To enable the capture from the I/O pin, the Capture On Pin x Enable bit in the CTRLA register (CTRLA.COPENx) must be written to '1'.

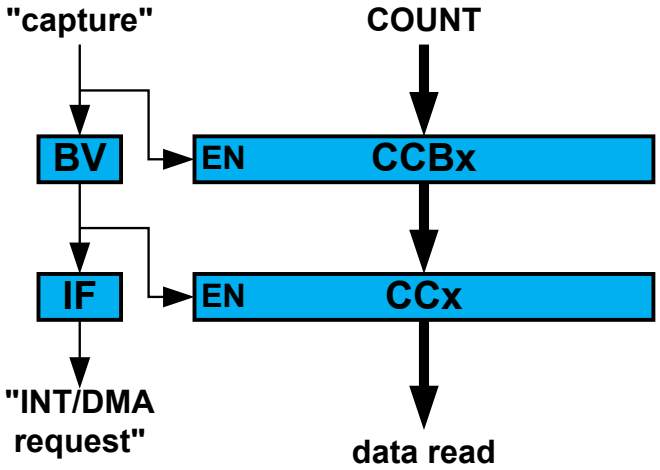
Notes:

1. Capture on I/Os is only possible in 'Event' and 'Time-Stamp' capture action modes. Other modes can only use internal events. (If I/Os toggling is needed in other modes, then the I/Os edge must be configured for generating internal events).
2. Capture on an event from the Event System is possible in 'Event', 'PPW/PWP/PW', and 'Time-Stamp' capture action modes. In this case, the event system channels must be configured to operate in asynchronous mode of operation.
3. Depending on CTRLA.COPENx, channel x can be configured for I/Os or internal event capture (both are mutually exclusive). One channel can be configured for I/Os capture while the other uses internal event capture.

By default, a capture operation is done when a rising edge is detected on the input signal. Capture on falling edge is available, its activation is depending on the input source:

- When the channel is used with a I/O pin, write a '1' to the corresponding Invert Enable bit in the Drive Control register (DRVCTRL.INVENx).
- When the channel is counting events from the Event System, write a '1' to the TC Event Input Invert Enable bit in Event Control register (EVCTRL.TCINV).

Figure 40-11. Capture Double Buffering

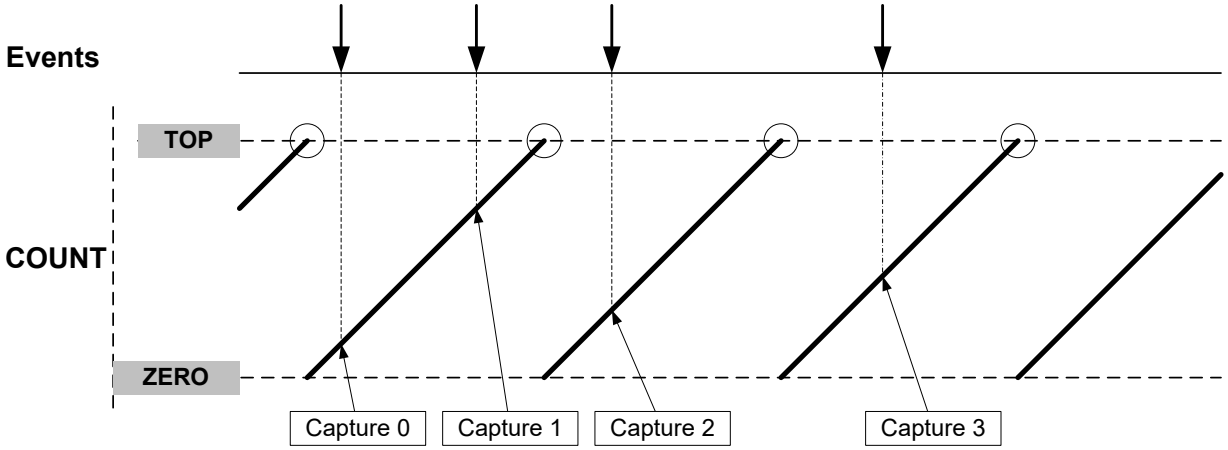


For input capture, the buffer register and the corresponding CCx act like a FIFO. When CCx is empty or read, any content in CCBUFx is transferred to CCx. The buffer valid flag is passed to set the CCx interrupt flag (IF) and generate the optional interrupt, event or DMA request. The CCBUFx register value can't be read, all captured data must be read from CCx register.

40.6.2.8.1 Event Capture Action on Events or I/Os

The compare and capture channels can be used as input capture channels to capture events from the Event System or I/O pins and give them a timestamp. This mode is selected when EVTCTRL.EVACT is configured either as OFF, RETRIGGER, COUNT or START. The following figure shows four capture events for one capture channel.

Figure 40-12. Input Capture Timing



The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

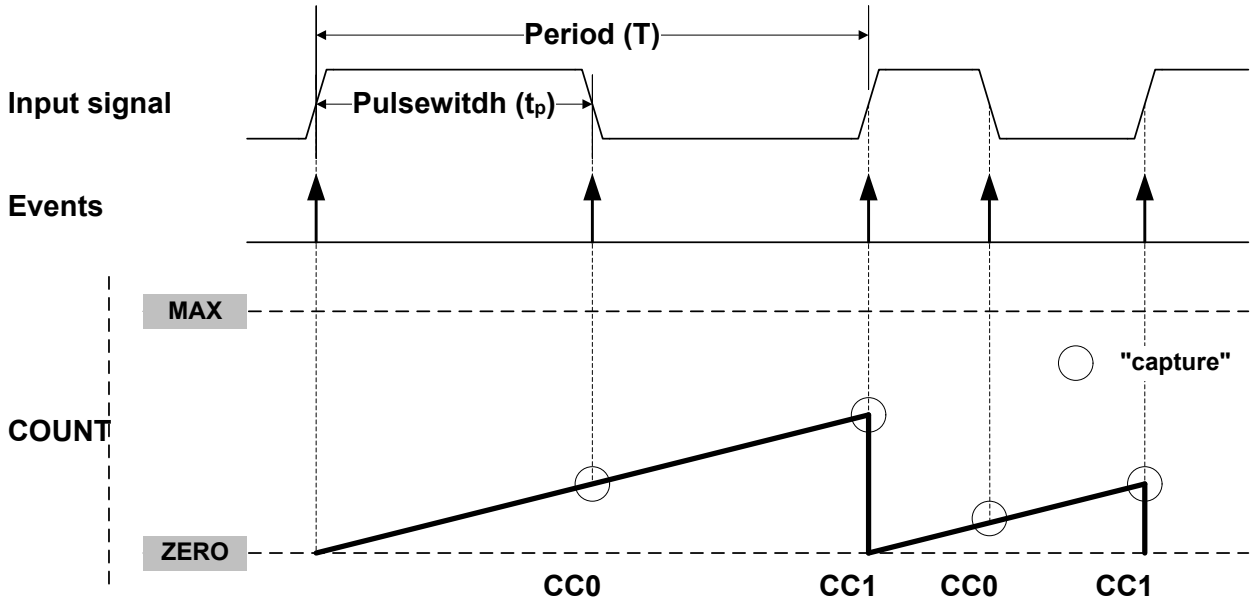
40.6.2.8.2 Period and Pulse-Width (PPW) Capture Action

The TC can perform two input captures and restart the counter on one of the edges. This enables the TC to measure the pulse width and period and to characterize the frequency *f* and duty cycle of an input signal:

$$f = \frac{1}{T}$$

$$\text{dutyCycle} = \frac{t_p}{T}$$

Figure 40-13. PWP Capture



Selecting PWP in the Event Action bit group in the Event Control register (EVCTRL.EVACT) enables the TC to perform one capture action on the rising edge and the other one on the falling edge. The period *T* will be captured into CC1 and the pulse width t_p in CC0. EVCTRL.EVACT=PPW (period and pulse-width) offers identical functionality, but will capture *T* into CC0 and t_p into CC1.

The TC Event Input Invert Enable bit in the Event Control register (EVCTRL.TCINV) is used to select whether the wraparound must occur on the rising edge or the falling edge. If EVCTRL.TCINV=1, the wraparound will happen on the falling edge. In case pin capture is enabled, this can also be achieved by modifying the value of the DRVCTRL.INVENx bit.

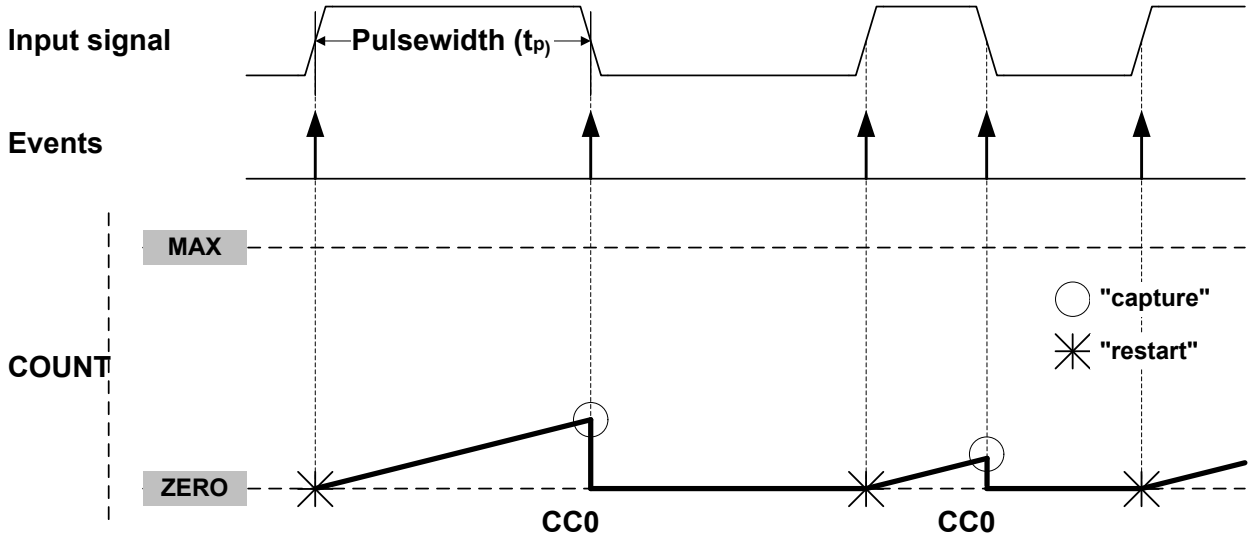
The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

Note: The corresponding capture is working only if the channel is enabled in capture mode (CTRLA.CAPTENx=1). If not, the capture action is ignored and the channel is enabled in compare mode of operation. Consequently, both channels must be enabled in order to fully characterize the input.

40.6.2.8.3 Pulse-Width (PW) Capture Action on Events

The TC performs the input capture on the falling edge of the input signal. When the edge is detected, the counter value is cleared and the TC stops counting. When a rising edge is detected on the input signal, the counter restarts the counting operation. To enable the operation on opposite edges, the input signal to capture must be inverted (refer to EVCTRL.TCEINV).

Figure 40-14. Pulse-Width Capture on Channel 0



The TC can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Interrupt flag (INTFLAG.MCx) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

40.6.3 Additional Features

40.6.3.1 One-Shot Operation

When one-shot is enabled, the counter automatically stops on the next Counter Overflow or Underflow condition. When the counter is stopped, the Stop bit in the Status register (STATUS.STOP) is automatically set and the waveform outputs are set to zero.

One-shot operation is enabled by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT), and disabled by writing a '1' to CTRLBCLR.ONESHOT. When enabled, the TC will count until an overflow or underflow occurs and stops counting operation. The one-shot operation can be restarted by a re-trigger software command, a re-trigger event, or a start event. When the counter restarts its operation, STATUS.STOP is automatically cleared.

40.6.3.2 Time-Stamp Capture on Events or I/Os

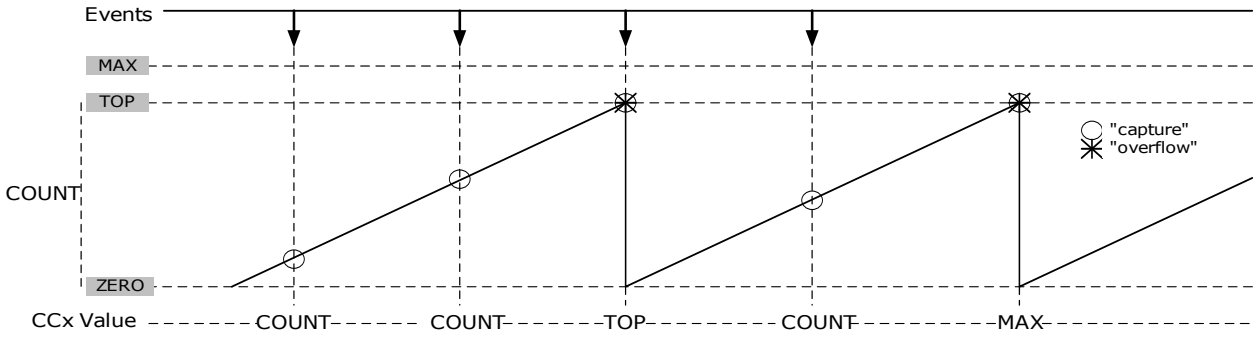
This feature is enabled when the Capture Time Stamp (STAMP) Event Action in Event Control register (EVCTRL.EVACT) is selected. The counter TOP value must be smaller than MAX.

When a capture event from the Event System or the I/O pin is detected, the COUNT value is copied into the corresponding Channel x Compare/Capture Value (CCx) register. In case of an overflow, the MAX value is copied into the corresponding CCx register.

When a valid captured value is present in the capture channel register, the corresponding Capture Channel x Interrupt Flag (INTFLAG.MCx) is set.

The timer/counter can detect capture overflow of the input capture channels: When a new capture event is detected while the Capture Channel interrupt flag (INTFLAG.MCx) is still set, the new time-stamp will not be stored and INTFLAG.ERR will be set.

Figure 40-15. Time Stamp



40.6.3.3 Minimum Capture

The minimum capture is enabled by writing the CAPTMIN mode in the Channel n Capture Mode bits in the Control A register (CTRLA.CAPTMODEn = CAPTMIN).

CCx Content:

In CAPTMIN operations, CCx keeps the Minimum captured values. Before enabling this mode of capture, the user must initialize the corresponding CCx register value to a value different from zero. If the CCx register initial value is zero, no captures will be performed using the corresponding channel.

MCx Behaviour:

In CAPTMIN operation, capture is performed only when on capture event time. The counter value is lower than the last captured value. The MCx interrupt flag is set only when on capture event time. The counter value is higher or equal to the value captured on the previous event. Therefore, the interrupt flag is set when a new absolute local Minimum value is detected.

40.6.3.4 Maximum Capture

The maximum capture is enabled by writing the CAPTMAX mode in the Channel n Capture Mode bits in the Control A register (CTRLA.CAPTMODEn = CAPTMAX).

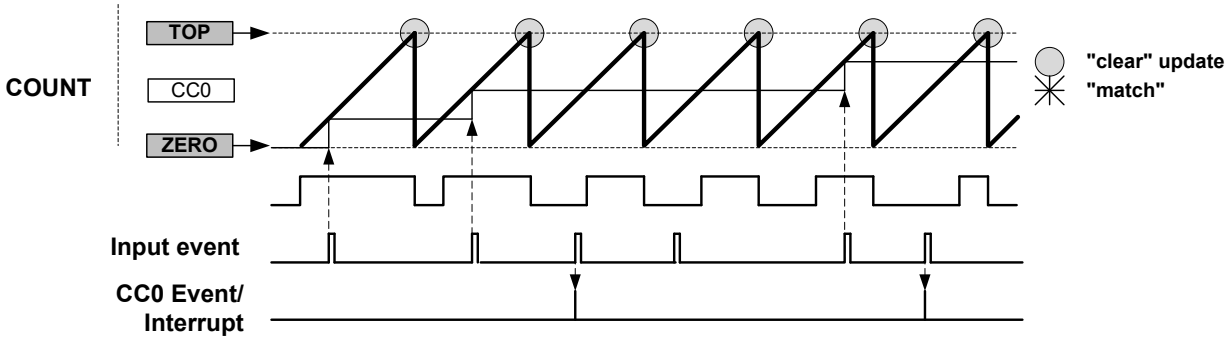
CCx Content:

In CAPTMAX operations, CCx keeps the Maximum captured values. Before enabling this mode of capture, the user must initialize the corresponding CCx register value to a value different from TOP. If the CCx register initial value is TOP, no captures will be performed using the corresponding channel.

MCx Behaviour:

In CAPTMAX operation, capture is performed only when on capture event time. The counter value is higher than the last captured value. The MCx interrupt flag is set only when on capture event time. The counter value is lower or equal to the value captured on the previous event. Therefore, the interrupt flag is set when a new absolute local Maximum value is detected.

Figure 40-16. Maximum Capture Operation with CC0 Initialized with ZERO Value



40.6.4 DMA Operation

The TC can generate the following DMA requests:

- Overflow (OVF): the request is set when an update condition (overflow, underflow or re-trigger) is detected, the request is cleared by hardware on DMA acknowledge.
- Match or Capture Channel x (MCx): for a compare channel, the request is set on each compare match detection, the request is cleared by hardware on DMA acknowledge. For a capture channel, the request is set when valid data is present in the CCx register, and cleared when CCx register is read.

40.6.5 Interrupts

The TC has the following interrupt sources:

- Overflow/Underflow (OVF)
- Match or Capture Channel x (MCx)
- Capture Overflow Error (ERR)

Each interrupt source has an interrupt flag associated with it. The interrupt flag in the Interrupt Flag Status and Clear register (INTFLAG) is set when the interrupt condition occurs.

Each interrupt can be individually enabled by writing a '1' to the corresponding bit in the Interrupt Enable Set register (INTENSET), and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear register (INTENCLR). The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until either the interrupt flag is cleared, the interrupt is disabled, or the TC is reset. See *INTFLAG* from Related Links for more details on how to clear the interrupt flags.

The TC has one common interrupt request line for all the interrupt sources. The user must read the INTFLAG register to determine which interrupt condition is present.

Note that interrupts must be globally enabled for interrupt requests to be generated. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)
40.7.4.7. INTFLAG

40.6.6 Events

The TC can generate the following output events:

- Overflow/Underflow (OVF)
- Match or Capture Channel x (MCX0-1)

Writing a '1' to an Event Output bit in the Event Control register (EVCTRL.MCEOx) enables the corresponding output event. The output event is disabled by writing EVCTRL.MCEOx='0'.

One of the following event actions can be selected by the Event Action bit group in the Event Control register (EVCTRL.EVACT):

- Disable event action (OFF)
- Start TC (START)
- Re-trigger TC (RETRIGGER)
- Count on event (COUNT)
- Capture time stamp (STAMP)
- Capture Period (PPW and PWP)
- Capture Pulse Width (PW)

Writing a '1' to the TC Event Input bit in the Event Control register (EVCTRL.TCEI) enables input events (EVU0-2) to the TC. Writing a '0' to this bit disables input events to the TC. The TC requires only asynchronous event inputs. See *Event System (EVSYS)* from Related Links for additional information on configuring the asynchronous events.

Related Links

28. Event System (EVSYS)

40.6.7 Sleep Mode Operation

The TC can be configured to operate in any sleep mode. To be able to run in standby, the RUNSTDBY bit in the Control A register (CTRLA.RUNSTDBY) must be '1'. This peripheral can wake up the device from any sleep mode using interrupts or perform actions through the Event System.

If the On Demand bit in the Control A register (CTRLA.ONDEMAND) is written to '1', the module stops requesting its peripheral clock when the STOP bit in STATUS register (STATUS.STOP) is set to '1'. When a re-trigger or start condition is detected, the TC requests the clock before the operation starts.

40.6.8 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset and Enable bits in Control A register (CTRLA.SWRST and CTRLA.ENABLE)
- Capture Channel Buffer Valid bit in STATUS register (STATUS.CCBUFVx)

The following registers are synchronized when written:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Count Value register (COUNT)
- Period Value and Period Buffer Value registers (PER and PERBUF)
- Channel x Compare/Capture Value and Channel x Compare/Capture Buffer Value registers (CCx and CCBUFx)

The following registers are synchronized when read:

- Count Value register (COUNT): synchronization is done on demand through READSYNC command (CTRLBSET.CMD)
- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Channel x Compare/Capture Value (CCx)

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

Required read synchronization is denoted by the "Read-Synchronized" property in the register description.

40.7 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers are optionally write-protected by the Peripheral Access Controller (PAC). Optional PAC write protection is denoted by the “PAC Write-Protection” property in each individual register description.

Some registers are synchronized when read and/or written. Synchronization is denoted by the “Write-Synchronized” or the “Read-Synchronized” property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the peripheral is disabled. Enable protection is denoted by the “Enable-Protected” property in each individual register description.

Note: All registers in this table have corresponding CLR, SET and INV registers at its virtual address, plus an offset of 0x4, 0x8 and 0xC, respectively. See *CLR, SET, and INV Registers* from Related Links.

Related Links

6.1.9. CLR, SET and INV Registers

40.7.1 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
		15:8					ALOCK	PRESCALER[2:0]		
		23:16				COPENx				CAPTENx
		31:24								
0x04	CTRLBCLR	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x05	CTRLBSET	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x06	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
		15:8			MCEO1	MCEO0				OVFEO
0x08	INTENCLR	7:0			MC1	MC0			ERR	OVF
0x09	INTENSET	7:0			MC1	MC0			ERR	OVF
0x0A	INTFLAG	7:0			MC1	MC0			ERR	OVF
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
0x0C	WAVE	7:0							WAVEGEN[1:0]	
0x0D	DRVCTRL	7:0								INVENx
0x0E	Reserved									
0x0F	DBGCTRL	7:0								DBGRUN
0x10	SYNCBUSY	7:0		CCx	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11 ... 0x13	Reserved									
0x14	COUNT	7:0	COUNT[7:0]							
0x15 ... 0x1A	Reserved									
0x1B	PER	7:0	PER[7:0]							
0x1C	CCx	7:0	CC[7:0]							
0x1D ... 0x2E	Reserved									
0x2F	PERBUF	7:0	PERBUF[7:0]							
0x30	CCBUFx	7:0	CCBUF[7:0]							

40.7.2 Register Description - 8-bit Mode

40.7.2.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				COPENx				CAPTENx
Access				R/W				R/W
Reset				0				0

Bit	15	14	13	12	11	10	9	8
					ALOCK	PRESCALER[2:0]		
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	W
Reset	0	0	0	0	0	0	0	0

Bit 20 – COPENx Capture On Pin x Enable [x=1..0]
Bit x of COPEN[:0] selects the trigger source for capture operation, either events or I/O pin input.
This bit is not synchronized.

Value	Description
0	Event from Event System is selected as trigger source for capture operation on channel x.
1	I/O pin is selected as trigger source for capture operation on channel x.

Bit 16 – CAPTENx Capture Channel x Enable [x=1..0]
Bit x of CAPTEN[:0] selects whether channel x is a capture or a compare channel.
These bits are not synchronized.

Value	Description
0	CAPTEN disables capture on channel x.
1	CAPTEN enables capture on channel x.

Bit 11 – ALOCK Auto Lock
When this bit is set, Lock bit update (LUPD) is set to '1' on each overflow/underflow or re-trigger event.
This bit is not synchronized.

Value	Description
0	The LUPD bit is not affected on overflow/underflow, and re-trigger event.
1	The LUPD bit is set on each overflow/underflow or re-trigger event.

Bits 10:8 – PRESCALER[2:0] Prescaler
These bits select the counter prescaler factor.
These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8

Value	Name	Description
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

Bit 7 – ONDEMAND Clock On Demand

This bit selects the clock requirements when the TC is stopped.
In standby mode, if the Run in Standby bit (CTRLA.RUNSTDBY) is '0', ONDEMAND is forced to '0'.
This bit is not synchronized.

Value	Description
0	The On Demand is disabled. If On Demand is disabled, the TC will continue to request the clock when its operation is stopped (STATUS.STOP=1).
1	The On Demand is enabled. When On Demand is enabled, the stopped TC will not request the clock. The clock is requested when a software re-trigger command is applied or when an event with start/re-trigger action is detected.

Bit 6 – RUNSTDBY Run in Standby

This bit is used to keep the TC running in standby mode.
This bit is not synchronized.

Value	Description
0	The TC is halted in standby.
1	The TC continues to run in standby.

Bits 5:4 – PRESCSYNC[1:0] Prescaler and Counter Synchronization

These bits select whether the counter must wrap around on the next GCLK_TCx clock or the next prescaled GCLK_TCx clock. It also makes it possible to reset the prescaler.
These bits are not synchronized.

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

Bits 3:2 – MODE[1:0] Timer Counter Mode

These bits select the counter mode.
These bits are not synchronized.

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
0x3	-	Reserved

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately, and the ENABLE Synchronization Busy bit in the SYNCBUSY register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the TC, except DBGCTRL, to their initial state, and the TC will be disabled.

Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.
This bit is not enable-protected.

40.7.2.2 Control B Clear

Name: CTRLBCLR
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to clear bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set register (CTRLBSET).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bits 7:5 – CMD[2:0] Command
These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.
Writing 0x0 to these bits has no effect.
Writing a '1' to any of these bits will clear the pending command.

Bit 2 – ONESHOT One-Shot on Counter
This bit controls one-shot operation of the TC.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will disable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update
This bit controls the update operation of the TC buffered registers.
When CTRLB.LUPD is set, no update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.
This bit has no effect when input capture operation is enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the LUPD bit.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

Bit 0 – DIR Counter Direction
This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the bit and make the counter count down.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

40.7.2.3 Control B Set

Name: CTRLBSET
Offset: 0x05
Reset: 0x00
Property: PAC Write-Protection, Read-synchronized, Write-Synchronized

This register allows the user to set bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear register (CTRLBCLR).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bits 7:5 – CMD[2:0] Command
These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK_TC clock cycle. When a command has been executed, the CMD bit group will be read back as zero.
Writing 0x0 to these bits has no effect.
Writing a value different from 0x0 to these bits will issue a command for execution.

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

Bit 2 – ONESHOT One-Shot on Counter
This bit controls one-shot operation of the TC.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will enable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update
This bit controls the update operation of the TC buffered registers.
When CTRLB.LUPD is set, no update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the LUPD bit.
This bit has no effect when input capture operation is enabled.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

Bit 0 – DIR Counter Direction
This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will set the bit and make the counter count down.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

40.7.2.4 Event Control

Name: EVCTRL
Offset: 0x06
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				OVFEO
Access			R/W	R/W				R/W
Reset			0	0				0

Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV		EVACT[2:0]		
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bits 12, 13 – MCEOx Match or Capture Channel x Event Output Enable [x = 1..0]
These bits enable the generation of an event for every match or capture on channel x.

Value	Description
0	Match/Capture event on channel x is disabled and will not be generated.
1	Match/Capture event on channel x is enabled and will be generated for every compare/capture.

Bit 8 – OVFEO Overflow/Underflow Event Output Enable
This bit enables the Overflow/Underflow event. When enabled, an event will be generated when the counter overflows/underflows.

Value	Description
0	Overflow/Underflow event is disabled and will not be generated.
1	Overflow/Underflow event is enabled and will be generated for every counter overflow/underflow.

Bit 5 – TCEI TC Event Enable
This bit is used to enable asynchronous input events to the TC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

Bit 4 – TCINV TC Inverted Event Input Polarity
This bit inverts the asynchronous input event source.

Value	Description
0	Input event source is not inverted.
1	Input event source is inverted.

Bits 2:0 – EVACT[2:0] Event Action
These bits define the event action the TC will perform on an event.

Value	Name	Description
0x0	OFF	Event action disabled
0x1	RETRIGGER	Start, restart or retrigger TC on event
0x2	COUNT	Count on event
0x3	START	Start TC on event
0x4	STAMP	Time stamp capture
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	PW	Pulse width capture

40.7.2.5 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable [x = 1..0]
Writing a '0' to these bits has no effect.
Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

Bit 1 – ERR Error Interrupt Disable
Writing a '0' to these bits has no effect.
Writing a '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Disable
Writing a '0' to these bits has no effect.
Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

40.7.2.6 Interrupt Enable Set

Name: INTENSET
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable [x = 1..0]
Writing a '0' to these bits has no effect.
Writing a '1' to MCx will set the corresponding Match or Capture Channel x Interrupt Enable bit, which enables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

Bit 1 – ERR Error Interrupt Enable
Writing a '0' to these bits has no effect.
Writing a '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Enable
Writing a '0' to these bits has no effect.
Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

40.7.2.7 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x [x = 1..0]
This flag is set on a comparison match, or when the corresponding CCx register contains a valid capture value. This flag is set on the next CLK_TC_CNT cycle, and will generate an interrupt request if the corresponding Match or Capture Channel x Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.MCx) is '1'.
Writing a '0' to these bits has no effect.
Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag
In capture operation, this flag is automatically cleared when CCx register is read.

Bit 1 – ERR Error Interrupt Flag
This flag is set when a new capture occurs on a channel while the corresponding Match or Capture Channel x interrupt flag is set, in which case there is no place to store the new capture.
Writing a '0' to these bits has no effect.
Writing a '1' to this bit clears the Error interrupt flag.

Bit 0 – OVF Overflow Interrupt Flag
This flag is set on the next CLK_TC_CNT cycle after an overflow condition occurs, and will generate an interrupt request if INTENCLR.OVF or INTENSET.OVF is '1'.
Writing a '0' to these bits has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

40.7.2.8 Status

Name: STATUS
Offset: 0x0B
Reset: 0x01
Property: Read-Synchronized

Bit	7	6	5	4	3	2	1	0
			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
Access			R/W	R/W	R/W		R	R
Reset			0	0	0		0	1

Bits 4, 5 – CCBUFVx Channel x Compare or Capture Buffer Valid [x = 1..0]
For a compare channel x, the bit x is set when a new value is written to the corresponding CCBUFx register. The bit x is cleared by writing a '1' to it when CTRLB.LUPD is set, or it is cleared automatically by hardware on UPDATE condition.
For a capture channel x, the bit x is set when a valid capture value is stored in the CCBUFx register. The bit x is cleared automatically when the CCx register is read.

Bit 3 – PERBUFV Period Buffer Valid
This bit is set when a new value is written to the PERBUF register. The bit is cleared by writing '1' to the corresponding location when CTRLB.LUPD is set, or automatically cleared by hardware on UPDATE condition. This bit is available only in 8-bit mode and will always read zero in 16- and 32-bit modes.

Bit 1 – SLAVE Client Status Flag
This bit is only available in 32-bit mode on the Client TC (i.e., TC1, TC3, TC5 and/or TC7). The bit is set when the associated Host TC (TC0, TC2, TC4 and/or TC6, respectively) is set to run in 32-bit mode.

Bit 0 – STOP Stop Status Flag
This bit is set when the TC is disabled, on a Stop command, or on an overflow/underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is '1'.

Value	Description
0	Counter is running.
1	Counter is stopped.

40.7.2.9 Waveform Generation Control

Name: WAVE
Offset: 0x0C
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
							WAVEGEN[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 1:0 – WAVEGEN[1:0] Waveform Generation Mode
These bits select the waveform generation operation. They affect the top value, as shown in Waveform Output Operations. They also control whether frequency or PWM waveform generation must be used. The waveform generation operations are explained in Waveform Output Operations. See *Waveform Output Operations* from Related Links.
These bits are not synchronized.

Value	Name	Operation	Top Value	Output Waveform on Match	Output Waveform on Wraparound
0x0	NFRQ	Normal frequency	PER[1] / Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER[1] / Max	Set	Clear
0x3	MPWM	Match PWM	CC0	Set	Clear

1. This depends on the TC mode: In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode, it is the respective MAX value.

Related Links
40.6.2.6.1. Waveform Output Operations

40.7.2.10 Driver Control

Name: DRVCTRL
Offset: 0x0D
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
								INVENx
Access								R/W
Reset								0

Bit 0 – INVENx Output Waveform x Invert Enable [x=1..0]
Bit x of INVEN[:0] selects inversion of the output or capture trigger input of channel x.

Value	Description
0	Disable inversion of the WO[x] output and IO input pin.
1	Enable inversion of the WO[x] output and IO input pin.

40.7.2.11 Debug Control

Name: DBGCTRL
Offset: 0x0F
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Run in Debug Mode

This bit is not affected by a software Reset, and must not be changed by software while the TC is enabled.

Value	Description
0	The TC is halted when the device is halted in debug mode.
1	The TC continues normal operation when the device is halted in debug mode.

40.7.2.12 Synchronization Busy

Name: SYNCBUSY
Offset: 0x10
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CCx	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit 6 – CCx Compare/Capture Channel x Synchronization Busy [x=0..1]
For details on CC channels number, refer to each TC feature list.
This bit is set when the synchronization of CCx between clock domains is started.
This bit is also set when the CCBUFx is written, and cleared on update condition. The bit is automatically cleared when the STATUS.CCBUFx bit is cleared.

Bit 5 – PER PER Synchronization Busy
This bit is cleared when the synchronization of PER between the clock domains is complete.
This bit is set when the synchronization of PER between clock domains is started.
This bit is also set when the PER is written, and cleared on update condition. The bit is automatically cleared when the STATUS.PERBUF bit is cleared.

Bit 4 – COUNT COUNT Synchronization Busy
This bit is cleared when the synchronization of COUNT between the clock domains is complete.
This bit is set when the synchronization of COUNT between clock domains is started.

Bit 3 – STATUS STATUS Synchronization Busy
This bit is cleared when the synchronization of STATUS between the clock domains is complete.
This bit is set when a '1' is written to the Capture Channel Buffer Valid status flags (STATUS.CCBUFVx) and the synchronization of STATUS between clock domains is started.

Bit 2 – CTRLB CTRLB Synchronization Busy
This bit is cleared when the synchronization of CTRLB between the clock domains is complete.
This bit is set when the synchronization of CTRLB between clock domains is started.

Bit 1 – ENABLE ENABLE Synchronization Busy
This bit is cleared when the synchronization of ENABLE bit between the clock domains is complete.
This bit is set when the synchronization of ENABLE bit between clock domains is started.

Bit 0 – SWRST SWRST Synchronization Busy
This bit is cleared when the synchronization of SWRST bit between the clock domains is complete.
This bit is set when the synchronization of SWRST bit between clock domains is started.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

40.7.2.13 Counter Value, 8-bit Mode

Name: COUNT
Offset: 0x14
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

Note: Prior to any read access, this register must be synchronized by the user by writing the according TC Command value to the Control B Set register (CTRLBSET.CMD = READSYNC).

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – COUNT[7:0] Counter Value
These bits contain the current counter value.

40.7.2.14 Period Value, 8-bit Mode

Name: PER
Offset: 0x1B
Reset: 0xFF
Property: Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	PER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 7:0 – PER[7:0] Period Value
These bits hold the value of the Period Buffer register PERBUF. The value is copied to PER register on UPDATE condition.

40.7.2.15 Channel x Compare/Capture Value, 8-bit Mode

Name: CCx
Offset: 0x1C
Reset: 0x00
Property: Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CC[7:0] Channel x Compare/Capture Value
These bits contain the compare/capture value in 8-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (WAVE.WAVEGEN), the CC0 register is used as a period register.

40.7.2.16 Period Buffer Value, 8-bit Mode

Name: PERBUF
Offset: 0x2F
Reset: 0xFF
Property: Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	PERBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 7:0 – PERBUF[7:0] Period Buffer Value
These bits hold the value of the period buffer register. The value is copied to PER register on UPDATE condition.

40.7.2.17 Channel x Compare Buffer Value, 8-bit Mode

Name: CCBUFx
Offset: 0x30
Reset: 0x00
Property: Write-Synchronized

Bit	7	6	5	4	3	2	1	0
	CCBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CCBUF[7:0] Channel x Compare Buffer Value
These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

40.7.3 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
		15:8					ALOCK	PRESCALER[2:0]		
		23:16				COPENx				CAPTENx
		31:24								
0x04	CTRLBCLR	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x05	CTRLBSET	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x06	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
		15:8			MCEO1	MCEO0				OVFEO
0x08	INTENCLR	7:0			MC1	MC0			ERR	OVF
0x09	INTENSET	7:0			MC1	MC0			ERR	OVF
0x0A	INTFLAG	7:0			MC1	MC0			ERR	OVF
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
0x0C	WAVE	7:0							WAVEGEN[1:0]	
0x0D	DRVCTRL	7:0								INVENx
0x0E	Reserved									
0x0F	DBGCTRL	7:0								DBGRUN
0x10	SYNCBUSY	7:0		CCx	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11 ... 0x13	Reserved									
0x14	COUNT	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
0x16 ... 0x1B	Reserved									
0x1C	CC0	7:0	CC[7:0]							
		15:8	CC[15:8]							
0x1E	CC1	7:0	CC[7:0]							
		15:8	CC[15:8]							
0x20 ... 0x2F	Reserved									
0x30	CCBUF0	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							
0x32	CCBUF1	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							

40.7.4 Register Description - 16-bit Mode

40.7.4.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				COPENx				CAPTENx
Access				R/W				R/W
Reset				0				0

Bit	15	14	13	12	11	10	9	8
					ALOCK	PRESCALER[2:0]		
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	W
Reset	0	0	0	0	0	0	0	0

Bit 20 – COPENx Capture On Pin x Enable [x=1..0]
Bit x of COPEN[:0] selects the trigger source for capture operation, either events or I/O pin input.
This bit is not synchronized.

Value	Description
0	Event from Event System is selected as trigger source for capture operation on channel x.
1	I/O pin is selected as trigger source for capture operation on channel x.

Bit 16 – CAPTENx Capture Channel x Enable [x=1..0]
Bit x of CAPTEN[:0] selects whether channel x is a capture or a compare channel.
These bits are not synchronized.

Value	Description
0	CAPTEN disables capture on channel x.
1	CAPTEN enables capture on channel x.

Bit 11 – ALOCK Auto Lock
When this bit is set, Lock bit update (LUPD) is set to '1' on each overflow/underflow or re-trigger event.
This bit is not synchronized.

Value	Description
0	The LUPD bit is not affected on overflow/underflow, and re-trigger event.
1	The LUPD bit is set on each overflow/underflow or re-trigger event.

Bits 10:8 – PRESCALER[2:0] Prescaler
These bits select the counter prescaler factor.
These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8

Value	Name	Description
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

Bit 7 – ONDEMAND Clock On Demand

This bit selects the clock requirements when the TC is stopped.
In standby mode, if the Run in Standby bit (CTRLA.RUNSTDBY) is '0', ONDEMAND is forced to '0'.
This bit is not synchronized.

Value	Description
0	The On Demand is disabled. If On Demand is disabled, the TC will continue to request the clock when its operation is stopped (STATUS.STOP=1).
1	The On Demand is enabled. When On Demand is enabled, the stopped TC will not request the clock. The clock is requested when a software re-trigger command is applied or when an event with start/re-trigger action is detected.

Bit 6 – RUNSTDBY Run in Standby

This bit is used to keep the TC running in standby mode.
This bit is not synchronized.

Value	Description
0	The TC is halted in standby.
1	The TC continues to run in standby.

Bits 5:4 – PRESCSYNC[1:0] Prescaler and Counter Synchronization

These bits select whether the counter must wrap around on the next GCLK_TCx clock or the next prescaled GCLK_TCx clock. It also makes it possible to reset the prescaler.
These bits are not synchronized.

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

Bits 3:2 – MODE[1:0] Timer Counter Mode

These bits select the counter mode.
These bits are not synchronized.

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
0x3	-	Reserved

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately, and the ENABLE Synchronization Busy bit in the SYNCBUSY register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the TC, except DBGCTRL, to their initial state, and the TC will be disabled.

Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.
This bit is not enable-protected.

40.7.4.2 Control B Clear

Name: CTRLBCLR
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to clear bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set register (CTRLBSET).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bits 7:5 – CMD[2:0] Command
Writing a `0' to these bits has no effect.
Writing a '1' to any of these bits will clear the pending command.

Bit 2 – ONESHOT One-Shot on Counter
This bit controls one-shot operation of the TC.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will disable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update
This bit controls the update operation of the TC buffered registers.
When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.
This bit has no effect when input capture operation is enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the LUPD bit.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

Bit 0 – DIR Counter Direction
This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

40.7.4.3 Control B Set

Name: CTRLBSET
Offset: 0x05
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

This register allows the user to set bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear register (CTRLBCLR).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bits 7:5 – CMD[2:0] Command
These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK_TCx clock cycle. When a command has been executed, the CMD bit group will be read back as zero.
Writing 0x0 to these bits has no effect.
Writing a value different from 0x0 to this bit field will issue a command for execution.

Important: This command requires synchronization before being executed.
A valid sequence is:the following:

- Issue CMD command (CTRLBSET.CMD = command)
- Wait for CMD synchronization (SYNCBUSY.CTRLB = 0)
- Wait for CMD read back as zero (CTRLBSET.CMD = 0)

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

Bit 2 – ONESHOT One-Shot on Counter
This bit controls one-shot operation of the TC.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will enable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update
This bit controls the update operation of the TC buffered registers.
When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the LUPD bit.
This bit has no effect when input capture operation is enabled.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

40.7.4.4 Event Control

Name: EVCTRL
Offset: 0x06
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				OVFEO
Access			R/W	R/W				R/W
Reset			0	0				0

Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV		EVACT[2:0]		
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bits 12, 13 – MCEOx Match or Capture Channel x Event Output Enable [x = 1..0]
These bits enable the generation of an event for every match or capture on channel x.

Value	Description
0	Match/Capture event on channel x is disabled and will not be generated.
1	Match/Capture event on channel x is enabled and will be generated for every compare/capture.

Bit 8 – OVFEO Overflow/Underflow Event Output Enable
This bit enables the Overflow/Underflow event. When enabled, an event will be generated when the counter overflows/underflows.

Value	Description
0	Overflow/Underflow event is disabled and will not be generated.
1	Overflow/Underflow event is enabled and will be generated for every counter overflow/underflow.

Bit 5 – TCEI TC Event Enable
This bit is used to enable asynchronous input events to the TC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

Bit 4 – TCINV TC Inverted Event Input Polarity
This bit inverts the asynchronous input event source.

Value	Description
0	Input event source is not inverted.
1	Input event source is inverted.

Bits 2:0 – EVACT[2:0] Event Action
These bits define the event action the TC will perform on an event.

Value	Name	Description
0x0	OFF	Event action disabled
0x1	RETRIGGER	Start, restart or retrigger TC on event
0x2	COUNT	Count on event
0x3	START	Start TC on event
0x4	STAMP	Time stamp capture
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	PW	Pulse width capture

40.7.4.5 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable
Writing a '0' to these bits has no effect.
Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

Bit 1 – ERR Error Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

40.7.4.6 Interrupt Enable Set

Name: INTENSET
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable
Writing a '0' to these bits has no effect.
Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

Bit 1 – ERR Error Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

40.7.4.7 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x
This flag is set on a comparison match, or when the corresponding CCx register contains a valid capture value. This flag is set on the next CLK_TC_CNT cycle, and will generate an interrupt request if the corresponding Match or Capture Channel x Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.MCx) is '1'.
Writing a '0' to one of these bits has no effect.
Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag
In capture operation, this flag is automatically cleared when CCx register is read.

Bit 1 – ERR Error Interrupt Flag
This flag is set when a new capture occurs on a channel while the corresponding Match or Capture Channel x interrupt flag is set, in which case there is nowhere to store the new capture.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Error interrupt flag.

Bit 0 – OVF Overflow Interrupt Flag
This flag is set on the next CLK_TC_CNT cycle after an overflow condition occurs, and will generate an interrupt request if INTENCLR.OVF or INTENSET.OVF is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

40.7.4.8 Status

Name: STATUS
Offset: 0x0B
Reset: 0x01
Property: Read-Synchronized

Bit	7	6	5	4	3	2	1	0
			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
Access			R/W	R/W	R/W		R	R
Reset			0	0	0		0	1

Bits 4, 5 – CCBUFV Channel x Compare or Capture Buffer Valid
For a compare channel x, the bit x is set when a new value is written to the corresponding CCBUFx register. The bit x is cleared by writing a '1' to it when CTRLB.LUPD is set, or it is cleared automatically by hardware on UPDATE condition.
For a capture channel x, the bit x is set when a valid capture value is stored in the CCBUFx register. The bit x is cleared automatically when the CCx register is read.

Bit 3 – PERBUFV Period Buffer Valid
This bit is set when a new value is written to the PERBUF register. The bit is cleared by writing '1' to the corresponding location when CTRLB.LUPD is set, or automatically cleared by hardware on UPDATE condition. This bit is available only in 8-bit mode and will always read zero in 16- and 32-bit modes.

Bit 1 – SLAVE Client Status Flag
This bit is only available in 32-bit mode on the Client TC (i.e., TC1 and/or TC3). The bit is set when the associated Host TC (TC0 and TC2, respectively) is set to run in 32-bit mode.

Bit 0 – STOP Stop Status Flag
This bit is set when the TC is disabled, on a Stop command, or on an overflow/underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is '1'.

Value	Description
0	Counter is running.
1	Counter is stopped.

40.7.4.9 Waveform Generation Control

Name: WAVE
Offset: 0x0C
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
							WAVEGEN[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 1:0 – WAVEGEN[1:0] Waveform Generation Mode
These bits select the waveform generation operation. They affect the top value, as shown in Waveform Output Operations. They also control whether frequency or PWM waveform generation must be used. The waveform generation operations are explained in Waveform Output Operations. See *Waveform Output Operations* from Related Links.
These bits are not synchronized.

Value	Name	Operation	Top Value	Output Waveform on Match	Output Waveform on Wraparound
0x0	NFRQ	Normal frequency	PER[1] / Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER[1] / Max	Set	Clear
0x3	MPWM	Match PWM	CC0	Set	Clear

1. This depends on the TC mode: In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode, it is the respective MAX value.

Related Links
40.6.2.6.1. Waveform Output Operations

40.7.4.10 Driver Control

Name: DRVCTRL
Offset: 0x0D
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
								INVENx
Access								R/W
Reset								0

Bit 0 – INVENx Output Waveform x Invert Enable
INVENx bit selects inversion of the output or capture trigger input of channel x.

Value	Description
0	Disable inversion of the WO[x] output and IO input pin.
1	Enable inversion of the WO[x] output and IO input pin.

40.7.4.11 Debug Control

Name: DBGCTRL
Offset: 0x0F
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Run in Debug Mode

This bit is not affected by a software Reset, and must not be changed by software while the TC is enabled.

Value	Description
0	The TC is halted when the device is halted in debug mode.
1	The TC continues normal operation when the device is halted in debug mode.

40.7.4.12 Synchronization Busy

Name: SYNCBUSY
Offset: 0x10
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CCx	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit 6 – CCx Compare/Capture Channel x Synchronization Busy [x=0..1]
For details on CC channels number, refer to each TC feature list.
This bit is set when the synchronization of CCx between clock domains is started.
This bit is also set when the CCBUFx is written, and cleared on update condition. The bit is automatically cleared when the STATUS.CCBUFx bit is cleared.

Bit 5 – PER PER Synchronization Busy
This bit is cleared when the synchronization of PER between the clock domains is complete.
This bit is set when the synchronization of PER between clock domains is started.
This bit is also set when the PER is written, and cleared on update condition. The bit is automatically cleared when the STATUS.PERBUF bit is cleared.

Bit 4 – COUNT COUNT Synchronization Busy
This bit is cleared when the synchronization of COUNT between the clock domains is complete.
This bit is set when the synchronization of COUNT between clock domains is started.

Bit 3 – STATUS STATUS Synchronization Busy
This bit is cleared when the synchronization of STATUS between the clock domains is complete.
This bit is set when a '1' is written to the Capture Channel Buffer Valid status flags (STATUS.CCBUFVx) and the synchronization of STATUS between clock domains is started.

Bit 2 – CTRLB CTRLB Synchronization Busy
This bit is cleared when the synchronization of CTRLB between the clock domains is complete.
This bit is set when the synchronization of CTRLB between clock domains is started.

Bit 1 – ENABLE ENABLE Synchronization Busy
This bit is cleared when the synchronization of ENABLE bit between the clock domains is complete.
This bit is set when the synchronization of ENABLE bit between clock domains is started.

Bit 0 – SWRST SWRST Synchronization Busy
This bit is cleared when the synchronization of SWRST bit between the clock domains is complete.
This bit is set when the synchronization of SWRST bit between clock domains is started.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

40.7.4.13 Counter Value, 16-bit Mode

Name: COUNT
Offset: 0x14
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

Note: Prior to any read access, this register must be synchronized by the user by writing the according TC Command value to the Control B Set register (CTRLBSET.CMD = READSYNC).

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – COUNT[15:0] Counter Value
These bits contain the current counter value.

40.7.4.14 Channel x Compare/Capture Value, 16-bit Mode

Name: CCx
Offset: 0x1C + x*0x02 [x=0..1]
Reset: 0x0000
Property: Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	CC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CC[15:0] Channel x Compare/Capture Value
These bits contain the compare/capture value in 16-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (WAVE.WAVEGEN), the CC0 register is used as a period register.

40.7.4.15 Channel x Compare Buffer Value, 16-bit Mode

Name: CCBUFx
Offset: 0x30 + x*0x02 [x=0..1]
Reset: 0x0000
Property: Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	CCBUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CCBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CCBUF[15:0] Channel x Compare Buffer Value
These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

40.7.5 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
		15:8					ALOCK	PRESCALER[2:0]		
		23:16				COPENx				CAPTENx
		31:24								
0x04	CTRLBCLR	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x05	CTRLBSET	7:0	CMD[2:0]					ONESHOT	LUPD	DIR
0x06	EVCTRL	7:0			TCEI	TCINV		EVACT[2:0]		
		15:8			MCEO1	MCEO0				OVFEO
0x08	INTENCLR	7:0			MC1	MC0			ERR	OVF
0x09	INTENSET	7:0			MC1	MC0			ERR	OVF
0x0A	INTFLAG	7:0			MC1	MC0			ERR	OVF
0x0B	STATUS	7:0			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
0x0C	WAVE	7:0							WAVEGEN[1:0]	
0x0D	DRVCTRL	7:0								INVENx
0x0E	Reserved									
0x0F	DBGCTRL	7:0								DBGRUN
0x10	SYNCBUSY	7:0		CCx	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
0x11 ... 0x13	Reserved									
0x14	COUNT	7:0	COUNT[7:0]							
		15:8	COUNT[15:8]							
		23:16	COUNT[23:16]							
		31:24	COUNT[31:24]							
0x18 ... 0x1B	Reserved									
0x1C	CC0	7:0	CC[7:0]							
		15:8	CC[15:8]							
		23:16	CC[23:16]							
		31:24	CC[31:24]							
0x20	CC1	7:0	CC[7:0]							
		15:8	CC[15:8]							
		23:16	CC[23:16]							
		31:24	CC[31:24]							
0x24 ... 0x2F	Reserved									
0x30	CCBUF0	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							
		23:16	CCBUF[23:16]							
		31:24	CCBUF[31:24]							
0x34	CCBUF1	7:0	CCBUF[7:0]							
		15:8	CCBUF[15:8]							
		23:16	CCBUF[23:16]							
		31:24	CCBUF[31:24]							

40.7.6 Register Description - 32-bit Mode

40.7.6.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Write-Synchronized, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				COPENx				CAPTENx
Access				R/W				R/W
Reset				0				0

Bit	15	14	13	12	11	10	9	8
					ALOCK	PRESCALER[2:0]		
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ONDEMAND	RUNSTDBY	PRESCSYNC[1:0]		MODE[1:0]		ENABLE	SWRST
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	W
Reset	0	0	0	0	0	0	0	0

Bit 20 – COPENx Capture On Pin x Enable [x=1..0]
Bit x of COPEN[:0] selects the trigger source for capture operation, either events or I/O pin input.
This bit is not synchronized.

Value	Description
0	Event from Event System is selected as trigger source for capture operation on channel x.
1	I/O pin is selected as trigger source for capture operation on channel x.

Bit 16 – CAPTENx Capture Channel x Enable [x=1..0]
Bit x of CAPTEN[:0] selects whether channel x is a capture or a compare channel.
These bits are not synchronized.

Value	Description
0	CAPTEN disables capture on channel x.
1	CAPTEN enables capture on channel x.

Bit 11 – ALOCK Auto Lock
When this bit is set, Lock bit update (LUPD) is set to '1' on each overflow/underflow or re-trigger event.
This bit is not synchronized.

Value	Description
0	The LUPD bit is not affected on overflow/underflow, and re-trigger event.
1	The LUPD bit is set on each overflow/underflow or re-trigger event.

Bits 10:8 – PRESCALER[2:0] Prescaler
These bits select the counter prescaler factor.
These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TC
0x1	DIV2	Prescaler: GCLK_TC/2
0x2	DIV4	Prescaler: GCLK_TC/4
0x3	DIV8	Prescaler: GCLK_TC/8

Value	Name	Description
0x4	DIV16	Prescaler: GCLK_TC/16
0x5	DIV64	Prescaler: GCLK_TC/64
0x6	DIV256	Prescaler: GCLK_TC/256
0x7	DIV1024	Prescaler: GCLK_TC/1024

Bit 7 – ONDEMAND Clock On Demand

This bit selects the clock requirements when the TC is stopped.
In standby mode, if the Run in Standby bit (CTRLA.RUNSTDBY) is '0', ONDEMAND is forced to '0'.
This bit is not synchronized.

Value	Description
0	The On Demand is disabled. If On Demand is disabled, the TC will continue to request the clock when its operation is stopped (STATUS.STOP=1).
1	The On Demand is enabled. When On Demand is enabled, the stopped TC will not request the clock. The clock is requested when a software re-trigger command is applied or when an event with start/re-trigger action is detected.

Bit 6 – RUNSTDBY Run in Standby

This bit is used to keep the TC running in standby mode.
This bit is not synchronized.

Value	Description
0	The TC is halted in standby.
1	The TC continues to run in standby.

Bits 5:4 – PRESCSYNC[1:0] Prescaler and Counter Synchronization

These bits select whether the counter must wrap around on the next GCLK_TCx clock or the next prescaled GCLK_TCx clock. It also makes it possible to reset the prescaler.
These bits are not synchronized.

Value	Name	Description
0x0	GCLK	Reload or reset the counter on next generic clock
0x1	PRESC	Reload or reset the counter on next prescaler clock
0x2	RESYNC	Reload or reset the counter on next generic clock. Reset the prescaler counter
0x3	-	Reserved

Bits 3:2 – MODE[1:0] Timer Counter Mode

These bits select the counter mode.
These bits are not synchronized.

Value	Name	Description
0x0	COUNT16	Counter in 16-bit mode
0x1	COUNT8	Counter in 8-bit mode
0x2	COUNT32	Counter in 32-bit mode
0x3	-	Reserved

Bit 1 – ENABLE Enable

Due to synchronization, there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately, and the ENABLE Synchronization Busy bit in the SYNCBUSY register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.
This bit is not enable-protected.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the TC, except DBGCTRL, to their initial state, and the TC will be disabled.

Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.
This bit is not enable-protected.

40.7.6.2 Control B Clear

Name: CTRLBCLR
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection, Read-Synchronized, Write-Synchronized

This register allows the user to clear bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set register (CTRLBSET).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bits 7:5 – CMD[2:0] Command
Writing a `0' to these bits has no effect.
Writing a '1' to any of these bits will clear the pending command.

Bit 2 – ONESHOT One-Shot on Counter
This bit controls one-shot operation of the TC.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will disable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update
This bit controls the update operation of the TC buffered registers.
When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.
This bit has no effect when input capture operation is enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the LUPD bit.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

Bit 0 – DIR Counter Direction
This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

40.7.6.3 Control B Set

Name: CTRLBSET
Offset: 0x05
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

This register allows the user to set bits in the CTRLB register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Clear register (CTRLBCLR).

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]					ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bits 7:5 – CMD[2:0] Command
These bits are used for software control of the TC. The commands are executed on the next prescaled GCLK_TCx clock cycle. When a command has been executed, the CMD bit group will be read back as zero.
Writing 0x0 to these bits has no effect.
Writing a value different from 0x0 to this bit field will issue a command for execution.

Important: This command requires synchronization before being executed.
A valid sequence is:the following:

- Issue CMD command (CTRLBSET.CMD = command)
- Wait for CMD synchronization (SYNCBUSY.CTRLB = 0)
- Wait for CMD read back as zero (CTRLBSET.CMD = 0)

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force a start, restart or retrigger
0x2	STOP	Force a stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT

Bit 2 – ONESHOT One-Shot on Counter
This bit controls one-shot operation of the TC.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will enable one-shot operation.

Value	Description
0	The TC will wrap around and continue counting on an overflow/underflow condition.
1	The TC will wrap around and stop on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update
This bit controls the update operation of the TC buffered registers.
When CTRLB.LUPD is set, no any update of the registers with value of its buffered register is performed on hardware UPDATE condition. Locking the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the LUPD bit.
This bit has no effect when input capture operation is enabled.

Value	Description
0	The CCBUFx and PERBUF buffer registers value are copied into CCx and PER registers on hardware update condition.
1	The CCBUFx and PERBUF buffer registers value are not copied into CCx and PER registers on hardware update condition.

Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

40.7.6.4 Event Control

Name: EVCTRL
Offset: 0x06
Reset: 0x0000
Property: PAC Write-Protection, Enable-Protected

Bit	15	14	13	12	11	10	9	8
			MCEO1	MCEO0				OVFEO
Access			R/W	R/W				R/W
Reset			0	0				0

Bit	7	6	5	4	3	2	1	0
			TCEI	TCINV		EVACT[2:0]		
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bits 12, 13 – MCEOx Match or Capture Channel x Event Output Enable [x = 1..0]
These bits enable the generation of an event for every match or capture on channel x.

Value	Description
0	Match/Capture event on channel x is disabled and will not be generated.
1	Match/Capture event on channel x is enabled and will be generated for every compare/capture.

Bit 8 – OVFEO Overflow/Underflow Event Output Enable
This bit enables the Overflow/Underflow event. When enabled, an event will be generated when the counter overflows/underflows.

Value	Description
0	Overflow/Underflow event is disabled and will not be generated.
1	Overflow/Underflow event is enabled and will be generated for every counter overflow/underflow.

Bit 5 – TCEI TC Event Enable
This bit is used to enable asynchronous input events to the TC.

Value	Description
0	Incoming events are disabled.
1	Incoming events are enabled.

Bit 4 – TCINV TC Inverted Event Input Polarity
This bit inverts the asynchronous input event source.

Value	Description
0	Input event source is not inverted.
1	Input event source is inverted.

Bits 2:0 – EVACT[2:0] Event Action
These bits define the event action the TC will perform on an event.

Value	Name	Description
0x0	OFF	Event action disabled
0x1	RETRIGGER	Start, restart or retrigger TC on event
0x2	COUNT	Count on event
0x3	START	Start TC on event
0x4	STAMP	Time stamp capture
0x5	PPW	Period captured in CC0, pulse width in CC1
0x6	PWP	Period captured in CC1, pulse width in CC0
0x7	PW	Pulse width capture

40.7.6.5 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x08
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to disable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set register (INTENSET).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable
Writing a '0' to these bits has no effect.
Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

Bit 1 – ERR Error Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Error Interrupt Enable bit, which disables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Disable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Overflow Interrupt Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

40.7.6.6 Interrupt Enable Set

Name: INTENSET
Offset: 0x09
Reset: 0x00
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear register (INTENCLR).

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x Interrupt Enable
Writing a '0' to these bits has no effect.
Writing a '1' to MCx will clear the corresponding Match or Capture Channel x Interrupt Enable bit, which disables the Match or Capture Channel x interrupt.

Value	Description
0	The Match or Capture Channel x interrupt is disabled.
1	The Match or Capture Channel x interrupt is enabled.

Bit 1 – ERR Error Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Error Interrupt Enable bit, which enables the Error interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Overflow Interrupt Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

40.7.6.7 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			MC1	MC0			ERR	OVF
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 4, 5 – MCx Match or Capture Channel x

This flag is set on a comparison match, or when the corresponding CCx register contains a valid capture value. This flag is set on the next CLK_TC_CNT cycle, and will generate an interrupt request if the corresponding Match or Capture Channel x Interrupt Enable bit in the Interrupt Enable Set register (INTENSET.MCx) is '1'.
Writing a '0' to one of these bits has no effect.
Writing a '1' to one of these bits will clear the corresponding Match or Capture Channel x interrupt flag
In capture operation, this flag is automatically cleared when CCx register is read.

Bit 1 – ERR Error Interrupt Flag

This flag is set when a new capture occurs on a channel while the corresponding Match or Capture Channel x interrupt flag is set, in which case there is nowhere to store the new capture.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Error interrupt flag.

Bit 0 – OVF Overflow Interrupt Flag

This flag is set on the next CLK_TC_CNT cycle after an overflow condition occurs, and will generate an interrupt request if INTENCLR.OVF or INTENSET.OVF is '1'.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

40.7.6.8 Status

Name: STATUS
Offset: 0x0B
Reset: 0x01
Property: Read-Synchronized

Bit	7	6	5	4	3	2	1	0
			CCBUFV1	CCBUFV0	PERBUFV		SLAVE	STOP
Access			R/W	R/W	R/W		R	R
Reset			0	0	0		0	1

Bits 4, 5 – CCBUFV Channel x Compare or Capture Buffer Valid
For a compare channel x, the bit x is set when a new value is written to the corresponding CCBUFx register. The bit x is cleared by writing a '1' to it when CTRLB.LUPD is set, or it is cleared automatically by hardware on UPDATE condition.
For a capture channel x, the bit x is set when a valid capture value is stored in the CCBUFx register. The bit x is cleared automatically when the CCx register is read.

Bit 3 – PERBUFV Period Buffer Valid
This bit is set when a new value is written to the PERBUF register. The bit is cleared by writing '1' to the corresponding location when CTRLB.LUPD is set, or automatically cleared by hardware on UPDATE condition. This bit is available only in 8-bit mode and will always read zero in 16- and 32-bit modes.

Bit 1 – SLAVE Client Status Flag
This bit is only available in 32-bit mode on the Client TC (i.e., TC1 and/or TC3). The bit is set when the associated Host TC (TC0 and TC2, respectively) is set to run in 32-bit mode.

Bit 0 – STOP Stop Status Flag
This bit is set when the TC is disabled, on a Stop command, or on an overflow/underflow condition when the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) is '1'.

Value	Description
0	Counter is running.
1	Counter is stopped.

40.7.6.9 Waveform Generation Control

Name: WAVE
Offset: 0x0C
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
							WAVEGEN[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 1:0 – WAVEGEN[1:0] Waveform Generation Mode
These bits select the waveform generation operation. They affect the top value, as shown in Waveform Output Operations. They also control whether frequency or PWM waveform generation must be used. The waveform generation operations are explained in Waveform Output Operations. See *Waveform Output Operations* from Related Links.
These bits are not synchronized.

Value	Name	Operation	Top Value	Output Waveform on Match	Output Waveform on Wraparound
0x0	NFRQ	Normal frequency	PER[1] / Max	Toggle	No action
0x1	MFRQ	Match frequency	CC0	Toggle	No action
0x2	NPWM	Normal PWM	PER[1] / Max	Set	Clear
0x3	MPWM	Match PWM	CC0	Set	Clear

1. This depends on the TC mode: In 8-bit mode, the top value is the Period Value register (PER). In 16- and 32-bit mode, it is the respective MAX value.

Related Links
40.6.2.6.1. Waveform Output Operations

40.7.6.10 Driver Control

Name: DRVCTRL
Offset: 0x0D
Reset: 0x00
Property: PAC Write-Protection, Enable-Protected

Bit	7	6	5	4	3	2	1	0
								INVENx
Access								R/W
Reset								0

Bit 0 – INVENx Output Waveform x Invert Enable
INVENx bit selects inversion of the output or capture trigger input of channel x.

Value	Description
0	Disable inversion of the WO[x] output and IO input pin.
1	Enable inversion of the WO[x] output and IO input pin.

40.7.6.11 Debug Control

Name: DBGCTRL
Offset: 0x0F
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN Run in Debug Mode

This bit is not affected by a software Reset, and must not be changed by software while the TC is enabled.

Value	Description
0	The TC is halted when the device is halted in debug mode.
1	The TC continues normal operation when the device is halted in debug mode.

40.7.6.12 Synchronization Busy

Name: SYNCBUSY
Offset: 0x10
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CCx	PER	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit 6 – CCx Compare/Capture Channel x Synchronization Busy [x=0..1]
For details on CC channels number, refer to each TC feature list.
This bit is set when the synchronization of CCx between clock domains is started.
This bit is also set when the CCBUFx is written, and cleared on update condition. The bit is automatically cleared when the STATUS.CCBUFx bit is cleared.

Bit 5 – PER PER Synchronization Busy
This bit is cleared when the synchronization of PER between the clock domains is complete.
This bit is set when the synchronization of PER between clock domains is started.
This bit is also set when the PER is written, and cleared on update condition. The bit is automatically cleared when the STATUS.PERBUF bit is cleared.

Bit 4 – COUNT COUNT Synchronization Busy
This bit is cleared when the synchronization of COUNT between the clock domains is complete.
This bit is set when the synchronization of COUNT between clock domains is started.

Bit 3 – STATUS STATUS Synchronization Busy
This bit is cleared when the synchronization of STATUS between the clock domains is complete.
This bit is set when a '1' is written to the Capture Channel Buffer Valid status flags (STATUS.CCBUFVx) and the synchronization of STATUS between clock domains is started.

Bit 2 – CTRLB CTRLB Synchronization Busy
This bit is cleared when the synchronization of CTRLB between the clock domains is complete.
This bit is set when the synchronization of CTRLB between clock domains is started.

Bit 1 – ENABLE ENABLE Synchronization Busy
This bit is cleared when the synchronization of ENABLE bit between the clock domains is complete.
This bit is set when the synchronization of ENABLE bit between clock domains is started.

Bit 0 – SWRST SWRST Synchronization Busy
This bit is cleared when the synchronization of SWRST bit between the clock domains is complete.
This bit is set when the synchronization of SWRST bit between clock domains is started.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST cleared by hardware.

40.7.6.13 Counter Value, 32-bit Mode

Name: COUNT
Offset: 0x14
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized

Note: Prior to any read access, this register must be synchronized by user by writing the according TC Command value to the Control B Set register (CTRLBSET.CMD=READSYNC).

Bit	31	30	29	28	27	26	25	24
	COUNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COUNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – COUNT[31:0] Counter Value
These bits contain the current counter value.

40.7.6.14 Channel x Compare/Capture Value, 32-bit Mode

Name: CCx
Offset: 0x1C + x*0x04 [x=0..1]
Reset: 0x00000000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	CC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CC[31:0] Channel x Compare/Capture Value
These bits contain the compare/capture value in 32-bit TC mode. In Match frequency (MFRQ) or Match PWM (MPWM) waveform operation (WAVE.WAVEGEN), the CC0 register is used as a period register.

40.7.6.15 Channel x Compare Buffer Value, 32-bit Mode

Name: CCBUFx
Offset: 0x30 + x*0x04 [x=0..1]
Reset: 0x00000000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
	CCBUF[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CCBUF[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CCBUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CCBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CCBUF[31:0] Channel x Compare Buffer Value
These bits hold the value of the Channel x Compare Buffer Value. When the buffer valid flag is '1' and double buffering is enabled (CTRLBCLR.LUPD=1), the data from buffer registers will be copied into the corresponding CCx register under UPDATE condition (CTRLBSET.CMD=0x3), including the software update command.

41. Timer/Counter for Control Applications (TCC)

41.1 Overview

The device provides three instances of the Timer/Counter for Control Applications (TCC).

Each TCC instance consists of a counter, a prescaler, compare/capture channels and control logic. The counter can be set to count events or clock pulses. The counter together with the compare/capture channels can be configured to time stamp input events, allowing capture of frequency and pulse-width. It can also perform waveform generation, such as frequency generation and pulse-width modulation.

Waveform extensions are featured for motor control, ballast, LED, H-bridge, power converters and other types of power control applications. They allow for low-side and high-side output with optional dead-time insertion. Waveform extensions can also generate a synchronized bit pattern across the waveform output pins. The fault options enable fault protection for safe and deterministic handling, disabling and/or shut-down of external drivers.

Note: The TCC configurations, such as channel numbers and features, may be reduced for some of the TCC instances.

Table 41-1. TCC Specific Configuration

					Extensions					
TCC No.	Counter Size (SIZE)	Host Link (Host_Client_MODE) (0 = NA, 1 = Host, 2 = Client)	Channels (CC_NUM)	Pins WO_NUM (OW_NUM)	Fault 1=YES	Dithering 1=YES	OutMatrix 1=YES (OTMX)	Dead Time Insertion 1=YES (DTI)	Swap 1=YES (SWAP)	Pattern Generation 1=YES (PG)
0	24	1	6	6	1	1	1	1	1	1
1	24	2	6	6	1	1	1	1	1	1
2	16	0	2	2	1	0	1	0	0	0

Note: Traditional Timer/Counter for Control Applications (TCC) documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

41.2 Features

- Compare/Capture Channels (CC) with:
  - Double buffered period setting
  - Double buffered compare or capture channel
  - Circular buffer on period and compare channel registers
- Waveform Generation:
  - Frequency generation
  - Single-slope pulse-width modulation (PWM)
  - Dual-slope PWM with half-cycle reload capability
- Input Capture:
  - Event capture
  - Frequency capture
  - Pulse-width capture
- Waveform Extensions:
  - Configurable distribution of compare channels outputs across port pins
  - Low-side and high-side output with programmable dead-time insertion
  - Waveform swap option with double buffer support

  - Pattern generation with double buffer support
  - Dithering support
- Fault Protection for Safe Disabling of Drivers:
  - Two recoverable fault sources
  - Two non-recoverable fault sources
  - Debugger can be a source of non-recoverable fault
- Input Events:
  - Two input events (EVx) for counter
  - One input event (MCx) for each channel
- Output Events:
  - Three output events (Count, re-trigger and overflow) are available for counter
  - One compare match/input capture event output for each channel
- Interrupts:
  - Overflow and re-trigger interrupt
  - Compare match/input capture interrupt
  - Interrupt on fault detection

41.3 Block Diagram

Figure 41-1. Timer/Counter for Control Applications - Block Diagram

41.4 Signal Description

Table 41-2. Signal Description

Pin Name	Type	Description
TCC/WO[0]	Digital output	Compare channel 0 waveform output
TCC/WO[1]	Digital output	Compare channel 1 waveform output
…	...	...
TCC/WO[WO_NUM-1]	Digital output	Compare channel n waveform output

See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

41.5 Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

41.5.1 I/O Lines

In order to use the I/O lines of this peripheral, the I/O pins must be configured using the I/O Peripheral Pin Select (PPS).

41.5.2 Power Management

This peripheral can continue to operate in any Sleep mode where its source clock is running. The interrupts can wake up the device from Sleep modes. Events connected to the event system can trigger other operations in the system without exiting Sleep modes.

41.5.3 Clocks

A generic clock (GCLK_TCCx) is required to clock the TCC. This clock must be configured and enabled in the generic clock controller before using the TCC. Note that TCC1 and TCC2 share a single peripheral clock generator.

The generic clocks (GCLK_TCCx) are asynchronous to the bus clock (PB1_CLK). Due to this asynchronicity, writing certain registers will require synchronization between the clock domains.

41.5.4 DMA

The DMA request lines are connected to the DMA Controller (DMAC). In order to use DMA requests with this peripheral, the DMAC must be configured first (see *Direct Memory Access Controller (DMAC)* from Related Links).

Related Links

22. Direct Memory Access Controller (DMAC)

41.5.5 Interrupts

The interrupt request line is connected to the Interrupt Controller. In order to use interrupt requests of this peripheral, the Interrupt Controller (NVIC) must be configured first. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)

41.5.6 Events

The events of this peripheral are connected to the Event System.

Related Links

28. Event System (EVSYS)

41.5.7 Debug Operation

When the CPU is halted in Debug mode, this peripheral will halt normal operation. This peripheral can be forced to continue operation during debugging - refer to the Debug Control (DBGCTRL) register for details.

Related Links

41.8.8. DBGCTRL

41.5.8 Register Access Protection

Registers with write access can be optionally write-protected by the Peripheral Access Controller (PAC), except for the following:

- Interrupt Flag register (INTFLAG)
- Status register (STATUS)
- Period and Period Buffer registers (PER, PERB)
- Compare/Capture and Compare/Capture Buffer registers (CCx, CCBx)
- Control Waveform register (WAVE)
- Pattern Generation Value and Pattern Generation Value Buffer registers (PATT, PATTB)

Note: Optional write protection is indicated by the "PAC Write Protection" property in the register description.

Write protection does not apply for accesses through an external debugger.

41.5.9 Analog Connections

Not applicable.

41.6 Functional Description

41.6.1 Principle of Operation

The following definitions are used throughout the documentation:

Table 41-3. Timer/Counter for Control Applications – Definitions

Name	Description
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. The TOP value can be the same as Period (PER) or the Compare Channel 0 (CC0) register value depending on the Waveform Generator mode in Waveform Output Operations. See *Waveform Output Generation Operations* from Related Links.
ZERO	The counter reaches ZERO when it contains all zeros.
MAX	The counter reaches maximum when it contains all ones.
UPDATE	The timer/counter signals an update when it reaches ZERO or TOP, depending on the direction settings.
Timer	The timer/counter clock control is handled by an internal source.
Counter	The clock control is handled externally (e.g., counting external events).
CC	For compare operations, the CC are referred to as "compare channels." For capture operations, the CC are referred to as "capture channels."

Each TCC instance has up to six compare/capture channels (CCx).

The Counter register (COUNT), Period registers with Buffer (PER and PERB), and Compare and Capture registers with buffers (CCx and CCBx) are 16- or 24-bit registers, depending on each TCC instance. Each Buffer register has a Buffer Valid (BUFV) flag that indicates when the buffer contains a new value.

Under normal operation, the counter value is continuously compared to the TOP or ZERO value to determine whether the counter has reached TOP or ZERO. In either case, the TCC can generate interrupt requests or generate events for the Event System. In Waveform Generator mode, these comparisons are used to set the waveform period or pulse width.

A prescaled generic clock (GCLK_TCCx) and events from the event system can be used to control the counter. The event system is also used as a source to the input capture.

The Recoverable Fault Unit enables event-controlled waveforms by acting directly on the generated waveforms of the TCC compare channels output. These events can restart, halt the timer/counter period, shorten the output pulse active time, or disable waveform output as long as the fault condition is present. This can typically be used for current sensing regulation, and zero-crossing and demagnetization re-triggering.

The MCE0 and MCE1 asynchronous event sources are shared with the recoverable fault unit. Only asynchronous events are used internally when fault unit extension is enabled. See *Event System (EVSYS)* from Related Links for further details on how to configure asynchronous events routing.

Recoverable fault sources can be filtered and/or windowed to avoid false triggering, for example from I/O pin glitches, by using digital filtering, input blanking and qualification options. See *Recoverable Faults* from Related Links.

In order to support applications with different types of motor control, ballast, LED, H-bridge, power converter and other types of power switching applications, the following independent units are implemented in some of the TCC instances as optional and successive units:

- Recoverable faults and non-recoverable faults
- Output matrix
- Dead-time insertion
- Swap
- Pattern generation

See *Timer/Counter for Control Applications - Block Diagram* in the *Block Diagram* from Related Links.

The output matrix (OTMX) can distribute and route out the TCC waveform outputs across the port pins in different configurations, each optimized for different application types. The Dead-Time Insertion (DTI) unit splits the four lower OTMX outputs into two non-overlapping signals: the non-inverted Low Side (LS) and inverted High Side (HS) of the

waveform output with optional dead-time insertion between LS and HS switching. The SWAP unit can swap the LS and HS pin outputs and can be used for fast decay motor control.

The pattern generation unit can be used to generate synchronized waveforms with constant logic level on TCC UPDATE conditions. This is useful for easy stepper motor and full bridge control.

The non-recoverable fault module enables event-controlled fault protection by acting directly on the generated waveforms of the timer/counter compare channel outputs. When a non-recoverable fault condition is detected, the output waveforms are forced to a preconfigured value that is safe for the application. This is typically used for instant and predictable shut-down and disabling high current or voltage drives.

The count event sources (TCE0 and TCE1) are shared with the non-recoverable fault extension. The events can be optionally filtered. If the filter options are not used, the non-recoverable faults provide an immediate asynchronous action on waveform output, even for cases where the clock is not present. See *Event System (EVSYS)* from Related Links for further details on how to configure asynchronous events routing.

Related Links

28. Event System (EVSYS)
40.6.2.6.1. Waveform Output Operations
41.3. Block Diagram
41.6.3.5. Recoverable Faults

41.6.2 Basic Operation

41.6.2.1 Initialization

The following registers are enable-protected, meaning that they can only be written when the TCC is disabled (CTRLA.ENABLE=0):

- Control A (CTRLA) register, except Run Standby (RUNSTDBY), Enable (ENABLE) and Software Reset (SWRST) bits
- Recoverable Fault n Control registers (FCTRLA and FCTRLB)
- Waveform Extension Control register (WEXCTRL)
- Drive Control register (DRVCTRL)
- Event Control register (EVCTRL)

Enable-protected bits in the CTRLA register can be written at the same time as CTRLA.ENABLE is written to '1', but not at the same time as CTRLA.ENABLE is written to '0'. Enable-protection is denoted by the "Enable-Protected" property in the register description.

Before the TCC is enabled, it must be configured as outlined by the following steps:

1. Enable the TCC bus clock if not already enabled by default (PB1_CLK).
2. If Capture mode is required, enable the channel in Capture mode by writing a '1' to the Capture Enable bit in the Control A register (CTRLA.CPTEN).

Optionally, the following configurations can be set before enabling TCC:

1. Select PRESCALER setting in the Control A register (CTRLA.PRESCALER).
2. Select Prescaler Synchronization setting in Control A register (CTRLA.PRESCSYNC).
3. If down-counting operation is desired, write the Counter Direction bit in the Control B Set register (CTRLBSET.DIR) to '1'.
4. Select the Waveform Generation operation in the WAVE register (WAVE.WAVEGEN).
5. Select the Waveform Output Polarity in the WAVE register (WAVE.POL).
6. The waveform output can be inverted for the individual channels using the Waveform Output Invert Enable bit group in the Driver register (DRVCTRL.INVEN).

41.6.2.2 Enabling, Disabling, and Resetting

The TCC is enabled by writing a '1' to the Enable bit in the Control A register (CTRLA.ENABLE). The TCC is disabled by writing a zero to CTRLA.ENABLE.

The TCC is reset by writing '1' to the Software Reset bit in the Control A register (CTRLA.SWRST). All registers in the TCC, except DBGCTRL, will be reset to their initial state, and the TCC will be disabled.

The TCC must be disabled before the TCC is reset to avoid undefined behavior.

41.6.2.3 Prescaler Selection

The GCLK_TCCx clock is fed into the internal prescaler.

The prescaler consists of a counter that counts up to the selected prescaler value, whereupon the output of the prescaler toggles.

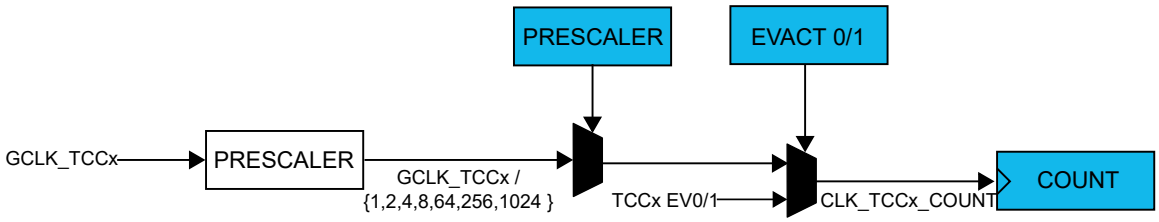
If the prescaler value is higher than one, the Counter Update condition can be optionally executed on the next GCLK_TCCx clock pulse or the next prescaled clock pulse. For further details, refer to the Prescaler (CTRLA.PRESCALER) and Counter Synchronization (CTRLA.PRESYNC) descriptions.

Prescaler outputs from 1 to 1/1024 are available. For a complete list of available prescaler outputs, see the register description for the Prescaler bit group in the Control A register (CTRLA.PRESCALER).

Note: When counting events, the prescaler is bypassed.

The joint stream of prescaler ticks and event action ticks is called CLK_TCCx_COUNT.

Figure 41-2. Prescaler



41.6.2.4 Counter Operation

Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each TCC clock input (CLK_TCCx_COUNT). A counter clear or reload mark the end of current counter cycle and the start of a new one.

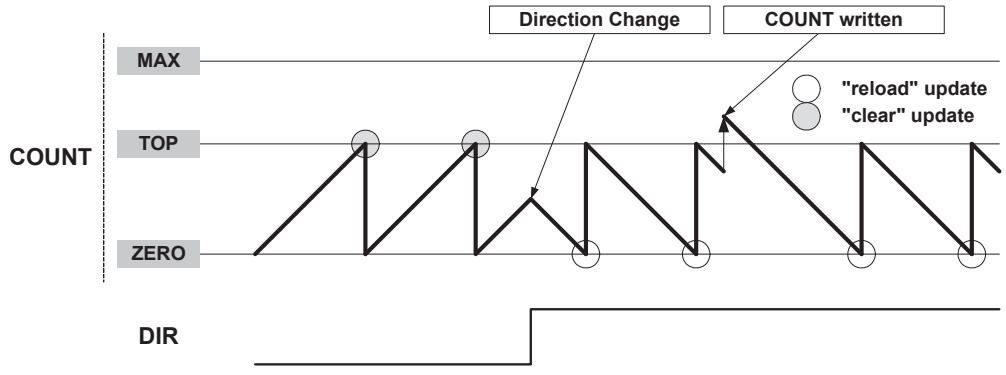
The counting direction is set by the Direction bit in the Control B register (CTRLB.DIR). If the bit is zero, it's counting up and one if counting down.

The counter will count up or down for each tick (clock or event) until it reaches TOP or ZERO. When it's counting up and TOP is reached, the counter will be set to zero at the next tick (overflow) and the Overflow Interrupt Flag in the Interrupt Flag Status and Clear register (INTFLAG.OVF) will be set. When down-counting, the counter is reloaded with the TOP value when ZERO is reached (underflow), and INTFLAG.OVF is set.

Note: When the TCC is counting down, the COUNT register must be initialized to the TOP value (PER or CC0 value depending on the mode).

INTFLAG.OVF can be used to trigger an interrupt, or an event. An overflow/underflow occurrence (i.e., a compare match with TOP/ZERO) will stop counting if the One-Shot bit in the Control B register is set (CTRLBSET.ONESHOT). The One-Shot feature is explained in the Additional Features section.

Figure 41-3. Counter Operation



Users can change the counter value (by writing directly in the COUNT register) even when the counter is running. The COUNT value will always be ZERO or TOP, depending on direction set by CTRLBSET.DIR or CTRLBCLR.DIR, when starting the TCC, unless a different value has been written to it, or the TCC has been stopped at a value other than ZERO. The write access has higher priority than count, clear, or reload. The direction of the counter can also be changed during normal operation.

Stop Command

A stop command can be issued from software by using TCC Command bits in the Control B Set register (CTRLBSET.CMD = 0x2, STOP).

Pause Event Action

A pause command can be issued when the stop event action is configured in the Input Event Action 1 bits in Event Control register (EVCTRL.EVACT1 = 0x3, STOP).

Re-Trigger Command and Event Action

A re-trigger command can be issued from software by using TCC Command bits in Control B Set register (CTRLBSET.CMD = 0x1, RETRIGGER), or from event when the re-trigger event action is configured in the Input Event 0/1 Action bits in Event Control register (EVCTRL.EVACTn = 0x1, RETRIGGER).

When the command is detected during counting operation, the counter will be reloaded or cleared, depending on the counting direction (CTRLBSET.DIR or CTRLBCLR.DIR). The Re-Trigger bit in the Interrupt Flag Status and Clear register will be set (INTFLAG.TRG). It is also possible to generate an event by writing a '1' to the Re-Trigger Event Output Enable bit in the Event Control register (EVCTRL.TRGEO). If the re-trigger command is detected when the counter is stopped, the counter will resume counting operation from the value in COUNT.

Note:
When a re-trigger event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACTn = 0x1, RETRIGGER), enabling the counter will not start the counter. The counter will start on the next incoming event and restart on corresponding following event.

Start Event Action

The start action can be selected in the Event Control register (EVCTRL.EVACT0 = 0x3, START) and can start the counting operation when previously stopped. The event has no effect if the counter is already counting. When the module is enabled, the counter operation starts when the event is received or when a re-trigger software command is applied.

Note:
When a start event action is configured in the Event Action bits in the Event Control register (EVCTRL.EVACT0 = 0x3, START), enabling the counter will not start the counter. The counter will start on the next incoming event, but it will not restart on subsequent events.

Count Event Action

The TCC can count events. When an event is received, the counter increases or decreases the value, depending on direction settings (CTRLBSET.DIR or CTRLBCLR.DIR).

The count event action is selected by the Event Action 0 bit group in the Event Control register (EVCTRL.EVACT0 = 0x5, COUNT).

Direction Event Action

The direction event action can be selected in the Event Control register (EVCTRL.EVACT1 = 0x2, DIR). When this event is used, the asynchronous event path specified in the event system must be configured or selected. The direction event action can be used to control the direction of the counter operation, depending on external events level. When received, the event level overrides the Direction settings (CTRLBSET.DIR or CTRLBCLR.DIR) and the direction bit value is updated accordingly.

Increment Event Action

The increment event action can be selected in the Event Control register (EVCTRL.EVACT0 = 0x4, INC) and can change the Counter state when an event is received. When the TCE0 event (TCCx_EV0) is received, the counter increments, whatever the direction setting (CTRLBSET.DIR or CTRLBCLR.DIR) is.

Decrement Event Action

The decrement event action can be selected in the Event Control register (EVCTRL.EVACT1 = 0x4, DEC) and can change the Counter state when an event is received. When the TCE1 (TCCx_EV1) event is received, the counter decrements, whatever the direction setting (CTRLBSET.DIR or CTRLBCLR.DIR) is.

Non-recoverable Fault Event Action

Non-recoverable fault actions can be selected in the Event Control register (EVCTRL.EVACTn = 0x7, FAULT). When received, the counter will be stopped and the output of the compare channels is overridden according to the Driver Control register settings (DRVCTRL.NREx and DRVCTRL.NRVx). TCE0 and TCE1 must be configured as asynchronous events.

Event Action Off

If the event action is disabled (EVCTRL.EVACTn = 0x0, OFF), enabling the counter will also start the counter.

Related Links

41.6.3.1. One-Shot Operation

41.6.2.5 Compare Operations

By default, the Compare/Capture channel is configured for compare operations. To perform capture operations, it must be re-configured.

When using the TCC with the Compare/Capture Value registers (CCx) for compare operations, the counter value is continuously compared to the values in the CCx registers. This can be used for timer or for waveform operation.

The Channel x Compare/Capture Buffer Value (CCBx) registers provide double buffer capability. The double buffering synchronizes the update of the CCx register with the buffer value at the UPDATE condition or a force update command (CTRLBSET.CMD=0x3, UPDATE). See *Double Buffering* from Related Links. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses and ensures glitch-free output.

Related Links

41.6.2.6. Double Buffering

41.6.2.5.1 Waveform Output Generation Operations

The compare channels can be used for waveform generation on output port pins. To make the waveform available on the connected pin, the following requirements must be fulfilled:

1. Choose a Waveform Generation mode in the Waveform Generation Operation bit in Waveform register (WAVE.WAVEGEN).
2. Optionally, invert the waveform output WO[x] by writing the corresponding Waveform Output x Inversion bit in the Driver Control register (DRVCTRL.INVENx).
3. Configure the pins with the I/O Pin Controller. See *I/O Ports and Peripheral Pin Select (PPS)* from Related Links.
   Note: Event must not be used when the compare channel is set in waveform output operating mode.

The counter value is continuously compared with each CCx value. On a comparison match, the Match or Capture Channel x bit in the Interrupt Flag Status and Clear register (INTFLAG.MCx) will be set on the next zero-to-one transition of CLK_TCC_COUNT (see Normal Frequency Operation). An interrupt and/or event can be generated on the same condition if Match/Capture occurs, i.e., INTENSET.MCx and/or EVCTRL.MCEOx is '1'. Both interrupt and event can be generated simultaneously.

There are seven waveform configurations for the Waveform Generation Operation bit group in the Waveform register (WAVE.WAVEGEN). This will influence how the waveform is generated and impose restrictions on the top value. The configurations are:

- Normal Frequency (NFRQ)
- Match Frequency (MFRQ)
- Normal Pulse-Width Modulation (NPWM)
- Dual-slope, interrupt/event at TOP (DSTOP)
- Dual-slope, interrupt/event at ZERO (DSBOTTOM)
- Dual-slope, interrupt/event at Top and ZERO (DSBOTH)
- Dual-slope, critical interrupt/event at ZERO (DSCRITICAL)

When using MFRQ configuration, the TOP value is defined by the CC0 register value. For the other waveform operations, the TOP value is defined by the Period (PER) register value.

For dual-slope waveform operations, the update time occurs when the counter reaches ZERO. For the other Waveforms Generation modes, the update time occurs on counter wraparound, on overflow, underflow or re-trigger.

The table below shows the update counter and overflow event/interrupt generation conditions in different operation modes.

Table 41-4. Counter Update and Overflow Event/interrupt Conditions

Name	Operation	TOP	Update	Output Waveform		OVFIF/Event	
				On Match	On Update	Up	Down
NFRQ	Normal Frequency	PER	TOP/ ZERO	Toggle	Stable	TOP	ZERO
MFRQ	Match Frequency	CC0	TOP/ ZERO	Toggle	Stable	TOP	ZERO
NPWM	Single-slope PWM	PER	TOP/ ZERO	See section 'Output Polarity' below		TOP	ZERO
DSCRITICAL	Dual-slope PWM	PER	ZERO			—	ZERO
DSBOTTOM	Dual-slope PWM	PER	ZERO			—	ZERO
DSBOTH	Dual-slope PWM	PER	TOP(1) & ZERO			TOP	ZERO
DSTOP	Dual-slope PWM	PER	ZERO			TOP	—

1. The UPDATE condition on TOP only will occur when circular buffer is enabled for the channel.

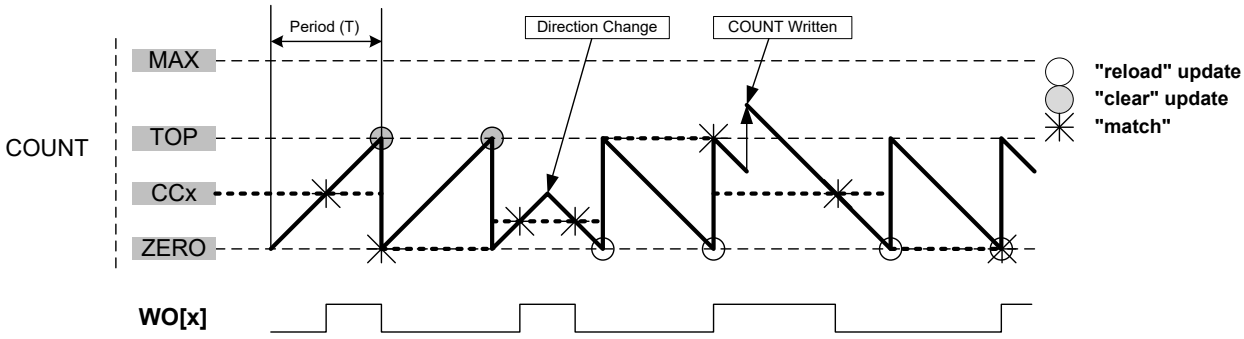
Related Links

6. I/O Ports and Peripheral Pin Select (PPS)

41.6.2.5.2 Normal Frequency (NFRQ)

For Normal Frequency generation, the period time (T) is controlled by the period register (PER). The waveform generation output (WO[x]) is toggled on each compare match between COUNT and CCx, and the corresponding Match or Capture Channel x Interrupt Flag (EVCTRL.MCEOx) will be set.

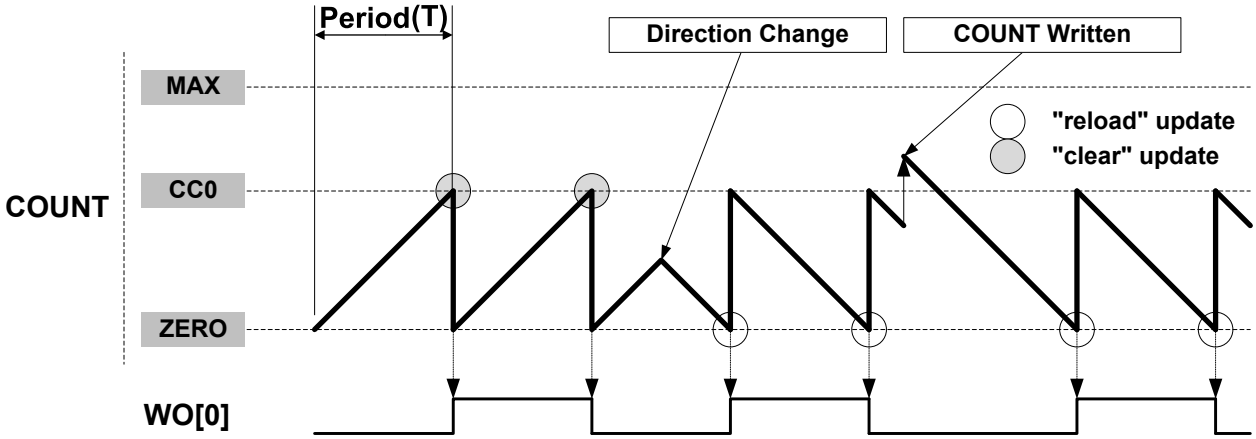
Figure 41-4. Normal Frequency Operation



41.6.2.5.3 Match Frequency (MFRQ)

For Match Frequency generation, the period time (T) is controlled by CC0 register instead of PER. WO[0] toggles on each update condition.

Figure 41-5. Match Frequency Operation



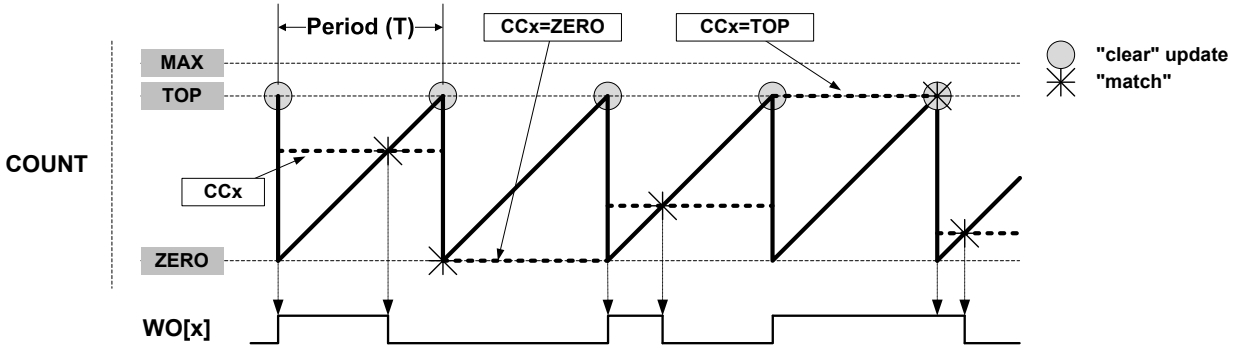
41.6.2.5.4 Normal Pulse-Width Modulation (NPWM)

NPWM uses single-slope PWM generation.

41.6.2.5.5 Single-Slope PWM Operation

For single-slope PWM generation, the period time (T) is controlled by Top value, and CCx controls the duty cycle of the generated waveform output. When up-counting, the WO[x] is set at start or compare match between the COUNT and TOP values, and cleared on compare match between COUNT and CCx register values. When down-counting, the WO[x] is cleared at start or compare match between the COUNT and ZERO values, and set on compare match between COUNT and CCx register values.

Figure 41-6. Single-Slope PWM Operation



The following equation calculates the exact resolution for a single-slope PWM (R_{PWM_SS}) waveform:

$$R_{PWM\_SS} = \frac{\log(TOP+1)}{\log(2)}$$

The PWM frequency depends on the Period register value (PER) and the peripheral clock frequency (f_{GCLK_TCCx}), and can be calculated by the following equation:

$$f_{PWM\_SS} = \frac{f_{GCLK\_TCCx}}{N(TOP+1)}$$

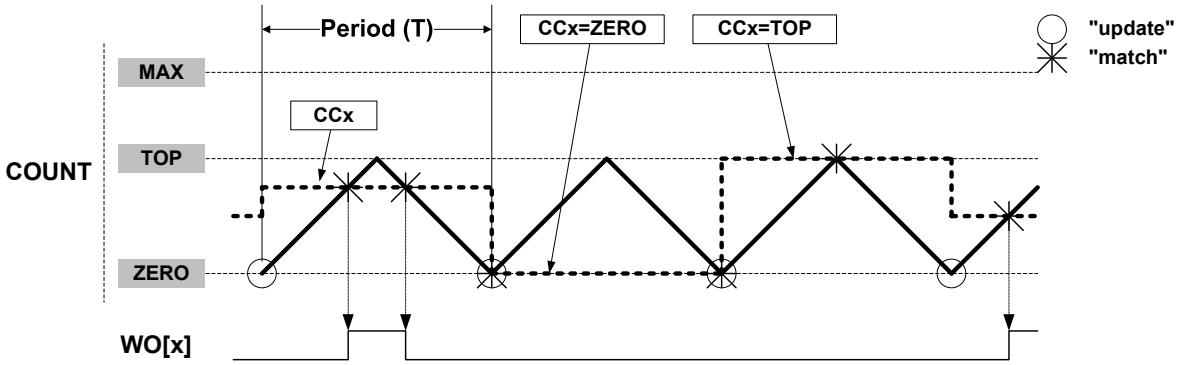
Where N represents the prescaler divider used (1, 2, 4, 8, 16, 64, 256, 1024).

41.6.2.5.6 Dual-Slope PWM Generation

For dual-slope PWM generation, the period setting (TOP) is controlled by PER, while CCx control the duty cycle of the generated waveform output. The figure below shows how the counter repeatedly counts from ZERO to PER and then from PER to ZERO. The waveform generator output is set on compare match when up-counting, and cleared on compare match when down-counting. An interrupt and/or event is generated on TOP (when counting upwards) and/or ZERO (when counting up or down).

In DSBOTH operation, the circular buffer must be enabled to enable the update condition on TOP.

Figure 41-7. Dual-Slope Pulse Width Modulation



Using dual-slope PWM results in a lower maximum operation frequency compared to single-slope PWM generation. The period (TOP) defines the PWM resolution. The minimum resolution is 1 bit (TOP=0x00000001).

The following equation calculates the exact resolution for dual-slope PWM (R_{PWM_DS}):

$$R_{PWM\_DS} = \frac{\log(PER+1)}{\log(2)}.$$

The PWM frequency f_{PWM_DS} depends on the period setting (TOP) and the peripheral clock frequency f_{GCLK_TCCx}, and can be calculated by the following equation:

$$f_{PWM\_DS} = \frac{f_{GCLK\_TCCx}}{2N \cdot PER}$$

N represents the prescaler divider used. The waveform generated will have a maximum frequency of half of the TCC clock frequency (f_{GCLK_TCCx}) when TOP is set to 0x00000001 and no prescaling is used.

The pulse width (P_{PWM_DS}) depends on the compare channel (CCx) register value and the peripheral clock frequency (f_{GCLK_TCCx}), and can be calculated by the following equation:

$$P_{PWM\_DS} = \frac{2N \cdot (TOP - CCx)}{f_{GCLK\_TCCx}}$$

N represents the prescaler divider used.

Note: In DSTOP, DSBOTTOM and DSBOTH operation, when TOP is lower than MAX/2, the CCx MSB bit defines the ramp on which the CCx Match interrupt or event is generated. (Rising if CCx[MSB] = 0, falling if CCx[MSB] = 1.)

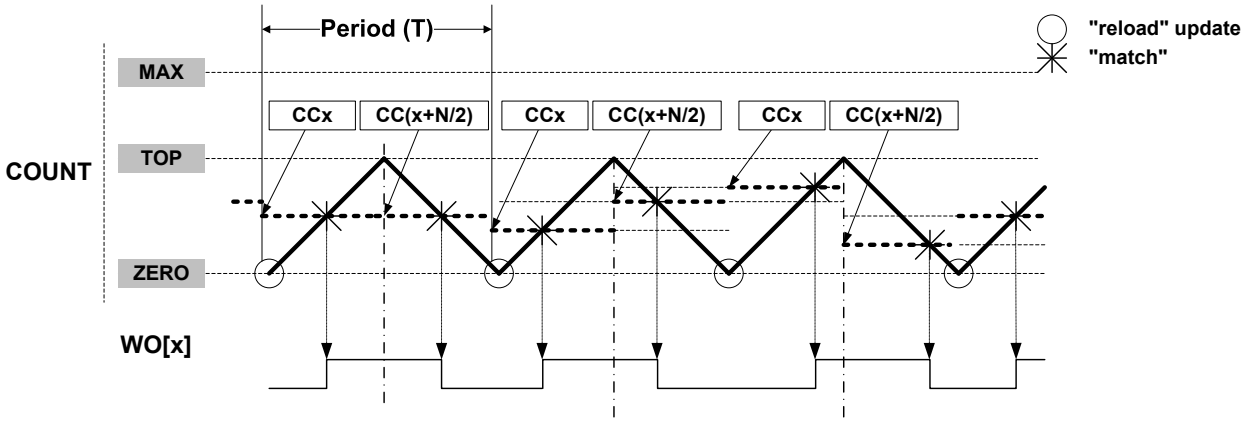
Related Links

41.6.3.2. Circular Buffer

41.6.2.5.7 Dual-Slope Critical PWM Generation

Critical mode generation allows generation of non-aligned centered pulses. In this mode, the period time is controlled by PER while CCx control the generated waveform output edge during up-counting and CC(x+CC_NUM/2) control the generated waveform output edge during down-counting.

Figure 41-8. Dual-Slope Critical Pulse Width Modulation (N=CC_NUM)



41.6.2.5.8 Output Polarity

The polarity (WAVE.POLx) is available in all waveform output generation. In single-slope and dual-slope PWM operation, it is possible to invert the pulse edge alignment individually on start or end of a PWM cycle for each compare channels. The table below shows the waveform output set/clear conditions, depending on the settings of timer/counter, direction, and polarity.

Table 41-5. Waveform Generation Set/Clear Conditions

Waveform Generation Operation	DIR	POLx	Waveform Generation Output Update	
			Set	Clear
Single-Slope PWM	0	0	Timer/counter matches TOP	Timer/counter matches CCx
		1	Timer/counter matches CC	Timer/counter matches TOP
	1	0	Timer/counter matches CC	Timer/counter matches ZERO
		1	Timer/counter matches ZERO	Timer/counter matches CC
Dual-Slope PWM	x	0	Timer/counter matches CC when counting up	Timer/counter matches CC when counting down
		1	Timer/counter matches CC when counting down	Timer/counter matches CC when counting up

In Normal and Match Frequency, the WAVE.POLx value represents the initial state of the waveform output.

41.6.2.6 Double Buffering

The Pattern (PATT), Period (PER) and Compare Channels (CCx) registers are all double buffered. Each buffer register has a buffer valid (PATTBV, PERBV and CCBVx) bit in the STATUS register that indicates that the Buffer register contains a valid value that can be copied into the corresponding register. .

When the Buffer Valid Flag bit in the STATUS register is '1' and the Lock Update bit in the CTRLB register is set to '0' (writing CTRLBCLR.LUPD to '1'), double buffering is enabled: the data from the buffer registers will be copied into the corresponding register under the hardware UPDATE conditions, then the Buffer Valid flags bit in the STATUS register is automatically cleared by the hardware.
Note: The software update command (CTRLBSET.CMD=0x3) acts independently of the LUPD value.

A compare register is double buffered as in the following figure.

Figure 41-9. Compare Channel Double Buffering

Both the registers (PATT/PER/CCx) and corresponding Buffer registers are available in the I/O register map, and the double buffering feature is not mandatory. The double buffering is disabled by writing a '1' to CTRLSET.LUPD.

Note: In NFRQ, MFRQ or PWM Down-Counting Counter mode (CTRLBSET.DIR=1), when double buffering is enabled (CTRLBCLR.LUPD=1), the PERB register is continuously copied into the PER independently of the update conditions.

Changing the Period

The counter period can be changed by writing a new Top value to the Period register (PER or CC0, depending on the Waveform Generation mode). Any period update on the registers (PER or CCx) is effective after the synchronization delay, whatever double buffering enabling is.

Figure 41-10. Unbuffered Single-Slope Up-Counting Operation

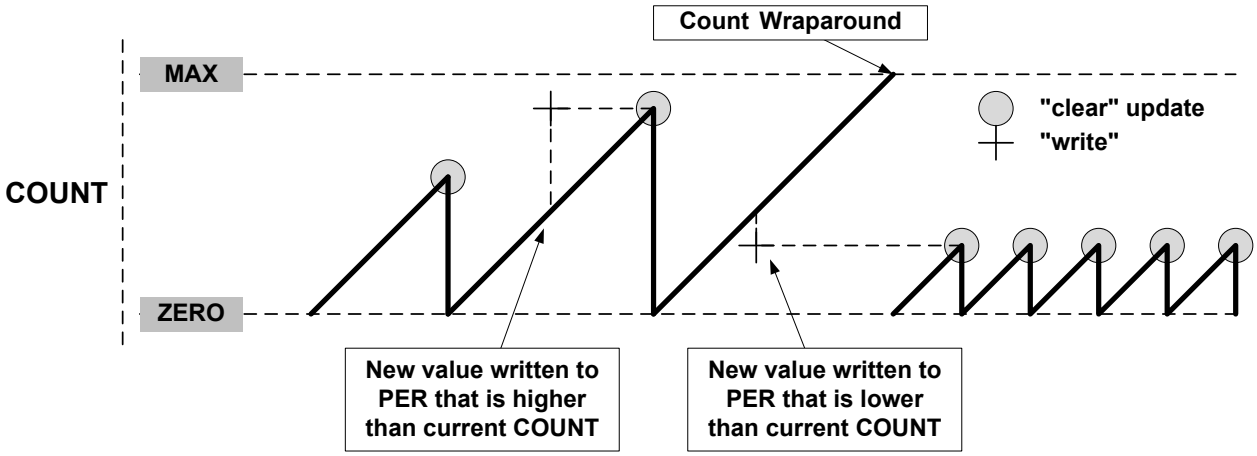
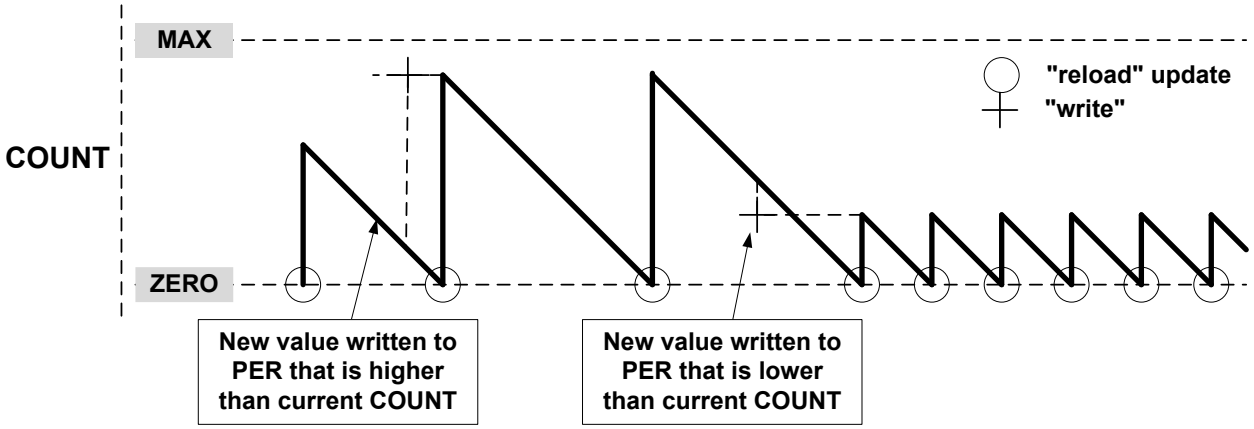
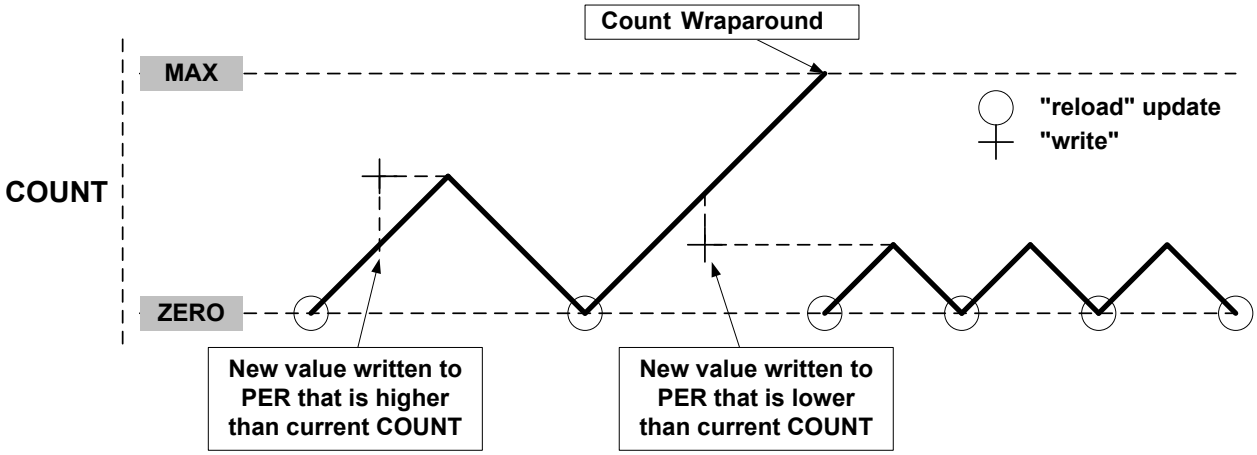


Figure 41-11. Unbuffered Single-Slope Down-Counting Operation



A counter wraparound can occur in any operation mode when up-counting without buffering (see Figure 41-10). COUNT and TOP are continuously compared, so when a new value that is lower than the current COUNT is written to TOP, COUNT will wrap before a compare match.

Figure 41-12. Unbuffered Dual-Slope Operation



When double buffering is used, the buffer can be written at any time and the counter will still maintain correct operation. The period register is always updated on the update condition, as shown in Figure 41-13. This prevents wraparound and the generation of odd waveforms.

Figure 41-13. Changing the Period Using Buffering

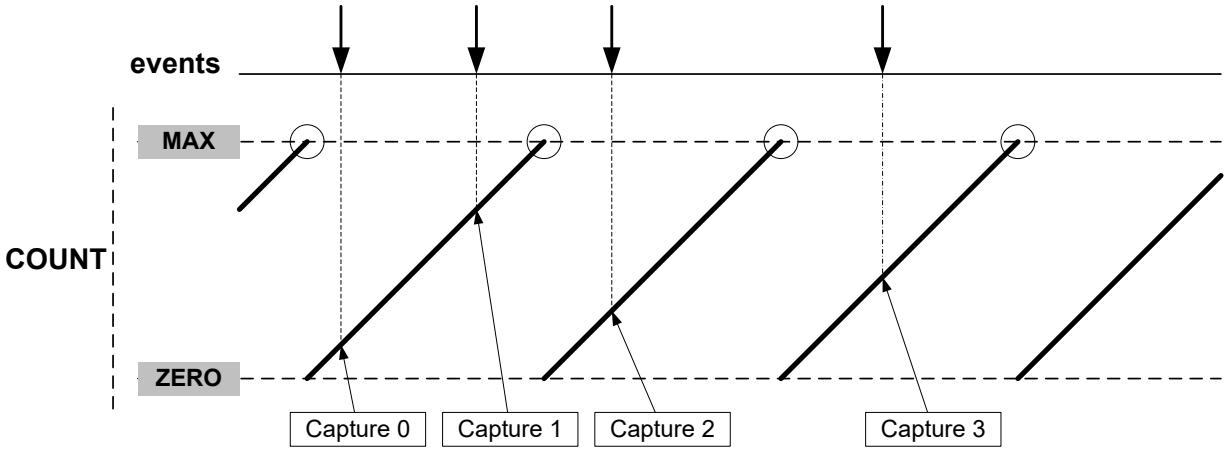
41.6.2.7 Capture Operations

To enable and use capture operations, the Match or Capture Channel x Event Input Enable bit in the Event Control register (EVCTRL.MCEIx) must be written to '1'. The capture channels to be used must also be enabled in the Capture Channel x Enable bit in the Control A register (CTRLA.CPTENx) before capturing can be performed.

Event Capture Action

The compare/capture channels can be used as input capture channels to capture events from the Event System, and give them a timestamp. The following figure shows four capture events for one capture channel. Event system channels must be configured to operate in asynchronous mode when used for capture operations.

Figure 41-14. Input Capture Timing



For input capture, the Buffer register and the corresponding CCx act like a FIFO. When CCx is empty or read, any content in CCBx is transferred to CCx. The Buffer Valid flag is passed to set the CCx Interrupt flag (IF) and generate the optional interrupt, event, or DMA request. The CCBx register value cannot be read, all captured data must be read from the CCx register.

Figure 41-15. Capture Double Buffering

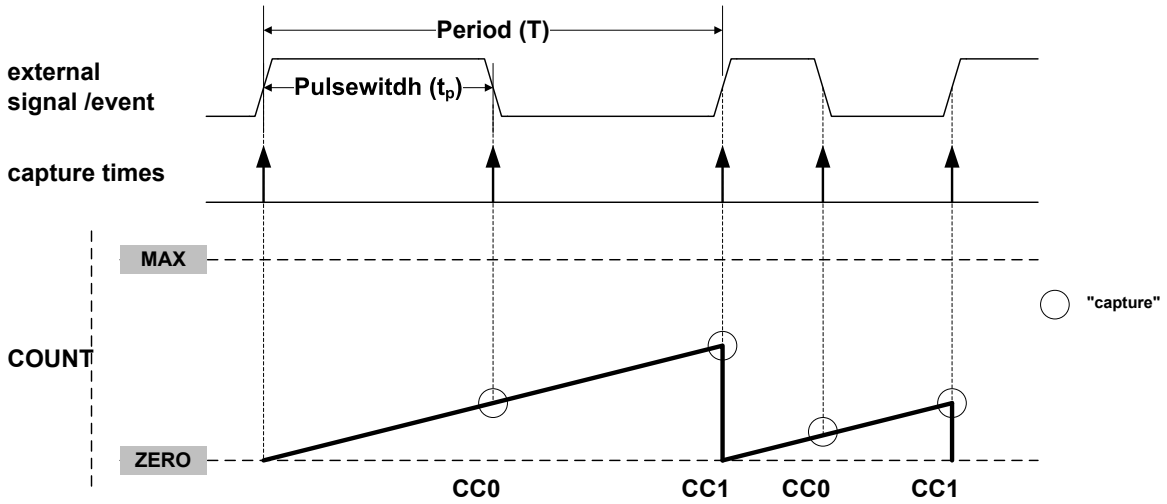
The TCC can detect capture overflow of the input capture channels. When a new capture event is detected while the Capture Buffer Valid flag (STATUS.CCBV) is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

Period and Pulse-Width (PPW) Capture Action

The TCC can perform two input captures and restart the counter on one of the edges. This enables the TCC to measure the pulse-width and period and to characterize the frequency *f* and *dutyCycle* of an input signal, as shown below:

$$f = \frac{1}{T} \quad , \quad dutyCycle = \frac{t_p}{T}$$

Figure 41-16. PWP Capture



Selecting PWP or PPW in the Timer/Counter Event Input 1 Action bit group in the Event Control register (EVCTRL.EVACT1) enables the TCC to perform one capture action on the rising edge and the other one on the falling edge. When using PPW event action, period *T* will be captured into CC0 and the pulse-width t_p into CC1. The PWP (Pulse-width and Period) event action offers the same functionality, but *T* will be captured into CC1 and t_p into CC0.

The Timer/Counter Event x Invert Enable bit in Event Control register (EVCTRL.TCEINVx) is used for event source x to select whether the wraparound must occur on the rising edge or the falling edge. If EVCTRL.TCEINVx=1, the wraparound will happen on the falling edge.

The corresponding capture is done only if the channel is enabled in Capture mode (CTRLA.CPTENx=1). If not, the capture action will be ignored and the channel will be enabled in compare mode of operation. When only one of these channel is required, the other channel can be used for other purposes.

The TCC can detect capture overflow of the input capture channels. When a new capture event is detected while the INTFLAG.MCx is still set, the new timestamp will not be stored and INTFLAG.ERR will be set.

Note: When up-counting (CTRLBSET.DIR=0), counter values lower than 1 cannot be captured in Capture Minimum mode (FCTRLn.CAPTURE=CAPTMIN). To capture the full range including value 0, the TCC must be configured in Down-counting mode (CTRLBSET.DIR=0).

Note: In dual-slope PWM operation, and when TOP is lower than MAX/2, the CCx MSB captures the CTRLB.DIR state to identify the ramp on which the capture has been done. For rising ramps CCx[MSB] is zero, for falling ramps CCx[MSB]=1.

41.6.3 Additional Features

41.6.3.1 One-Shot Operation

When one-shot is enabled, the counter automatically stops on the next Counter Overflow or Underflow condition. When the counter is stopped, the Stop bit in the Status register (STATUS.STOP) is set and the waveform outputs are set to the value defined by DRVCTRL.NREx and DRVCTRL.NRVx.

One-shot operation can be enabled by writing a '1' to the One-Shot bit in the Control B Set register (CTRLBSET.ONESHOT) and disabled by writing a '1' to CTRLBCLR.ONESHOT. When enabled, the TCC will count until an overflow or underflow occurs and stop counting. The one-shot operation can be restarted by a re-trigger

software command, a re-trigger event or a start event. When the counter restarts its operation, STATUS.STOP is automatically cleared.

41.6.3.2 Circular Buffer

The Period register (PER) and the Compare Channels register (CC0 to) support circular buffer operation. When circular buffer operation is enabled, the PER or CCx values are copied into the corresponding buffer registers at each update condition. Circular buffering is dedicated to RAMP2, RAMP2A, and DSBOTH operations.

Figure 41-17. Circular Buffer on Channel 0

41.6.3.3 Dithering Operation

The TCC supports dithering on Pulse-width or Period on a 16, 32 or 64 PWM cycles frame.

Dithering consists in adding some extra clocks cycles in a frame of several PWM cycles, and can improve the accuracy of the *average* output pulse width and period. The extra clock cycles are added on some of the compare match signals, one at a time, through a "blue noise" process that minimizes the flickering on the resulting dither patterns.

Dithering is enabled by writing the corresponding configuration in the Enhanced Resolution bits in CTRLA register (CTRLA.RESOLUTION):

- DITH4 enable dithering every 16 PWM frames
- DITH5 enable dithering every 32 PWM frames
- DITH6 enable dithering every 64 PWM frames

The DITHERCY bits of COUNT, PER and CCx define the number of extra cycles to add into the frame (DITHERCY bits from the respective COUNT, PER or CCx registers). The remaining bits of COUNT, PER, CCx define the compare value itself.

The pseudo code, giving the extra cycles insertion regarding the cycle is:

```
int extra_cycle(resolution, dithercy, cycle){
  int MASK;
  int value
  switch (resolution){
    DITH4: MASK = 0x0f;
    DITH5: MASK = 0x1f;
    DITH6: MASK = 0x3f;
  }
  value = cycle * dithercy;
  if (((MASK & value) + dithercy) > MASK)
    return 1;
 return 0;
}
```

Dithering on Period

Writing DITHERCY in PER will lead to an average PWM period configured by the following formulas.

DITH4 mode:

$$PwmPeriod = \left(\frac{\mathrm{DITHERCY}}{16} + \mathrm{PER}\right)\left(\frac{1}{f_{\mathrm{GCLK\_TCC}x}}\right)$$

Note: If DITH4 mode is enabled, the last 4 significant bits from PER/CCx or COUNT register correspond to the DITHERCY value, rest of the bits corresponds to PER/CCx or COUNT value.

DITH5 mode:

$$PwmPeriod = \left(\frac{\mathrm{DITHERCY}}{32} + \mathrm{PER}\right)\left(\frac{1}{f_{\mathrm{GCLK\_TCC}x}}\right)$$

DITH6 mode:

$$PwmPeriod = \left(\frac{\mathrm{DITHERCY}}{64} + \mathrm{PER}\right)\left(\frac{1}{f_{\mathrm{GCLK\_TCC}x}}\right)$$

Dithering on Pulse-Width

Writing DITHERCY in CCx will lead to an average PWM pulse width configured by the following formula.

DITH4 mode:

$$PwmPulseWidth = \left(\frac{\text{DITHERCY}}{16} + \text{CCx}\right)\left(\frac{1}{f_{\text{GCLK\_TCC}x}}\right)$$

DITH5 mode:

$$PwmPulseWidth = \left(\frac{\text{DITHERCY}}{32} + \text{CCx}\right)\left(\frac{1}{f_{\text{GCLK\_TCC}x}}\right)$$

DITH6 mode:

$$PwmPulseWidth = \left(\frac{\text{DITHERCY}}{64} + \text{CCx}\right)\left(\frac{1}{f_{\text{GCLK\_TCC}x}}\right)$$

Note: The PWM period will remain static in this case.

41.6.3.4 Ramp Operations

Three ramp operation modes are supported. All of them require the timer/counter running in single-slope PWM generation. The Ramp mode is selected by writing to the Ramp Mode bits in the Waveform Control register (WAVE.RAMP).

RAMP1 Operation

This is the default PWM operation.

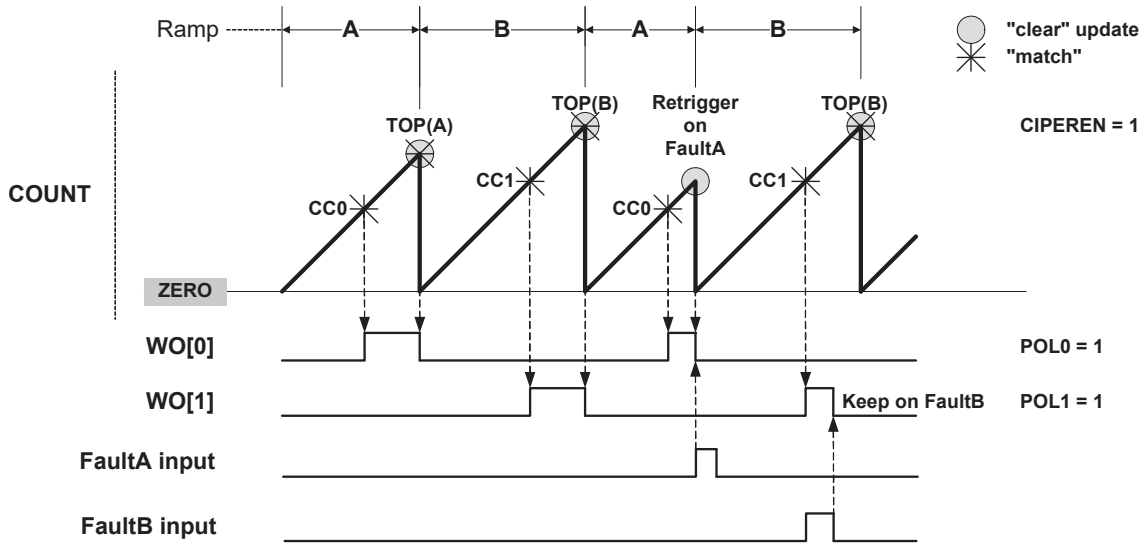
RAMP2 Operation

These operation modes are dedicated for power factor correction (PFC), Half-Bridge and Push-Pull SMPS topologies, where two consecutive timer/counter cycles are interleaved (see the following figure). In cycle A, odd channel output is disabled, and in cycle B, even channel output is disabled. The ramp index changes after each update, but can be software modified using the Ramp index command bits in the Control B Set register (CTRLBSET.IDXCMD).

Standard RAMP2 (RAMP2) Operation

Ramp A and B periods are controlled by the PER register value. The PER value can be different on each ramp by the Circular Period buffer option in the Wave register (WAVE.CIPEREN=1). This mode uses a two-channel TCC to generate two output signals, or one output signal with another CC channel enabled in Capture mode.

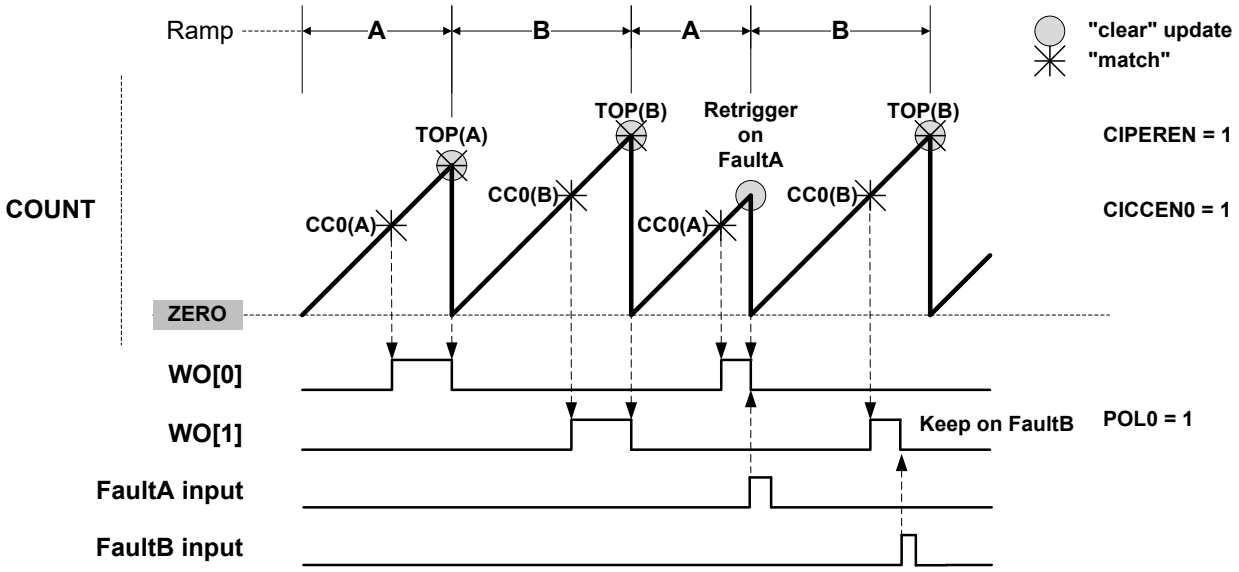
Figure 41-18. RAMP2 Standard Operation



Alternate RAMP2 (RAMP2A) Operation

Alternate RAMP2 operation is similar to RAMP2, but CC0 controls both WO[0] and WO[1] waveforms when the corresponding circular buffer option is enabled (CIPEREN=1). The waveform polarity is the same on both outputs. Channel 1 can be used in capture mode.

Figure 41-19. RAMP2 Alternate Operation



41.6.3.5 Recoverable Faults

Recoverable faults can restart or halt the timer/counter. Two faults, called Fault A and Fault B, can trigger recoverable fault actions on the compare channels CC0 and CC1 of the TCC. The compare channels' outputs can be clamped to inactive state either as long as the fault condition is present, or from the first valid fault condition detection on until the end of the timer/counter cycle.

Fault Inputs

The first two channel input events (TCCxMC0 and TCCxMC1) can be used as Fault A and Fault B inputs, respectively. Event system channels connected to these fault inputs must be configured as asynchronous. The TCC must work in a PWM mode.

Fault Filtering

There are three filters available for each input Fault A and Fault B. They are configured by the corresponding Recoverable Fault n Configuration registers (FCTRLA and FCTRLB). The three filters can either be used independently or in any combination.

Input Filtering By default, the event detection is asynchronous. When the event occurs, the fault system will immediately and asynchronously perform the selected fault action on the compare channel output, also in device power modes where the clock is not available. To avoid false fault detection on external events (e.g. due to a glitch on an I/O port) a digital filter can be enabled and configured by the Fault B Filter Value bits in the Fault n Configuration registers (FCTRLn.FILTERVAL). If the event width is less than FILTERVAL (in clock cycles), the event will be discarded. A valid event will be delayed by FILTERVAL clock cycles.

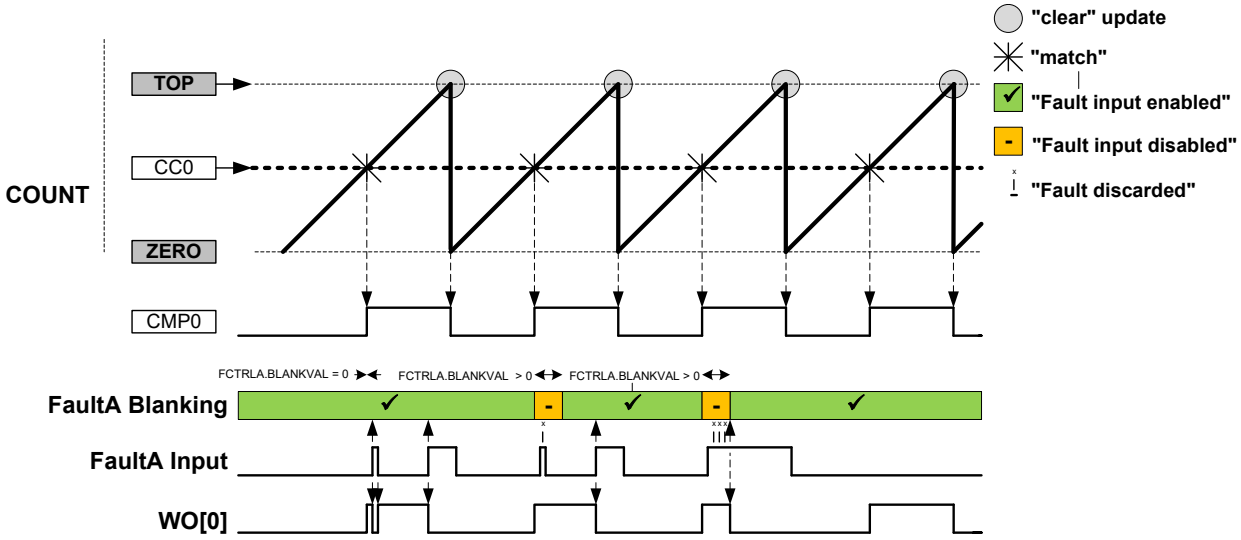
Fault Blanking This ignores any fault input for a certain time just after a selected waveform output edge. This can be used to prevent false fault triggering due to signal bouncing, as shown in the figure below. Blanking can be enabled by writing an edge triggering configuration to the Fault n Blanking Mode bits in the Recoverable Fault n Configuration register (FCTRLn.BLANK). The desired duration of the blanking must be written to the Fault n Blanking Time bits (FCTRLn.BLANKVAL).
The blanking time $t_b$is calculated by

$$t_b = \frac{1 + \mathrm{BLANKVAL}}{f_{\mathrm{GCLK\_TCCx\_PRESC}}}$$

Here, $f_{GCLK\_TCCx\_PRESC}$ is the frequency of the prescaled peripheral clock frequency f_{GCLK_TCCx}.

The maximum blanking time (FCTRLn.BLANKVAL= 255) at f_{GCLK_TCCx}=96MHz is 2.67µs (no prescaler) or 170µs (prescaling). For f_{GCLK_TCCx}=1MHz, the maximum blanking time is either 170µs (no prescaling) or 10.9ms (prescaling enabled).

Figure 41-20. Fault Blanking in RAMP1 Operation with Inverted Polarity



Fault Qualification

This is enabled by writing a '1' to the Fault n Qualification bit in the Recoverable Fault n Configuration register (FCTRLn.QUAL). When the recoverable fault qualification is enabled (FCTRLn.QUAL=1), the fault input is disabled all the time the corresponding channel output has an inactive level, as shown in the figures below.

Figure 41-21. Fault Qualification in RAMP1 Operation

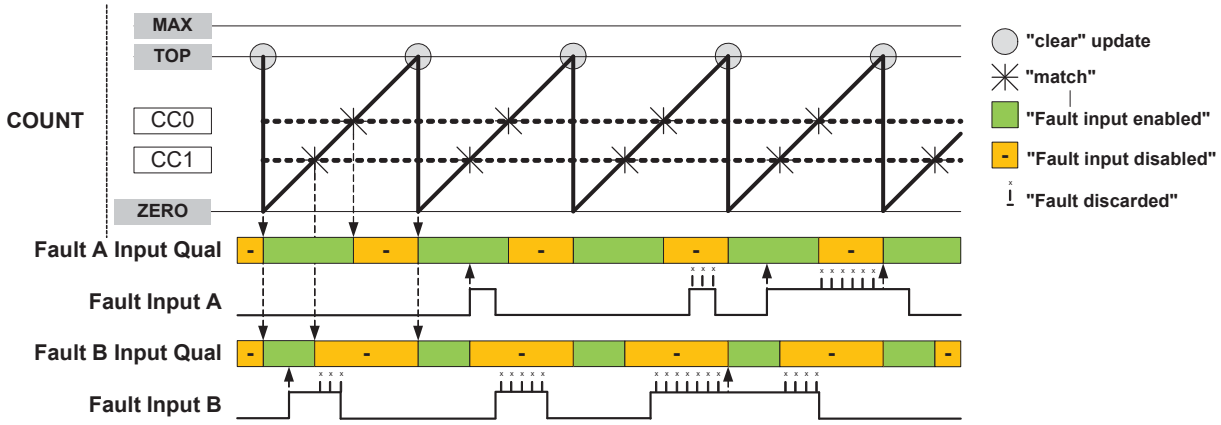
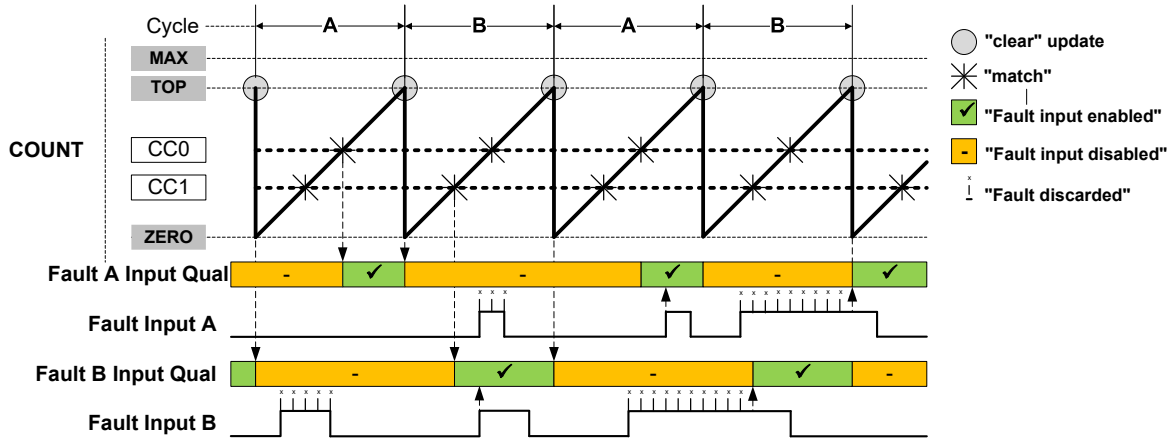


Figure 41-22. Fault Qualification in RAMP2 Operation with Inverted Polarity



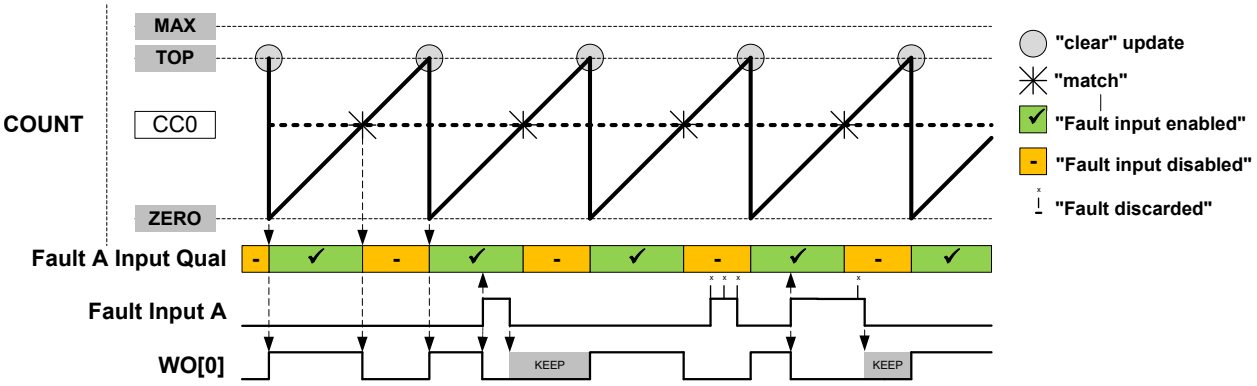
Fault Actions

Different fault actions can be configured individually for Fault A and Fault B. Most fault actions are not mutually exclusive; hence two or more actions can be enabled at the same time to achieve a result that is a combination of fault actions.

Keep Action

This is enabled by writing the Fault n Keeper bit in the Recoverable Fault n Configuration register (FCTRLn.KEEP) to '1'. When enabled, the corresponding channel output will be clamped to zero as long as the fault condition is present. The clamp will be released on the start of the first cycle after the fault condition is no longer present, see next Figure.

Figure 41-23. Waveform Generation with Fault Qualification and Keep Action



Restart Action

This is enabled by writing the Fault n Restart bit in Recoverable Fault n Configuration register (FCTRLn.RESTART) to '1'. When enabled, the timer/counter will be restarted as soon as the corresponding fault condition is present. The ongoing cycle is stopped and the timer/counter starts a new cycle, see Figure 41-24. In Ramp 1 mode, when the new cycle starts, the compare outputs will be clamped to inactive level as long as the fault condition is present.

Note: For RAMP2 operation, when a new timer/counter cycle starts the cycle index will change automatically, see Figure 41-25. Fault A and Fault B are qualified only during the cycle A and cycle B respectively: Fault A is disabled during cycle B, and Fault B is disabled during cycle A.

Figure 41-24. Waveform Generation in RAMP1 mode with Restart Action

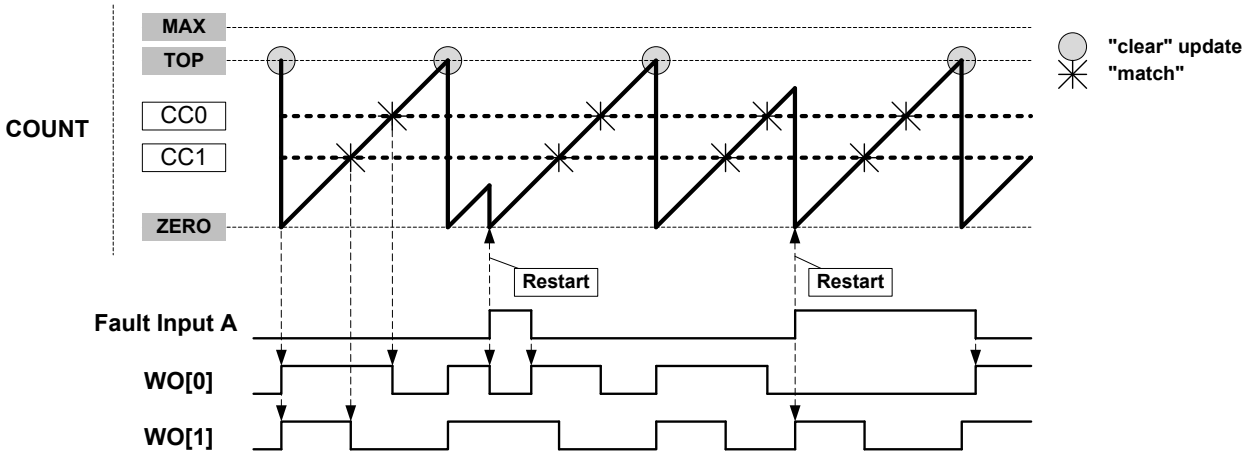
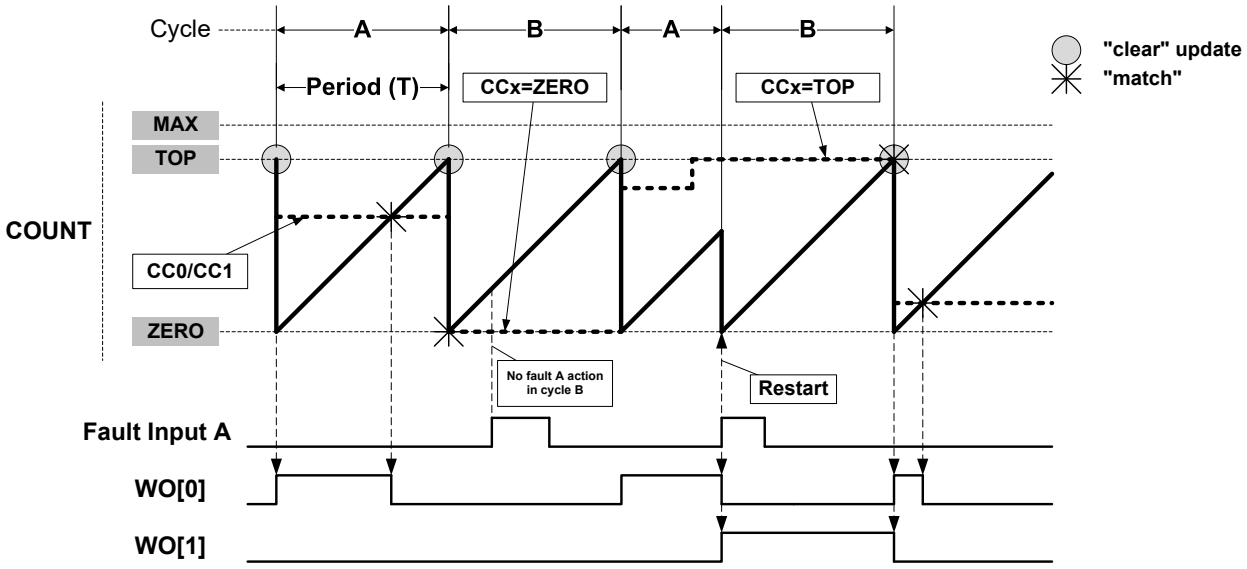


Figure 41-25. Waveform Generation in RAMP2 mode with Restart Action



Capture Action

Several capture actions can be selected by writing the Fault n Capture Action bits in the Fault n Control register (FCTRLn.CAPTURE). When one of the capture operations is selected, the counter value is captured when the fault occurs. These capture operations are available:

- CAPT - the equivalent to a standard capture operation.
- CAPTMIN - gets the minimum time stamped value: on each new local minimum captured value, an event or interrupt is issued.
- CAPTMAX - gets the maximum time stamped value: on each new local maximum captured value, an event or interrupt (IT) is issued, see Figure 41-26.
- LOCMIN - notifies by event or interrupt when a local minimum captured value is detected.
- LOCMAX - notifies by event or interrupt when a local maximum captured value is detected.
- DERIV0 - notifies by event or interrupt when a local extreme captured value is detected, see Figure 41-27.

CCx Content:

In CAPTMIN and CAPTMAX operations, CCx keeps the respective extremum captured values, see Figure 41-26. In LOCMIN, LOCMAX or DERIV0 operation, CCx follows the counter value at fault time, see Figure 41-27.

Before enabling CAPTMIN or CAPTMAX mode of capture, the user must initialize the corresponding CCx register value to a value different from zero (for CAPTMIN) top (for CAPTMAX). If the CCx register initial value is zero (for CAPTMIN) top (for CAPTMAX), no captures will be performed using the corresponding channel.

MCx Behaviour:

In LOCMIN and LOCMAX operation, capture is performed on each capture event. The MCx interrupt flag is set only when the captured value is above or equal (for LOCMIN) or below or equal (for LOCMAX) to the previous captured value. So interrupt flag is set when a new relative local Minimum (for CAPTMIN) or Maximum (for CAPTMAX) value has been detected. DERIV0 is equivalent to an OR function of (LOCMIN, LOCMAX).

In CAPT operation, capture is performed on each capture event. The MCx interrupt flag is set on each new capture.

In CAPTMIN and CAPTMAX operation, capture is performed only when on capture event time, the counter value is lower (for CAPTMIN) or higher (for CAPMAX) than the last captured value. The MCx interrupt flag is set only when on capture event time, the counter value is higher or equal (for CAPTMIN) or lower or equal (for CAPTMAX) to the value captured on the previous event. So interrupt flag is set when a new absolute local Minimum (for CAPTMIN) or Maximum (for CAPTMAX) value has been detected.

Interrupt Generation

In CAPT mode, an interrupt is generated on each filtered Fault n and each dedicated CCx channel capture counter value. In other modes, an interrupt is only generated on an extreme captured value.

Figure 41-26. Capture Action “CAPTMAX”

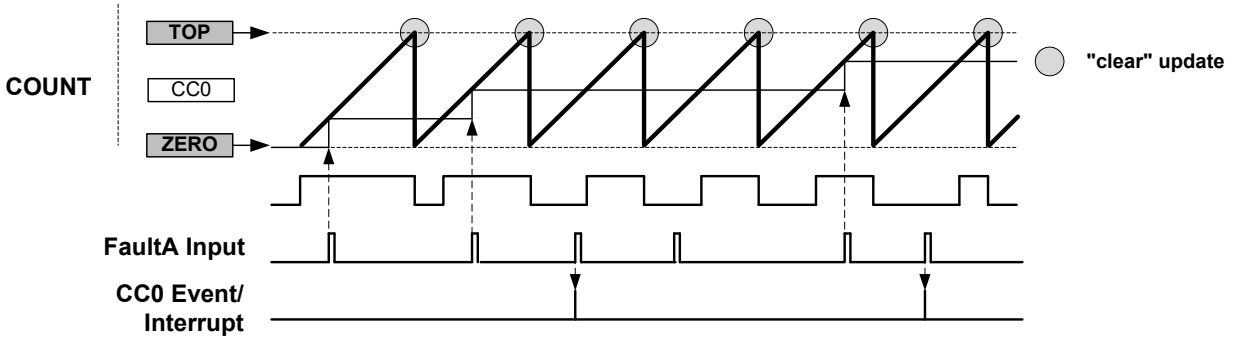
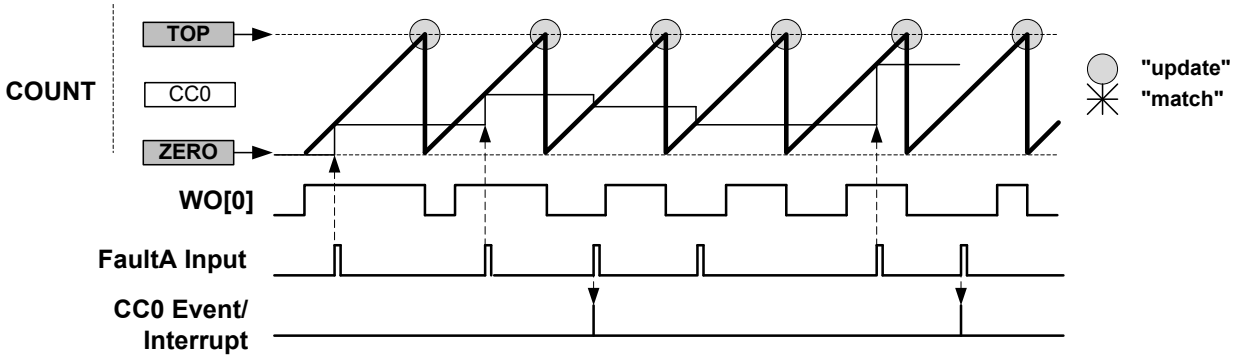


Figure 41-27. Capture Action “DERIV0”



Hardware Halt Action

This is configured by writing 0x1 to the Fault n Halt mode bits in the Recoverable Fault n Configuration register (FCTRLn.HALT). When enabled, the timer/counter is halted and the cycle is extended as long as the corresponding fault is present.

The next figure ('Waveform Generation with Halt and Restart Actions') shows an example where both restart action and hardware halt action are enabled for Fault A. The compare channel 0 output

is clamped to inactive level as long as the timer/counter is halted. The timer/counter resumes the counting operation as soon as the fault condition is no longer present. As the restart action is enabled in this example, the timer/counter is restarted after the fault condition is no longer present.

The figure after that ('Waveform Generation with Fault Qualification, Halt, and Restart Actions') shows a similar example, but with additionally enabled fault qualification. Here, counting is resumed after the fault condition is no longer present.

Note that in RAMP2 and RAMP2A operations, when a new timer/counter cycle starts, the cycle index will automatically change.

Figure 41-28. Waveform Generation with Halt and Restart Actions

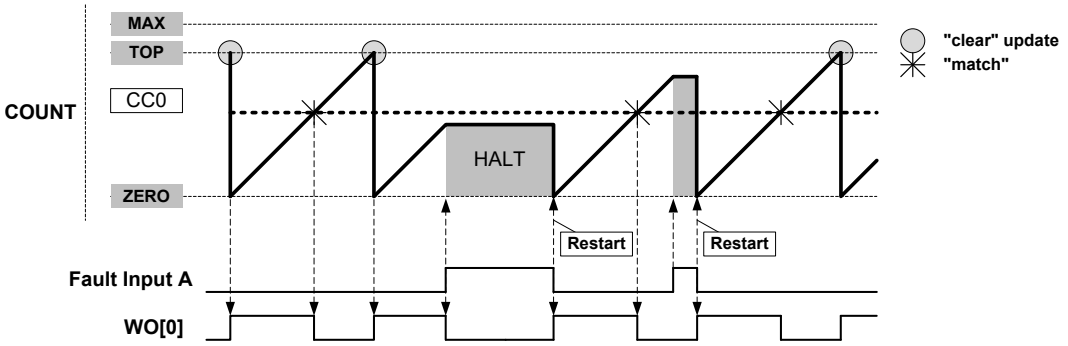
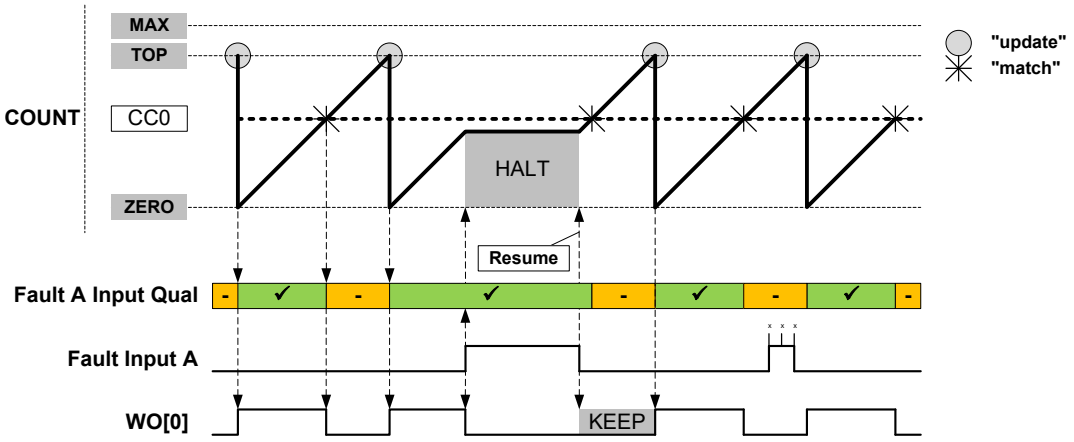


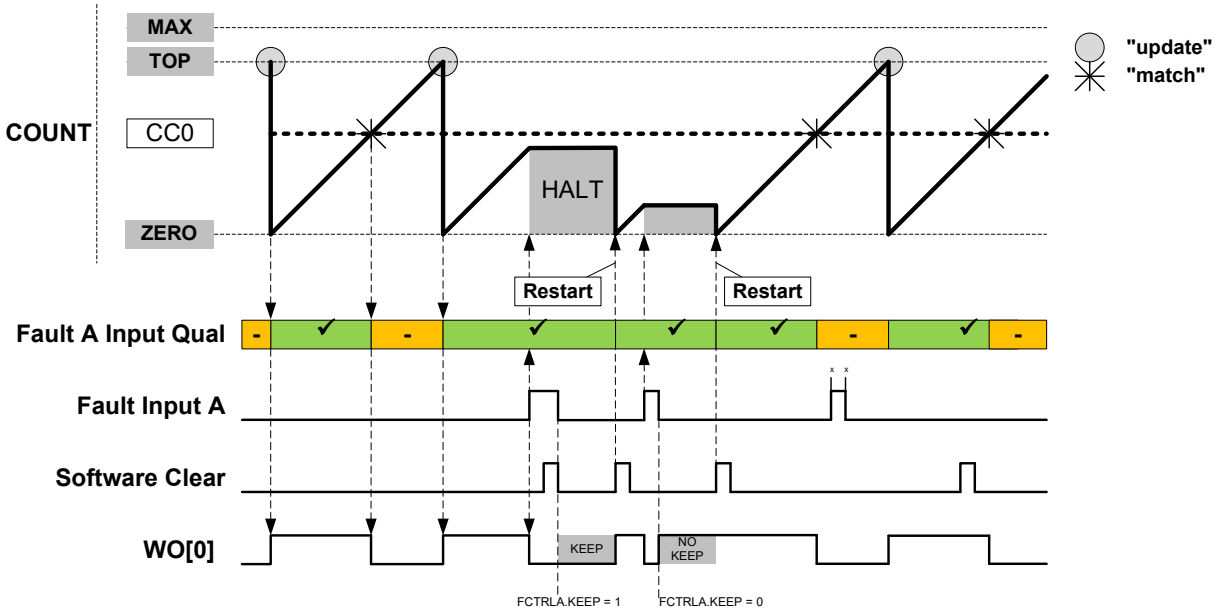
Figure 41-29. Waveform Generation with Fault Qualification, Halt, and Restart Actions



Software Halt Action

This is configured by writing 0x2 to the Fault n Halt mode bits in the Recoverable Fault n configuration register (FCTRLn.HALT). Software halt action is similar to hardware halt action, but in order to restart the timer/counter, the corresponding fault condition must not be present anymore, and the corresponding FAULT n bit in the STATUS register must be cleared by software.

Figure 41-30. Waveform Generation with Software Halt, Fault Qualification, Keep and Restart Actions



41.6.3.6 Non-Recoverable Faults

The non-recoverable fault action will force all the compare outputs to a pre-defined level programmed into the Driver Control register (DRVCTRL.NRE and DRVCTRL.NRV). The non-recoverable fault input (EV0 and EV1) actions are enabled in Event Control register (EVCTRL.EVACT0 and EVCTRL.EVACT1).

To avoid false fault detection on external events (e.g. a glitch on an I/O port) a digital filter can be enabled using Non-Recoverable Fault Input x Filter Value bits in the Driver Control register (DRVCTRL.FILTERVALn). Therefore, the event detection is synchronous, and event action is delayed by the selected digital filter value clock cycles.

When the Fault Detection on Debug Break Detection bit in Debug Control register (DGBCTRL.FDDBD) is written to '1', a non-recoverable Debug Faults State and an interrupt (DFS) is generated when the system goes in debug operation.

41.6.3.7 Waveform Extension

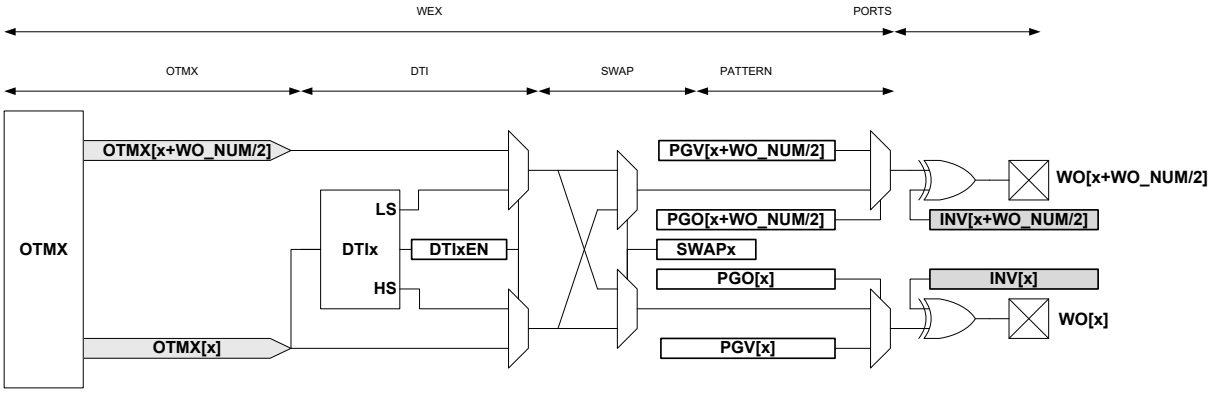
Waveform Extension Stage Details displays the schematic diagram of actions of the four optional units that follow the recoverable fault stage on a port pin pair: Output Matrix (OTMX), Dead-Time Insertion (DTI), SWAP and Pattern Generation. The DTI and SWAP units can be seen as a four port pair slices:

- Slice 0 DTI0 / SWAP0 acting on port pins (WO[0], WO[WO_NUM/2 +0])
- Slice 1 DTI1 / SWAP1 acting on port pins (WO[1], WO[WO_NUM/2 +1])

And generally:

- Slice n DTIx / SWAPx acting on port pins (WO[x], WO[WO_NUM/2 +x])

Figure 41-31. Waveform Extension Stage Details



The **output matrix (OTMX)** unit distributes compare channels, according to the selectable configurations in the following table.

Table 41-6. Output Matrix Channel Pin Routing Configuration

WEXCTRL.OTMX	OTMX[7]	OTMX[6]	OTMX[5]	OTMX[4]	OTMX[3]	OTMX[2]	OTMX[1]	OTMX[0]
0x0	CC1	CC0	CC5	CC4	CC3	CC2	CC1	CC0
0x1	CC1	CC0	CC2	CC1	CC0	CC2	CC1	CC0
0x2	CC0	CC0	CC0	CC0	CC0	CC0	CC0	CC0
0x3	CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC0

- Configuration 0x0 is the default configuration. The channel location is the default one and channels are distributed on outputs modulo the number of channels. Channel 0 is routed to the Output matrix output OTMX[0], and Channel 1 to OTMX[1]. If there are more outputs than channels, then channel 0 is duplicated to the Output matrix output OTMX[CC_NUM], channel 1 to OTMX[CC_NUM+1] and so on.
- Configuration 0x1 distributes the channels on output modulo half the number of channels. This assigns twice the number of output locations to the lower channels than the default configuration. This can be used, for example, to control the four transistors of a full bridge using only two compare channels.
  Using pattern generation, some of these four outputs can be overwritten by a constant level, enabling flexible drive of a full bridge in all quadrant configurations.
- Configuration 0x2 distributes compare channel 0 (CC0) to all port pins. With pattern generation, this configuration can control a stepper motor.
- Configuration 0x3 distributes the compare channel CC0 to the first output, and the channel CC1 to all other outputs. Together with pattern generation and the fault extension, this configuration can control up to seven LED strings, with a boost stage.

The table below is an example showing four compare channels on four outputs.

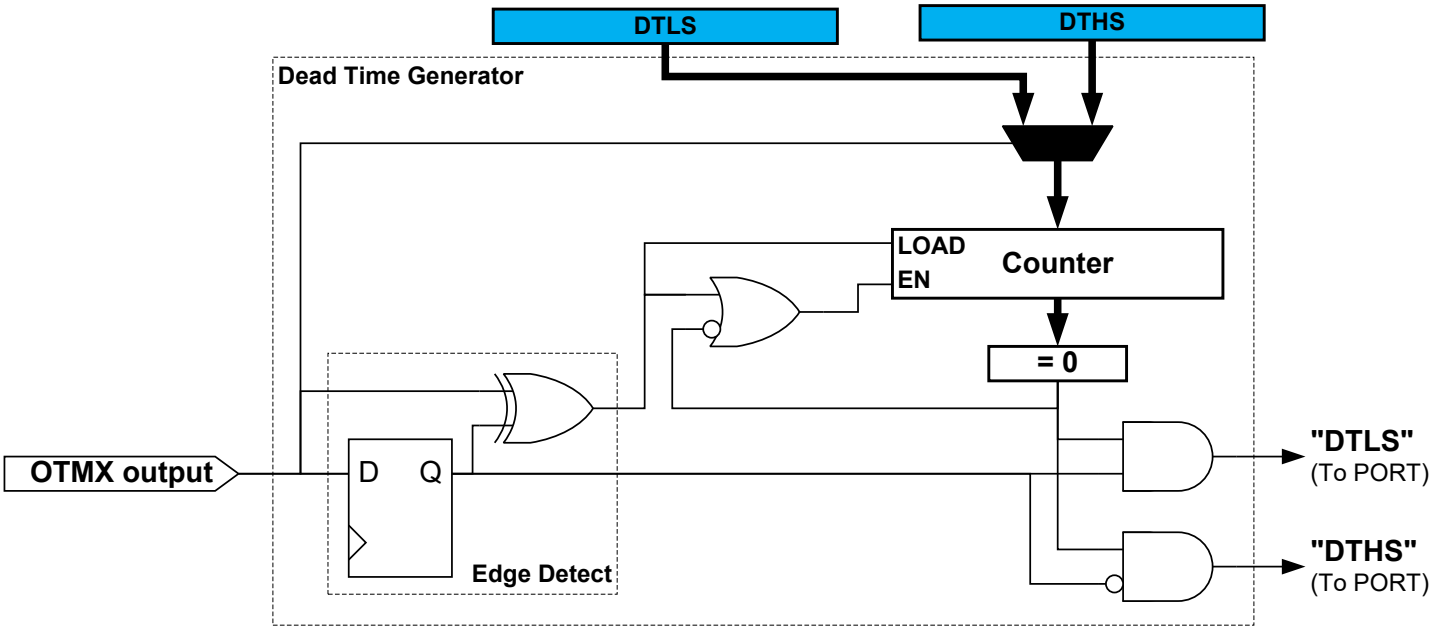
Table 41-7. Four Compare Channels on Four Outputs

WEXCTRL.OTMX	OTMX[3]	OTMX[2]	OTMX[1]	OTMX[0]
0x0	CC3	CC2	CC1	CC0
0x1	CC1	CC0	CC1	CC0
0x2	CC0	CC0	CC0	CC0
0x3	CC1	CC1	CC1	CC0

The dead-time insertion (DTI) unit generates OFF time with the non-inverted low side (LS) and inverted high side (HS) of the wave generator output forced at low level. This OFF time is called dead time. Dead-time insertion ensures that the LS and HS will never switch simultaneously.

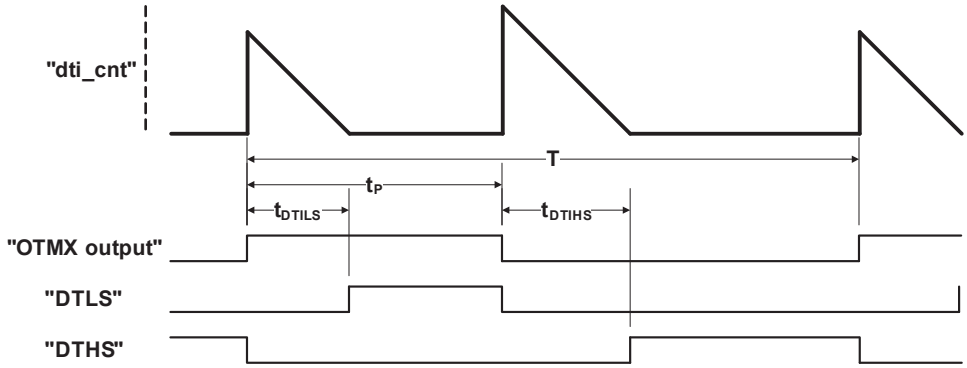
The DTI stage consists of four equal dead-time insertion generators; one for each of the first four compare channels. The following figure shows the block diagram of one DTI generator. The four channels have a common register which controls the dead time, which is independent of high side and low side setting.

Figure 41-32. Dead-Time Generator Block Diagram



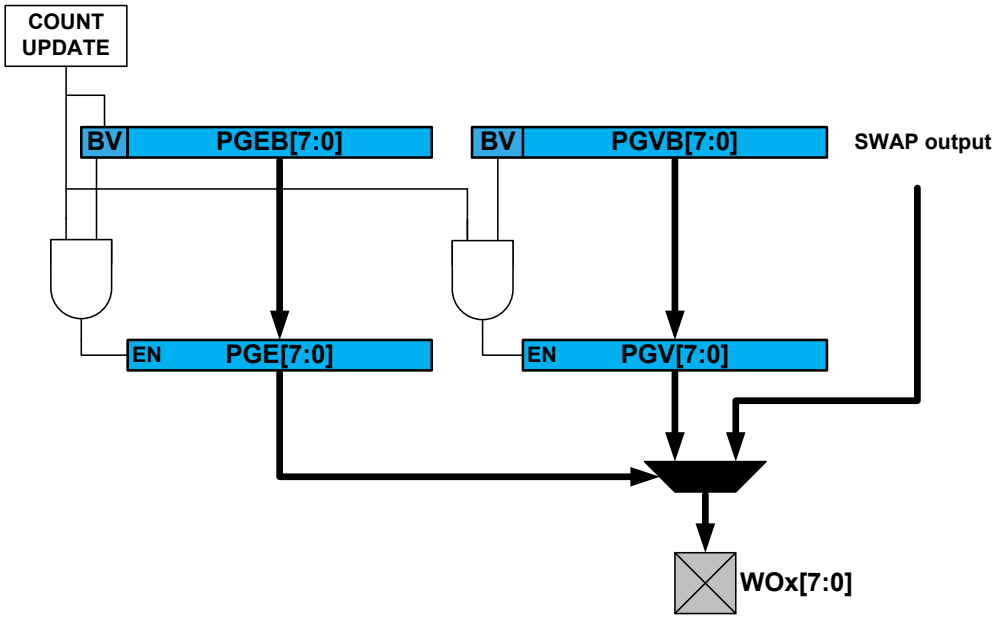
As shown in the following figure, the 8-bit dead-time counter is decremented by one for each peripheral clock cycle until it reaches zero. A non-zero counter value will force both the low side and high side outputs into their OFF state. When the output matrix (OTMX) output changes, the dead-time counter is reloaded according to the edge of the input. When the output changes from low to high (positive edge) it initiates a counter reload of the DTLS register. When the output changes from high to low (negative edge) it reloads the DTHS register.

Figure 41-33. Dead-Time Generator Timing Diagram



The pattern generator unit produces a synchronized bit pattern across the port pins it is connected to. The pattern generation features are primarily intended for handling the commutation sequence in brushless DC motors (BLDC), stepper motors, and full bridge control. For more information, refer to the following figure.

Figure 41-34. Pattern Generator Block Diagram



As with other double-buffered timer/counter registers, the register update is synchronized to the UPDATE condition set by the timer/counter waveform generation operation. If synchronization is not required by the application, the software can simply access directly the PATT.PGE, PATT.PGV bits registers.

41.6.4 DMA, Interrupts, and Events

Table 41-8. Module Requests for TCC

Condition	Interrupt request	Event output	Event input	DMA request	DMA request is cleared
Overflow / Underflow	Yes	Yes		Yes(1)	On DMA acknowledge
Channel Compare Match or Capture	Yes	Yes	Yes(2)	Yes(3)	For circular buffering: on DMA acknowledge For capture channel: when CCx register is read
Retrigger	Yes	Yes			
Count	Yes	Yes			
Capture Overflow Error	Yes				
Debug Fault State	Yes				
Recoverable Faults	Yes				
Non-Recoverable Faults	Yes				
TCCx Event 0 input			Yes(4)		
TCCx Event 1 input			Yes(5)		

Notes:

1. DMA request set on Overflow, Underflow or Re-trigger conditions.
2. Can perform capture or generate recoverable fault on an event input.
3. In Capture or Circular modes.
4. On event input, either action can be executed:

- re-trigger counter
- control counter direction
- stop the counter
- decrement the counter
- perform period and pulse width capture
- generate non-recoverable fault

5. On event input, either action can be executed:
   - re-trigger counter
   - increment or decrement counter depending on direction
   - start the counter
   - increment or decrement counter based on direction
   - increment counter regardless of direction
   - generate non-recoverable fault

41.6.4.1 DMA Operation

The TCC can generate the following DMA requests:

Counter overflow (OVF)	The TCC generates a DMA request on each cycle when an update condition (Overflow, Underflow or Re-trigger) is detected. In both cases, the request is cleared by hardware on DMA acknowledge.
Channel Match (MCx)	A DMA request is set only on a compare match . The request is cleared by hardware on DMA acknowledge.
Channel Capture (MCx)	For a capture channel, the request is set when valid data is present in the CCx register, and cleared once the CCx register is read.

DMA Operation with Circular Buffer

When circular buffer operation is enabled, the Buffer registers must be written in a correct order and synchronized to the update times of the timer. The DMA triggers of the TCC provide a way to ensure a safe and correct update of circular buffers.

Note: Circular buffer are intended to be used with RAMP2, RAMP2A and DSBOTH operation only.

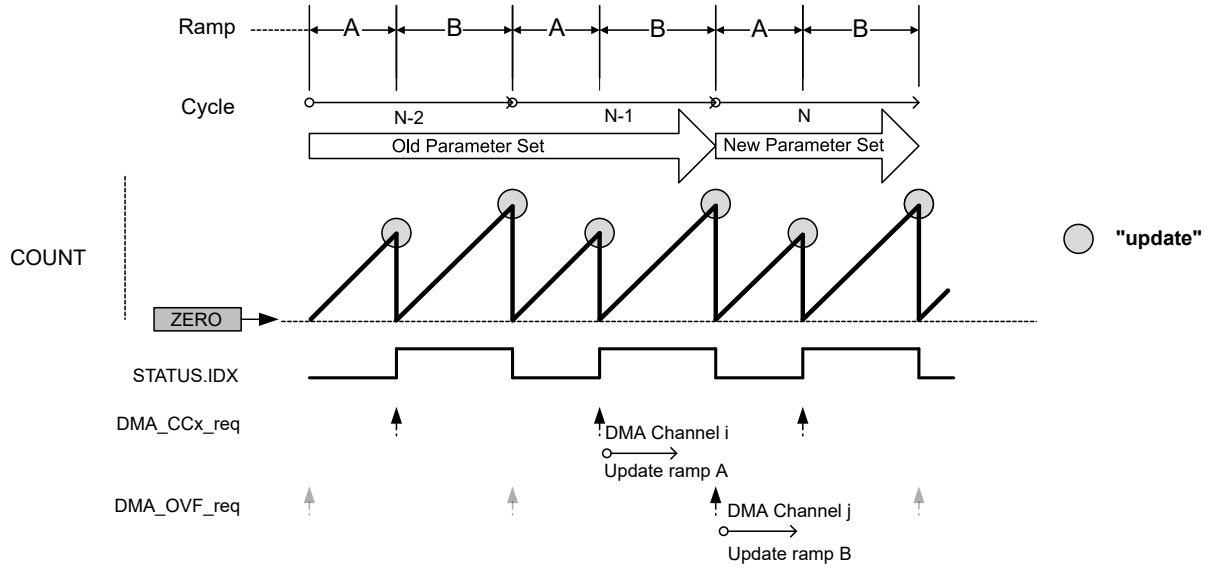
DMA Operation with Circular Buffer in RAMP2 and RAMP2A Mode

When a CCx channel is selected as a circular buffer, the related DMA request is not set on a compare match detection, but on start of ramp B.

If at least one circular buffer is enabled, the DMA overflow request is conditioned to the start of ramp A with an effective DMA transfer on previous ramp B (DMA acknowledge).

The update of all circular buffer values for ramp A can be done through a DMA channel triggered on a MC trigger. The update of all circular buffer values for ramp B, can be done through a second DMA channel triggered by the overflow DMA request.

Figure 41-35. DMA Triggers in RAMP and RAMP2 Operation Mode and Circular Buffer Enabled

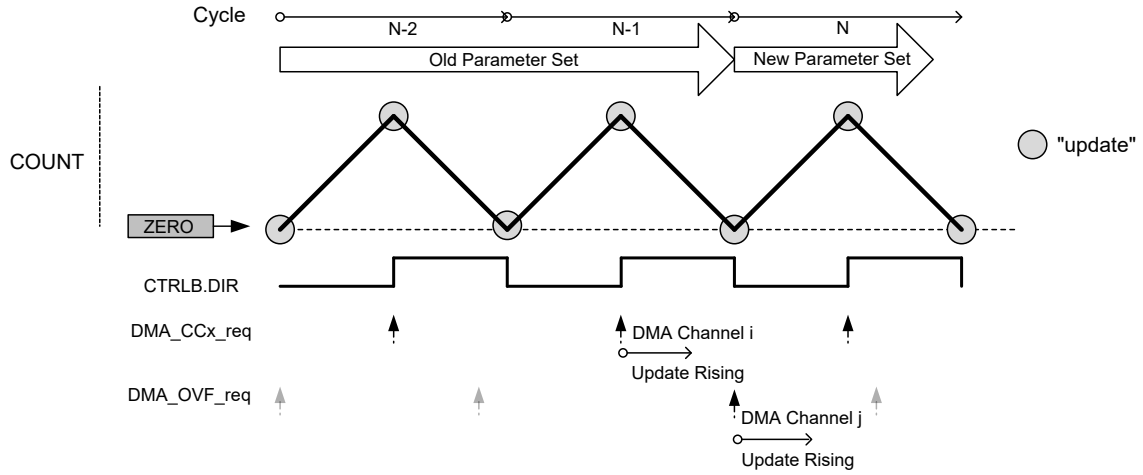


DMA Operation with Circular Buffer in DSBOTH Mode
When a CC channel is selected as a circular buffer, the related DMA request is not set on a compare match detection, but on start of down-counting phase.

If at least one circular buffer is enabled, the DMA overflow request is conditioned to the start of up-counting phase with an effective DMA transfer on previous down-counting phase (DMA acknowledge).

When up-counting, all circular buffer values can be updated through a DMA channel triggered by MC trigger. When down-counting, all circular buffer values can be updated through a second DMA channel, triggered by the OVF DMA request.

Figure 41-36. DMA Triggers in DSBOTH Operation Mode and Circular Buffer Enabled



41.6.4.2 Interrupts

The TCC has the following interrupt sources:

- Overflow/Underflow (OVF)
- Retrigger (TRG)
- Count (CNT) – Refer also to the description of EVCTRL.CNTSEL
- Capture Overflow Error (ERR)
- Debug Fault State (DFS)

- Recoverable Faults (FAULTn)
- Non-recoverable Faults (FAULTx)
- Compare Match or Capture Channels (MCx)

These interrupts are asynchronous wake-up sources.

Each interrupt source has an Interrupt flag associated with it. The Interrupt flag in the Interrupt Flag Status and Clear (INTFLAG) register is set when the Interrupt condition occurs. Each interrupt can be individually enabled by writing a '1' to the corresponding bit in the Interrupt Enable Set (INTENSET) register, and disabled by writing a '1' to the corresponding bit in the Interrupt Enable Clear (INTENCLR) register. The status of enabled interrupts can be read from either INTENSET or INTENCLR.

An interrupt request is generated when the Interrupt flag is set and the corresponding interrupt is enabled. The interrupt request remains active until the Interrupt flag is cleared, the interrupt is disabled or the TCC is reset. See *INTFLAG* from Related Links for details on how to clear Interrupt flags. The TCC has one common interrupt request line for all the interrupt sources. The user must read the INTFLAG register to determine which Interrupt condition is present.

Interrupts must be globally enabled for interrupt requests to be generated. See *Nested Vector Interrupt Controller (NVIC)* from Related Links.

Related Links

10.2. Nested Vector Interrupt Controller (NVIC)
41.8.12. INTFLAG

41.6.4.3 Events

The TCC can generate the following output events:

- Overflow/Underflow (OVF)
- Trigger (TRG)
- Counter (CNT) (For further details, refer to the EVCTRL.CNTSEL description.)
- Compare Match or Capture on compare/capture channels: MCx

Writing a '1' ('0') to an Event Output bit in the Event Control Register (EVCTRL.xxEO) enables (disables) the corresponding output event. See *Event System (EVSYS)* from Related Links.

The TCC can take the following actions on a channel input event (MCx):

- Capture event
- Generate a recoverable or non-recoverable fault

The TCC can take the following actions on counter Event 1 (TCCx EV1):

- Counter re-trigger
- Counter direction control
- Stop the counter
- Decrement the counter on event
- Period and pulse width capture
- Non-recoverable fault

The TCC can take the following actions on counter Event 0 (TCCx EV0):

- Counter re-trigger
- Count on event (increment or decrement, depending on counter direction)
- Counter start – Start counting on the event rising edge. Further events will not restart the counter; the counter will keep counting using prescaled GCLK_TCCx, until it reaches TOP or ZERO, depending on the direction.
- Counter increment on event. This will increment the counter, irrespective of the counter direction.
- Count during active state of an asynchronous event (increment or decrement, depending on counter direction). In this case, the counter will be incremented or decremented on each cycle of the prescaled clock, as long as the event is active.
- Non-recoverable fault

The counter Event Actions are available in the Event Control registers (EVCTRL.EVACT0 and EVCTRL.EVACT1). See *EVCTRL* from Related Links.

Writing a '1' ('0') to an Event Input bit in the Event Control register (EVCTRL.MCEIx or EVCTRL.TCEIx) enables (disables) the corresponding action on input event.

Note: When several events are connected to the TCC, the enabled action will apply for each of the incoming events. See *Event System (EVSYS)* from Related Links for details on how to configure the Event System.

Related Links

28. Event System (EVSYS)

41.8.9. EVCTRL

41.6.5 Sleep Mode Operation

The TCC can be configured to operate in any Sleep mode. To be able to run in standby the RUNSTDBY bit in the Control A register (CTRLA.RUNSTDBY) must be '1'. The MODULE can in any Sleep mode wake-up the device using interrupts or perform actions through the Event System.

41.6.6 Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domains, some registers need to be synchronized when written or read.

The following bits are synchronized when written:

- Software Reset and Enable bits in Control A register (CTRLA.SWRST and CTRLA.ENABLE)

The following registers are synchronized when written:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Status register (STATUS)
- Pattern and Pattern Buffer registers (PATT and PATTB)
- Waveform register (WAVE)
- Count Value register (COUNT)
- Period Value and Period Buffer Value registers (PER and PERB)
- Compare/Capture Channel x and Channel x Compare/Capture Buffer Value registers (CCx and CCBx)

The following registers are synchronized when read:

- Control B Clear and Control B Set registers (CTRLBCLR and CTRLBSET)
- Count Value register (COUNT): synchronization is done on demand through READSYNC command (CTRLBSET.CMD)
- Pattern and Pattern Buffer registers (PATT and PATTB)
- Waveform register (WAVE)
- Period Value and Period Buffer Value registers (PER and PERB)
- Compare/Capture Channel x and Channel x Compare/Capture Buffer Value registers (CCx and CCBx)

Required write synchronization is denoted by the "Write-Synchronized" property in the register description.

Required read synchronization is denoted by the "Read-Synchronized" property in the register description.

41.7 Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0		RESOLUTION[1:0]					ENABLE	SWRST
		15:8	MSYNC	ALOCK	PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
		23:16	DMAOS							
		31:24			CPTEN5	CPTEN4	CPTEN3	CPTEN2	CPTEN1	CPTEN0
0x04	CTRLBCLR	7:0	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
0x05	CTRLBSET	7:0	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
0x06 ... 0x07	Reserved									
0x08	SYNCBUSY	7:0	PER	WAVE	PATT	COUNT	STATUS	CTRLB	ENABLE	SWRST
		15:8								
		23:16								
		31:24								
0x0C	FCTRLA	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]	
		15:8	BLANKPRESC	CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]	
		23:16	BLANKVAL[7:0]							
		31:24					FILTERVAL[3:0]			
0x10	FCTRLB	7:0	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]	
		15:8	BLANKPRESC	CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]	
		23:16	BLANKVAL[7:0]							
		31:24					FILTERVAL[3:0]			
0x14	WEXCTRL	7:0							OTMX[1:0]	
		15:8					DTIEN3	DTIEN2	DTIEN1	DTIEN0
		23:16	DTLS[7:0]							
		31:24	DTHS[7:0]							
0x18	DRVCTRL	7:0	NRE7	NRE6	NRE5	NRE4	NRE3	NRE2	NRE1	NRE0
		15:8	NRV7	NRV6	NRV5	NRV4	NRV3	NRV2	NRV1	NRV0
		23:16	INVEN7	INVEN6	INVEN5	INVEN4	INVEN3	INVEN2	INVEN1	INVEN0
		31:24	FILTERVAL1[3:0]				FILTERVAL0[3:0]			
0x1C ... 0x1D	Reserved									
0x1E	DBGCTRL	7:0						FDDBD		DBGRUN
0x1F	Reserved									
0x20	EVCTRL	7:0	CNTSEL[1:0]		EVACT1[2:0]			EVACT0[2:0]		
		15:8	TCEI1	TCEI0	TCINV1	TCINV0		CNTEO	TRGEO	OVFEO
		23:16								
		31:24								
0x24	INTENCLR	7:0					ERR	CNT	TRG	OVF
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
		23:16								
		31:24								
0x28	INTENSET	7:0					ERR	CNT	TRG	OVF
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
		23:16								
		31:24								
0x2C	INTFLAG	7:0					ERR	CNT	TRG	OVF
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
		23:16								
		31:24								
0x30	STATUS	7:0	PERBV		PATTBV		DFS		IDX	STOP
		15:8	FAULT1	FAULT0	FAULTB	FAULTA	FAULT1IN	FAULT0IN	FAULTBIN	FAULTAIN
		23:16								
		31:24								

.....continued

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x34	COUNT	7:0								
		15:8								
		23:16								
		31:24								
0x38	PATT	7:0	PGE7	PGE6	PGE5	PGE4	PGE3	PGE2	PGE1	PGE0
		15:8	PGV7	PGV6	PGV5	PGV4	PGV3	PGV2	PGV1	PGV0
0x3A ... 0x3B	Reserved									
0x3C	WAVE	7:0	CIPEREN					WAVEGEN[2:0]		
		15:8					CICCEN3	CICCEN2	CICCEN1	CICCEN0
		23:16								
		31:24					SWAP3	SWAP2	SWAP1	SWAP0
0x40	PER	7:0			DITHER[5:0]					
		15:8								
		23:16								
		31:24								
0x44 ... 0x63	Reserved									
0x64	PATTB	7:0	PGEB7	PGEB6	PGEB5	PGEB4	PGEB3	PGEB2	PGEB1	PGEB0
		15:8	PGVB7	PGVB6	PGVB5	PGVB4	PGVB3	PGVB2	PGVB1	PGVB0
0x66 ... 0x6B	Reserved									
0x6C	PERB	7:0			DITHERB[5:0]					
		15:8								
		23:16								
		31:24								

41.8 Register Description

Registers can be 8, 16 or 32 bits wide. Atomic 8-, 16- and 32-bit accesses are supported. In addition, the 8-bit quarters and 16-bit halves of a 32-bit register, and the 8-bit halves of a 16-bit register can be accessed directly.

Some registers require synchronization when read and/or written. Synchronization is denoted by the “Read-Synchronized” and/or “Write-Synchronized” property in each individual register description.

Optional write protection by the Peripheral Access Controller (PAC) is denoted by the "PAC Write Protection" property in each individual register description.

Some registers are enable-protected, meaning they can only be written when the module is disabled. Enable-protection is denoted by the “Enable-Protected” property in each individual register description.

41.8.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected, Write-Synchronized (ENABLE, SWRST)

Bit	31	30	29	28	27	26	25	24
			CPTEN5	CPTEN4	CPTEN3	CPTEN2	CPTEN1	CPTEN0
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMAOS							
Access	R/W							
Reset	0							

Bit	15	14	13	12	11	10	9	8
	MSYNC	ALOCK	PRESCSYNC[1:0]		RUNSTDBY	PRESCALER[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RESOLUTION[1:0]					ENABLE	SWRST
Access		R/W	R/W				R/W	R/W
Reset		0	0				0	0

Bits 24, 25, 26, 27, 28, 29 – CPTENx Capture Channel x Enable
These bits are used to select the capture or compare operation on channel x.
Writing a '1' to CPTENx enables capture on channel x.
Writing a '0' to CPTENx disables capture on channel x.

Bit 23 – DMAOS DMA One-Shot Trigger Mode
This bit enables the DMA One-shot Trigger Mode.
Writing a '1' to this bit will generate a DMA trigger on TCC cycle following a TCC_CTRLBSET_CMD_DMAOS command.
Writing a '0' to this bit will generate DMA triggers on each TCC cycle.
This bit is not synchronized.

Bit 15 – MSYNC Host Synchronization (only for TCC Client instance)
This bit must be set if the TCC counting operation must be synchronized on its Host TCC.
This bit is not synchronized.

Value	Description
0	The TCC controls its own counter.
1	The counter is controlled by its Host TCC.

Bit 14 – ALOCK Auto Lock
This bit is not synchronized.

Value	Description
0	The Lock Update bit in the Control B register (CTRLB.LUPD) is not affected by overflow/underflow, and re-trigger events
1	CTRLB.LUPD is set to '1' on each overflow/underflow or re-trigger event.

Bits 13:12 – PRESCSYNC[1:0] Prescaler and Counter Synchronization
These bits select if on re-trigger event, the Counter is cleared or reloaded on either the next GCLK_TCCx clock, or on the next prescaled GCLK_TCCx clock. It is also possible to reset the prescaler on re-trigger event.

These bits are not synchronized.

Value	Name	Description	
		Counter Reloaded	Prescaler
0x0	GCLK	Reload or reset Counter on next GCLK	-
0x1	PRESC	Reload or reset Counter on next prescaler clock	-
0x2	RESYNC	Reload or reset Counter on next GCLK	Reset prescaler counter
0x3	Reserved		

Bit 11 – RUNSTDBY Run in Standby

This bit is used to keep the TCC running in Standby mode.
This bit is not synchronized.

Value	Description
0	The TCC is halted in standby.
1	The TCC continues to run in standby.

Bits 10:8 – PRESCALER[2:0] Prescaler

These bits select the Counter prescaler factor.
These bits are not synchronized.

Value	Name	Description
0x0	DIV1	Prescaler: GCLK_TCCx
0x1	DIV2	Prescaler: GCLK_TCCx/2
0x2	DIV4	Prescaler: GCLK_TCCx/4
0x3	DIV8	Prescaler: GCLK_TCCx/8
0x4	DIV16	Prescaler: GCLK_TCCx/16
0x5	DIV64	Prescaler: GCLK_TCCx/64
0x6	DIV256	Prescaler: GCLK_TCCx/256
0x7	DIV1024	Prescaler: GCLK_TCCx/1024

Bits 6:5 – RESOLUTION[1:0] Dithering Resolution

These bits increase the TCC resolution by enabling the dithering options.
These bits are not synchronized.

Table 41-9. Dithering

Value	Name	Description
0x0	NONE	The dithering is disabled.
0x1	DITH4	Dithering is done every 16 PWM frames. PER[3:0] and CCx[3:0] contain dithering pattern selection.
0x2	DITH5	Dithering is done every 32 PWM frames. PER[4:0] and CCx[4:0] contain dithering pattern selection.
0x3	DITH6	Dithering is done every 64 PWM frames. PER[5:0] and CCx[5:0] contain dithering pattern selection.

Bit 1 – ENABLE Enable

Due to synchronization there is delay from writing CTRLA.ENABLE until the peripheral is enabled/disabled. The value written to CTRLA.ENABLE will read back immediately and the ENABLE bit in the SYNCBUSY register (SYNCBUSY.ENABLE) will be set. SYNCBUSY.ENABLE will be cleared when the operation is complete.

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

Bit 0 – SWRST Software Reset

Writing a '0' to this bit has no effect.
Writing a '1' to this bit resets all registers in the TCC (except DBGCTRL) to their initial state, and the TCC will be disabled.

Writing a '1' to CTRLA.SWRST will always take precedence; all other writes in the same write-operation will be discarded.

Notes:

1. When the CTRLA.SWRST is written, the user must poll the SYNCBUSY.SWRST bit to know when the reset operation is complete.
2. During a SWRST, access to registers or bits without SWRST are disallowed until the SYNCBUSY.SWRST bit is cleared by hardware.

Value	Description
0	There is no Reset operation ongoing.
1	The Reset operation is ongoing.

41.8.2 Control B Clear

Name: CTRLBCLR
Offset: 0x04
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized, Read-Synchronized

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBSET) register.

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:5 – CMD[2:0] TCC Command
Writing a ‘0’ to these bits has no effect.
Writing a '1' to any of these bits will clear the pending command.

Bits 4:3 – IDXCMD[1:0] Ramp Index Command
These bits can be used to force cycle A and cycle B changes in RAMP2 and RAMP2A operation. On timer/counter update condition, the command is executed, the IDX flag in STATUS register is updated and the IDXCMD command is cleared.
Writing zero to these bits has no effect.
Writing a '1' to any of these bits will clear the pending command.

Value	Name	Description
0x0	DISABLE	DISABLE Command disabled: IDX toggles between cycles A and B
0x1	SET	Set IDX: cycle B will be forced in the next cycle
0x2	CLEAR	Clear IDX: cycle A will be forced in next cycle
0x3	HOLD	Hold IDX: the next cycle will be the same as the current cycle.

Bit 2 – ONESHOT One-Shot
This bit controls one-shot operation of the TCC. When one-shot operation is enabled, the TCC will stop counting on the next overflow/underflow condition or on a stop command.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will disable the one-shot operation.

Value	Description
0	The TCC will update the counter value on overflow/underflow condition and continue operation.
1	The TCC will stop counting on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update
This bit controls the update operation of the TCC buffered registers.
When CTRLB.LUPD is cleared, the hardware UPDATE registers with value from their buffered registers is enabled.
This bit has no effect when input capture operation is enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will enable the registers updates on hardware UPDATE condition.

Value	Description
0	The CCBx, PERB, PGVB, and PGEB buffer registers values *are* copied into the corresponding CCx, PER, PGV, and PGE registers on hardware update condition.
1	The CCBx, PERB, PGVB, and PGEB buffer registers values are *not* copied into the corresponding CCx, PER, PGV, and PGE registers on hardware update condition.

Bit 0 – DIR Counter Direction
This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will clear the bit and make the counter count up.

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

41.8.3 Control B Set

Name: CTRLBSET
Offset: 0x05
Reset: 0x00
Property: PAC Write-Protection, Write-Synchronized, Read-Synchronized

This register allows the user to change this register without doing a read-modify-write operation. Changes in this register will also be reflected in the Control B Set (CTRLBCLR) register.

Bit	7	6	5	4	3	2	1	0
	CMD[2:0]			IDXCMD[1:0]		ONESHOT	LUPD	DIR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:5 – CMD[2:0] TCC Command
These bits can be used for software control of re-triggering and stop commands of the TCC. When a command has been executed, the CMD bit field will be read back as zero. The commands are executed on the next prescaled GCLK_TCCx clock cycle.
Writing zero to this bit group has no effect
Writing a value different from 0x0 to this bit field will issue a command for execution.

Important: This command requires synchronization before being executed.
A valid sequence is the following:

- Issue CMD command (CTRLBSET.CMD = command)
- Wait for CMD synchronization (SYNCBUSY.CTRLB = 0)
- Wait for CMD read back as zero (CTRLBSET.CMD = 0)

Value	Name	Description
0x0	NONE	No action
0x1	RETRIGGER	Force start, restart or retrigger
0x2	STOP	Force stop
0x3	UPDATE	Force update of double buffered registers
0x4	READSYNC	Force a read synchronization of COUNT
0x5	DMAOS	One-shot DMA trigger

Bits 4:3 – IDXCMD[1:0] Ramp Index Command
These bits can be used to force cycle A and cycle B changes in RAMP2 and RAMP2A operation. On timer/counter update condition, the command is executed, the IDX flag in STATUS register is updated and the IDXCMD command is cleared.
Writing a zero to these bits has no effect.
Writing a valid value to these bits will set a command.

Value	Name	Description
0x0	DISABLE	Command disabled: IDX toggles between cycles A and B
0x1	SET	Set IDX: cycle B will be forced in the next cycle
0x2	CLEAR	Clear IDX: cycle A will be forced in next cycle
0x3	HOLD	Hold IDX: the next cycle will be the same as the current cycle.

Bit 2 – ONESHOT One-Shot
This bit controls one-shot operation of the TCC. When in one-shot operation, the TCC will stop counting on the next overflow/underflow condition or a stop command.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will enable the one-shot operation.

Value	Description
0	The TCC will count continuously.

Value	Description
1	The TCC will stop counting on the next underflow/overflow condition.

Bit 1 – LUPD Lock Update

This bit controls the update operation of the TCC buffered registers.
When CTRLB.LUPD is set, the hardware UPDATE registers with value from their buffered registers is disabled. Disabling the update ensures that all buffer registers are valid before an hardware update is performed. After all the buffer registers are loaded correctly, the buffered registers can be unlocked.
This bit has no effect when input capture operation is enabled.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will disable the registers updates on hardware UPDATE condition.

Value	Description
0	The CCBx, PERB, PGVB, and PGEB buffer registers values *are* copied into the corresponding CCx, PER, PGV, and PGE registers on hardware update condition.
1	The CCBx, PERB, PGVB, and PGEB buffer registers values are *not* copied into CCx, PER, PGV, and PGE registers on hardware update condition.

Bit 0 – DIR Counter Direction

This bit is used to change the direction of the counter.
Writing a '0' to this bit has no effect
Writing a '1' to this bit will set the bit and make the counter count down.
Note: When the TCC is counting down, the COUNT register must be initialized to the TOP value (PER or CC0 value depending on the mode).

Value	Description
0	The timer/counter is counting up (incrementing).
1	The timer/counter is counting down (decrementing).

41.8.4 Synchronization Busy

Name: SYNCBUSY
Offset: 0x08
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PER	WAVE	PATT	COUNT	STATUS	CTRLB	ENABLE	SWRST
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 7 – PER PER Synchronization Busy
This bit is cleared when the synchronization of the PER register between the clock domains is complete.
This bit is set when the synchronization of the PER register between clock domains is started.

Bit 6 – WAVE WAVE Synchronization Busy
This bit is cleared when the synchronization of the WAVE register between the clock domains is complete.
This bit is set when the synchronization of the WAVE register between clock domains is started.

Bit 5 – PATT PATT Synchronization Busy
This bit is cleared when the synchronization of the PATTERN register between the clock domains is complete.
This bit is set when the synchronization of the PATTERN register between clock domains is started.

Bit 4 – COUNT COUNT Synchronization Busy
This bit is cleared when the synchronization of the COUNT register between the clock domains is complete.
This bit is set when the synchronization of the COUNT register between clock domains is started.

Bit 3 – STATUS STATUS Synchronization Busy
This bit is cleared when the synchronization of the STATUS register between the clock domains is complete.
This bit is set when the synchronization of the STATUS register between clock domains is started.

Bit 2 – CTRLB CTRLB Synchronization Busy
This bit is cleared when the synchronization of the CTRLB register between the clock domains is complete.
This bit is set when the synchronization of the CTRLB register between clock domains is started.

Bit 1 – ENABLE ENABLE Synchronization Busy
This bit is cleared when the synchronization of the ENABLE bit between the clock domains is complete.
This bit is set when the synchronization of the ENABLE bit between clock domains is started.

Bit 0 – SWRST SWRST Synchronization Busy

This bit is cleared when the synchronization of the SWRST bit between the clock domains is complete.
This bit is set when the synchronization of the SWRST bit between clock domains is started.
Note: During a SWRST, access to registers/bits without SWRST are disallowed until SYNCBUSY.SWRST is cleared by hardware.

41.8.5 Fault Control A and B

Name: FCTRLn
Offset: 0x0C + n*0x04 [n=0..1]
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
					FILTERVAL[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BLANKVAL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BLANKPRESC	CAPTURE[2:0]			CHSEL[1:0]		HALT[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RESTART	BLANK[1:0]		QUAL	KEEP		SRC[1:0]	
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bits 27:24 – FILTERVAL[3:0] Recoverable Fault n Filter Value
These bits define the filter value applied on MCEx (x=0,1) event input line. The value must be set to zero when MCEx event is used as synchronous event.

Bits 23:16 – BLANKVAL[7:0] Recoverable Fault n Blanking Value
These bits determine the duration of the blanking of the fault input source. Activation and edge selection of the blank filtering are done by the BLANK bits (FCTRLn.BLANK).
When enabled, the fault input source is internally disabled for BLANKVAL* prescaled GCLK_TCCx periods after the detection of the waveform edge.

Bit 15 – BLANKPRESC Recoverable Fault n Blanking Value Prescaler
This bit enables a factor 64 prescaler factor on used as base frequency of the BLANKVAL value.

Value	Description
0	Blank time is BLANKVAL* prescaled GCLK_TCCx.
1	Blank time is BLANKVAL* 64 * prescaled GCLK_TCCx.

Bits 14:12 – CAPTURE[2:0] Recoverable Fault n Capture Action
These bits select the capture and Fault n interrupt/event conditions.

Table 41-10. Fault n Capture Action

Value	Name	Description
0x0	DISABLE	Capture on valid recoverable Fault n is disabled
0x1	CAPT	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each new captured value.
0x2	CAPTMIN	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0], if COUNT value is lower than the last stored capture value (CC). INTFLAG.FAULTn flag rises on each local minimum detection.

.....continued		
Value	**Name**	**Description**
0x3	CAPTMAX	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0], if COUNT value is higher than the last stored capture value (CC). INTFLAG.FAULTn flag rises on each local maximun detection.
0x4	LOCMIN	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each local minimum value detection.
0x5	LOCMAX	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each local maximun detection.
0x6	DERIV0	On rising edge of a valid recoverable Fault n, capture counter value on channel selected by CHSEL[1:0]. INTFLAG.FAULTn flag rises on each local maximun or minimum detection.
0x7	CAPTMARK	Capture with ramp index as MSB value.

Bits 11:10 – CHSEL[1:0] Recoverable Fault n Capture Channel

These bits select the channel for capture operation triggered by recoverable Fault n.

Value	Name	Description
0x0	CC0	Capture value stored into CC0
0x1	CC1	Capture value stored into CC1
0x2	CC2	Capture value stored into CC2
0x3	CC3	Capture value stored into CC3

Bits 9:8 – HALT[1:0] Recoverable Fault n Halt Operation

These bits select the halt action for recoverable Fault n.

Value	Name	Description
0x0	DISABLE	Halt action disabled
0x1	HW	Hardware halt action
0x2	SW	Software halt action
0x3	NR	Non-recoverable fault

Bit 7 – RESTART Recoverable Fault n Restart

Setting this bit enables restart action for Fault n.

Value	Description
0	Fault n restart action is disabled.
1	Fault n restart action is enabled.

Bits 6:5 – BLANK[1:0] Recoverable Fault n Blanking Operation

These bits, select the blanking start point for recoverable Fault n.

Value	Name	Description
0x0	START	Blanking applied from start of the Ramp period
0x1	RISE	Blanking applied from rising edge of the waveform output
0x2	FALL	Blanking applied from falling edge of the waveform output
0x3	BOTH	Blanking applied from each toggle of the waveform output

Bit 4 – QUAL Recoverable Fault n Qualification

Setting this bit enables the recoverable Fault n input qualification.

Value	Description
0	The recoverable Fault n input is not disabled on CMPx value condition.
1	The recoverable Fault n input is disabled when output signal is at inactive level (CMPx == 0).

Bit 3 – KEEP Recoverable Fault n Keep

Setting this bit enables the Fault n keep action.

Value	Description
0	The Fault n state is released as soon as the recoverable Fault n is released.
1	The Fault n state is released at the end of TCC cycle.

Bits 1:0 – SRC[1:0] Recoverable Fault n Source

These bits select the TCC event input for recoverable Fault n.

Event system channel connected to MCEx event input, must be configured to route the event asynchronously, when used as a recoverable Fault n input.

Value	Name	Description
0x0	DISABLE	Fault input disabled
0x1	ENABLE	MCEx (x=0,1) event input
0x2	INVERT	Inverted MCEx (x=0,1) event input
0x3	ALTFAULT	Alternate fault (A or B) state at the end of the previous period.

41.8.6 Waveform Extension Control

Name: WEXCTRL
Offset: 0x14
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	DTHS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DTLS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
					DTIEN3	DTIEN2	DTIEN1	DTIEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
							OTMX[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 31:24 – DTHS[7:0] Dead-Time High Side Outputs Value
This register holds the number of GCLK_TCCx clock cycles for the dead-time high side.

Bits 23:16 – DTLS[7:0] Dead-time Low Side Outputs Value
This register holds the number of GCLK_TCCx clock cycles for the dead-time low side.

Bits 8, 9, 10, 11 – DTIENx Dead-time Insertion Generator x Enable
Setting any of these bits enables the dead-time insertion generator for the corresponding output matrix. This will override the output matrix [x] and [x+WO_NUM/2], with the low side and high side waveform respectively.

Value	Description
0	No dead-time insertion override.
1	Dead time insertion override on signal outputs[x] and [x+WO_NUM/2], from matrix outputs[x] signal.

Bits 1:0 – OTMX[1:0] Output Matrix
These bits define the matrix routing of the TCC waveform generation outputs to the port pins, according to 41.6.3.7. Waveform Extension.

41.8.7 Driver Control

Name: DRVCTRL
Offset: 0x18
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
	FILTERVAL1[3:0]				FILTERVAL0[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INVEN7	INVEN6	INVEN5	INVEN4	INVEN3	INVEN2	INVEN1	INVEN0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NRV7	NRV6	NRV5	NRV4	NRV3	NRV2	NRV1	NRV0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	NRE7	NRE6	NRE5	NRE4	NRE3	NRE2	NRE1	NRE0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:28 – FILTERVAL1[3:0] Non-Recoverable Fault Input 1 Filter Value
These bits define the filter value applied on TCE1 event input line. When the TCE1 event input line is configured as a synchronous event, this value must be 0x0.

Bits 27:24 – FILTERVAL0[3:0] Non-Recoverable Fault Input 0 Filter Value
These bits define the filter value applied on TCE0 event input line. When the TCE0 event input line is configured as a synchronous event, this value must be 0x0.

Bits 16, 17, 18, 19, 20, 21, 22, 23 – INVENx Waveform Output x Inversion
These bits are used to select inversion on the output of channel x.
Writing a '1' to INVENx inverts output from WO[x].
Writing a '0' to INVENx disables inversion of output from WO[x].

Bits 8, 9, 10, 11, 12, 13, 14, 15 – NRVx NRVx Non-Recoverable State x Output Value
These bits define the value of the enabled override outputs, under non-recoverable fault condition.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – NREx Non-Recoverable State x Output Enable
These bits enable the override of individual outputs by NRVx value, under non-recoverable fault condition.

Value	Description
0	Non-recoverable fault tri-state the output.
1	Non-recoverable faults set the output to NRVx level.

41.8.8 Debug control

Name: DBGCTRL
Offset: 0x1E
Reset: 0x00
Property: PAC Write-Protection

Bit	7	6	5	4	3	2	1	0
						FDDBD		DBGRUN
Access						R/W		R/W
Reset						0		0

Bit 2 – FDDBD Fault Detection on Debug Break Detection

This bit is not affected by software Reset and must not be changed by software while the TCC is enabled.
By default this bit is zero, and the on-chip debug (OCD) fault protection is disabled. When this bit is written to '1', OCD break request from the OCD system will trigger non-recoverable fault. When this bit is set, OCD fault protection is enabled and OCD break request from the OCD system will trigger a non-recoverable fault.

Value	Description
0	No faults are generated when TCC is halted in Debug mode.
1	A non recoverable fault is generated and FAULTD flag is set when TCC is halted in Debug mode.

Bit 0 – DBGRUN Debug Running State

This bit is not affected by software Reset and must not be changed by software while the TCC is enabled.

Value	Description
0	The TCC is halted when the device is halted in Debug mode.
1	The TCC continues normal operation when the device is halted in Debug mode.

41.8.9 Event Control

Name: EVCTRL
Offset: 0x20
Reset: 0x00000000
Property: PAC Write-Protection, Enable-Protected

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	TCEI1	TCEI0	TCINV1	TCINV0		CNTEO	TRGEO	OVFEO
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
	CNTSEL[1:0]		EVACT1[2:0]			EVACT0[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 14, 15 – TCEI Timer/Counter Event Input x Enable
This bit is used to enable input event x to the TCC.

Value	Description
0	Incoming event x is disabled.
1	Incoming event x is enabled.

Bits 12, 13 – TCINV Timer/Counter Event x Invert Enable
This bit inverts the event x input.

Value	Description
0	Input event source x is not inverted.
1	Input event source x is inverted.

Bit 10 – CNTEO Timer/Counter Event Output Enable
This bit is used to enable the counter cycle event. When enabled, an event will be generated on begin or end of counter cycle depending of CNTSEL[1:0] settings.

Value	Description
0	Counter cycle output event is disabled and will not be generated.
1	Counter cycle output event is enabled and will be generated depend of CNTSEL[1:0] value.

Bit 9 – TRGEO Retrigger Event Output Enable
This bit is used to enable the counter retrigger event. When enabled, an event will be generated when the counter retriggers operation.

Value	Description
0	Counter retrigger event is disabled and will not be generated.
1	Counter retrigger event is enabled and will be generated for every counter retrigger.

Bit 8 – OVFEO Overflow/Underflow Event Output Enable
This bit is used to enable the overflow/underflow event. When enabled an event will be generated when the counter reaches the TOP or the ZERO value.

Value	Description
0	Overflow/underflow counter event is disabled and will not be generated.
1	Overflow/underflow counter event is enabled and will be generated for every counter overflow/ underflow.

Bits 7:6 – CNTSEL[1:0] Timer/Counter Interrupt and Event Output Selection

These bits define on which part of the counter cycle the counter event output is generated.

Value	Name	Description
0x0	BEGIN	An interrupt/event is generated at begin of each counter cycle
0x1	END	An interrupt/event is generated at end of each counter cycle
0x2	BETWEEN	An interrupt/event is generated between each counter cycle.
0x3	BOUNDARY	An interrupt/event is generated at begin of first counter cycle, and end of last counter cycle.

Bits 5:3 – EVACT1[2:0] Timer/Counter Event Input 1 Action

These bits define the action the TCC will perform on TCE1 event input.

Value	Name	Description
0x0	OFF	Event action disabled.
0x1	RETRIGGER	Start, restart or re-trigger TC on event
0x2	DIR (asynch)	Direction control
0x3	STOP	Stop TC on event
0x4	DEC	Decrement TC on event
0x5	PPW	Period captured into CC0 Pulse Width on CC1
0x6	PWP	Period captured into CC1 Pulse Width on CC0
0x7	FAULT	Non-recoverable Fault

Bits 2:0 – EVACT0[2:0] Timer/Counter Event Input 0 Action

These bits define the action the TCC will perform on TCE0 event input 0.

Value	Name	Description
0x0	OFF	Event action disabled.
0x1	RETRIGGER	Start, restart or re-trigger TC on event
0x2	COUNTEV	Count on event.
0x3	START	Start TC on event
0x4	INC	Increment TC on EVENT
0x5	COUNT (async)	Count on active state of asynchronous event
0x6		
0x7	FAULT	Non-recoverable Fault

41.8.10 Interrupt Enable Clear

Name: INTENCLR
Offset: 0x24
Reset: 0x00000000
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Set (INTENSET) register.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 15 – FAULT1 Non-Recoverable Fault x Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

Bit 14 – FAULT0 Non-Recoverable Fault x Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

Bit 13 – FAULTB Recoverable Fault B Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Recoverable Fault B Interrupt Disable/Enable bit, which disables the Recoverable Fault B interrupt.

Value	Description
0	The Recoverable Fault B interrupt is disabled.
1	The Recoverable Fault B interrupt is enabled.

Bit 12 – FAULTA Recoverable Fault A Interrupt Enable
Writing a '0' to this bit has no effect.

Writing a '1' to this bit will clear the Recoverable Fault A Interrupt Disable/Enable bit, which disables the Recoverable Fault A interrupt.

Value	Description
0	The Recoverable Fault A interrupt is disabled.
1	The Recoverable Fault A interrupt is enabled.

Bit 11 – DFS Non-Recoverable Debug Fault Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Debug Fault State Interrupt Disable/Enable bit, which disables the Debug Fault State interrupt.

Value	Description
0	The Debug Fault State interrupt is disabled.
1	The Debug Fault State interrupt is enabled.

Bit 10 – UFS Non-Recoverable Update Fault Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which disables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault interrupt is enabled.

Bit 3 – ERR Error Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Error Interrupt Disable/Enable bit, which disables the Compare interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 2 – CNT Counter Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Counter Interrupt Disable/Enable bit, which disables the Counter interrupt.

Value	Description
0	The Counter interrupt is disabled.
1	The Counter interrupt is enabled.

Bit 1 – TRG Retrigger Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Retrigger Interrupt Disable/Enable bit, which disables the Retrigger interrupt.

Value	Description
0	The Retrigger interrupt is disabled.
1	The Retrigger interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Overflow Interrupt Disable/Enable bit, which disables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

41.8.11 Interrupt Enable Set

Name: INTENSET
Offset: 0x28
Reset: 0x00000000
Property: PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Interrupt Enable Clear (INTENCLR) register.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 15 – FAULT1 Non-Recoverable Fault x Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Non-Recoverable Fault x Interrupt Disable/Enable bit, which enables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

Bit 14 – FAULT0 Non-Recoverable Fault x Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

Bit 13 – FAULTB Recoverable Fault B Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Recoverable Fault B Interrupt Disable/Enable bit, which enables the Recoverable Fault B interrupt.

Value	Description
0	The Recoverable Fault B interrupt is disabled.
1	The Recoverable Fault B interrupt is enabled.

Bit 12 – FAULTA Recoverable Fault A Interrupt Enable
Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Recoverable Fault A Interrupt Disable/Enable bit, which enables the Recoverable Fault A interrupt.

Value	Description
0	The Recoverable Fault A interrupt is disabled.
1	The Recoverable Fault A interrupt is enabled.

Bit 11 – DFS Non-Recoverable Debug Fault Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Debug Fault State Interrupt Disable/Enable bit, which enables the Debug Fault State interrupt.

Value	Description
0	The Debug Fault State interrupt is disabled.
1	The Debug Fault State interrupt is enabled.

Bit 10 – UFS Non-Recoverable Update Fault Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which disables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault interrupt is enabled.

Bit 3 – ERR Error Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Error Interrupt Disable/Enable bit, which enables the Compare interrupt.

Value	Description
0	The Error interrupt is disabled.
1	The Error interrupt is enabled.

Bit 2 – CNT Counter Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Retrigger Interrupt Disable/Enable bit, which enables the Counter interrupt.

Value	Description
0	The Counter interrupt is disabled.
1	The Counter interrupt is enabled.

Bit 1 – TRG Retrigger Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Retrigger Interrupt Disable/Enable bit, which enables the Retrigger interrupt.

Value	Description
0	The Retrigger interrupt is disabled.
1	The Retrigger interrupt is enabled.

Bit 0 – OVF Overflow Interrupt Enable

Writing a '0' to this bit has no effect.
Writing a '1' to this bit will set the Overflow Interrupt Disable/Enable bit, which enables the Overflow interrupt request.

Value	Description
0	The Overflow interrupt is disabled.
1	The Overflow interrupt is enabled.

41.8.12 Interrupt Flag Status and Clear

Name: INTFLAG
Offset: 0x2C
Reset: 0x00000000
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	DFS	UFS		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
					ERR	CNT	TRG	OVF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 15 – FAULT1 Non-Recoverable Fault x Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a Non-Recoverable Fault x occurs.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Non-Recoverable Fault x interrupt flag.

Bit 14 – FAULT0 Non-Recoverable Fault x Interrupt Enable
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Non-Recoverable Fault x Interrupt Disable/Enable bit, which disables the Non-Recoverable Fault x interrupt.

Value	Description
0	The Non-Recoverable Fault x interrupt is disabled.
1	The Non-Recoverable Fault x interrupt is enabled.

Bit 13 – FAULTB Recoverable Fault B Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a Recoverable Fault B occurs.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

Bit 12 – FAULTA Recoverable Fault A Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a Recoverable Fault B occurs.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Recoverable Fault B interrupt flag.

Bit 11 – DFS Non-Recoverable Debug Fault State Interrupt Flag
This flag is set on the next CLK_TCC_COUNT cycle after a Debug Fault State occurs.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Debug Fault State interrupt flag.

Bit 10 – UFS Non-Recoverable Update Fault Interrupt Enable

This flag is set when the RAMP index changes and the Lock Update bit is set (CTRLBSET.LUPD).
Writing a '0' to this bit has no effect.
Writing a '1' to this bit will clear the Non-Recoverable Update Fault Interrupt Disable/Enable bit, which disables the Non-Recoverable Update Fault interrupt.

Value	Description
0	The Non-Recoverable Update Fault interrupt is disabled.
1	The Non-Recoverable Update Fault interrupt is enabled.

Bit 3 – ERR Error Interrupt Flag

This flag is set if a new capture occurs on a channel when the corresponding Match or Capture Channel x interrupt flag is one. In which case, there is nowhere to store the new capture.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the error interrupt flag.

Bit 2 – CNT Counter Interrupt Flag

This flag is set on the next CLK_TCC_COUNT cycle after a counter event occurs.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the CNT interrupt flag.

Bit 1 – TRG Retrigger Interrupt Flag

This flag is set on the next CLK_TCC_COUNT cycle after a counter retrigger occurs.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the re-trigger interrupt flag.

Bit 0 – OVF Overflow Interrupt Flag

This flag is set on the next CLK_TCC_COUNT cycle after an overflow condition occurs.
Writing a '0' to this bit has no effect.
Writing a '1' to this bit clears the Overflow interrupt flag.

41.8.13 Status

Name: STATUS
Offset: 0x30
Reset: 0x00000001
Property: -

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FAULT1	FAULT0	FAULTB	FAULTA	FAULT1IN	FAULT0IN	FAULTBIN	FAULTAIN
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PERBV		PATTBV		DFS		IDX	STOP
Access	R/W		R/W		R/W		R	R
Reset	0		0		0		0	1

Bits 14, 15 – FAULT Non-recoverable Fault x State
This bit is set by hardware as soon as non-recoverable Fault x condition occurs.
This bit is cleared by writing a one to this bit and when the corresponding FAULTxIN status bit is low.
Once this bit is clear, the timer/counter will restart from the last COUNT value. To restart the timer/counter from BOTTOM, the timer/counter restart command must be executed before clearing the corresponding STATEx bit. For further details on timer/counter commands, refer to the available commands description (CTRLBSET.CMD).

Bit 13 – FAULTB Recoverable Fault B State
This bit is set by hardware as soon as recoverable Fault B condition occurs.
This bit can be cleared by hardware when the Fault B action is resumed or by writing a '1' to this bit when the corresponding FAULTBIN bit is low. If the software halt command is enabled (FAULTB.HALT=SW), clearing this bit will release the timer/counter.

Bit 12 – FAULTA Recoverable Fault A State
This bit is set by hardware as soon as the recoverable Fault A condition occurs.
This bit can be cleared by hardware when the Fault A action is resumed or by writing a '1' to this bit when the corresponding FAULTAIN bit is low. If the software halt command is enabled (FAULTA.HALT=SW), clearing this bit will release the timer/counter.

Bit 11 – FAULT1IN Non-Recoverable Fault 1 Input
This bit is set while an active Non-Recoverable Fault 1 input is present.

Bit 10 – FAULT0IN Non-Recoverable Fault 0 Input
This bit is set while an active Non-Recoverable Fault 0 input is present.

Bit 9 – FAULTBIN Recoverable Fault B Input
This bit is set while an active Recoverable Fault B input is present.

Bit 8 – FAULTAIN Recoverable Fault A Input
This bit is set while an active Recoverable Fault A input is present.

Bit 7 – PERBV Period Buffer Valid
This bit is set when a new value is written to the PERB register. This bit is automatically cleared by hardware on the UPDATE condition when CTRLB.LUPD is set or by writing a '1' to this bit.

Bit 5 – PATTBV Pattern Generator Value Buffer Valid
This bit is set when a new value is written to the PATTB register. This bit is automatically cleared by hardware on the UPDATE condition when CTRLB.LUPD is set or by writing a '1' to this bit.

Bit 3 – DFS Debug Fault State
This bit is set by hardware in Debug mode when the DDBGCTRL.FDDBD bit is set. The bit is cleared by writing a '1' to this bit and when the TCC is not in Debug mode.
When the bit is set, the counter is halted and the Waveforms state depends on the DRVCTRL.NRE and DRVCTRL.NRV registers.

Bit 1 – IDX Ramp Index
In RAMP2 and RAMP2A operation, the bit is cleared during the cycle A and set during the cycle B. In RAMP1 operation, the bit always reads zero. See *Ramp Operations* from Related Links..

Bit 0 – STOP Stop
This bit is set when the TCC is disabled either on a STOP command or on an UPDATE condition when One-Shot operation mode is enabled (CTRLBSET.ONESHOT=1).
This bit is clear on the next incoming counter increment or decrement.

Value	Description
0	Counter is running.
1	Counter is stopped.

Related Links
41.6.3.4. Ramp Operations

41.8.14 Counter Value

Name: COUNT
Offset: 0x34
Reset: 0x00000000
Property: PAC Write-Protection, Write-Synchronized

Note: Prior to any read access, this register must be synchronized by user by writing the according TCC Command value to the Control B Set register (CTRLBSET.CMD = READSYNC).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

41.8.15 Pattern

Name: PATT
Offset: 0x38
Reset: 0x0000
Property: Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	PGV7	PGV6	PGV5	PGV4	PGV3	PGV2	PGV1	PGV0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PGE7	PGE6	PGE5	PGE4	PGE3	PGE2	PGE1	PGE0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 8, 9, 10, 11, 12, 13, 14, 15 – PGV Pattern Generation Output Value
This register holds the values of pattern for each waveform output.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PGE Pattern Generation Output Enable
This register holds the enable status of pattern generation for each waveform output. A bit written to '1' will override the corresponding SWAP output with the corresponding PGVn value.

41.8.16 Waveform

Name: WAVE
Offset: 0x3C
Reset: 0x00000000
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
					SWAP3	SWAP2	SWAP1	SWAP0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					CICCEN3	CICCEN2	CICCEN1	CICCEN0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CIPEREN					WAVEGEN[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bits 24, 25, 26, 27 – SWAP Swap DTI Output Pair x
Setting these bits enables the output swap of DTI outputs [x] and [x+WO_NUM/2]. Note, the DTIxEN settings will not affect the swap operation.

Bits 8, 9, 10, 11 – CICCEN Circular CC Enable x
Setting this bit enables the compare circular buffer option on the first four Compare/Capture channels. When the bit is set, the CCx register value is copied-back into the CCx register on the UPDATE condition.

Bit 7 – CIPEREN Circular Period Enable
Setting this bit enables the period circular buffer option. When the bit is set, the PER register value is copied-back into the PERB register on UPDATE condition.
These bits select the Ramp operation (RAMP). These bits are not synchronized.

Value	Name	Description
0x0	RAMP1	RAMP1 operation
0x1	RAMP2A	Alternative RAMP2 operation
0x2	RAMP2	RAMP2 operation
0x3	RAMP2C	

Bits 2:0 – WAVEGEN[2:0] Waveform Generation Operation
These bits select the waveform generation operation. The settings impact the top value and control if the frequency or PWM waveform generation must be used. These bits are not synchronized.

Value	Name	Description						
		Operation	Top	Update	Waveform Output On Match	Waveform Output On Update	OVFIF/Event Up Down	
0x0	NFRQ	Normal Frequency	PER	TOP/Zero	Toggle	Stable	TOP	Zero
0x1	MFRQ	Match Frequency	CC0	TOP/Zero	Toggle	Stable	TOP	Zero
0x2	NPWM	Normal PWM	PER	TOP/Zero	Set	Clear	TOP	Zero

.....continued

Value	Name	Description						
		Operation	Top	Update	Waveform Output On Match	Waveform Output On Update	OVFIF/Event Up Down	
0x3								
0x4	DSCRITICAL	Dual-slope PWM	PER	Zero	~DIR	Stable	—	Zero
0x5	DSBOTTOM	Dual-slope PWM	PER	Zero	~DIR	Stable	—	Zero
0x6	DSBOTH	Dual-slope PWM	PER	TOP & Zero	~DIR	Stable	TOP	Zero
0x7	DSTOP	Dual-slope PWM	PER	Zero	~DIR	Stable	TOP	—

41.8.17 Period Value

Name: PER
Offset: 0x40
Reset: 0xFFFFFFFF
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			DITHER[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	1	1

Bits 5:0 – DITHER[5:0] Dithering Cycle Number
These bits hold the number of extra cycles that are added on the PWM pulse period every 64 PWM frames.
Note: This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

41.8.18 Pattern Buffer

Name: PATTB
Offset: 0x64
Reset: 0x0000
Property: Write-Synchronized

Bit	15	14	13	12	11	10	9	8
	PGVB7	PGVB6	PGVB5	PGVB4	PGVB3	PGVB2	PGVB1	PGVB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PGEB7	PGEB6	PGEB5	PGEB4	PGEB3	PGEB2	PGEB1	PGEB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 8, 9, 10, 11, 12, 13, 14, 15 – PGVB Pattern Generation Output Value Buffer
This register is the buffer for the PGV register. If double buffering is used, valid content in this register is copied to the PGV register on an UPDATE condition.

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PGEB Pattern Generation Output Enable Buffer
This register is the buffer of the PGE register. If double buffering is used, valid content in this register is copied into the PGE register at an UPDATE condition.

41.8.19 Period Buffer Value

Name: PERB
Offset: 0x6C
Reset: 0xFFFFFFFF
Property: Write-Synchronized

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			DITHERB[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	1	1

Bits 5:0 – DITHERB[5:0] Dithering Buffer Cycle Number
These bits represent the PER.DITHER bits buffer. When the double buffering is enabled, the value of this bit field is copied to the PER.DITHER bits on an UPDATE condition.
Note: This bit field consists of the n LSB of the register. n is dependent on the value of the Resolution bits in the Control A register (CTRLA.RESOLUTION):

CTRLA.RESOLUTION	Bits [n:0]
0x0 - NONE	-
0x1 - DITH4	3:0
0x2 - DITH5	4:0
0x3 - DITH6	5:0 (depicted)

42. Zigbee Bluetooth Radio Subsystem (ZBT)

42.1 Overview

The PIC32CX-BZ2 and the WBZ45 support on-chip IEEE 802.15.4 and 802.15.3 compliant Zigbee, Bluetooth® Low Energy 5.2 interface with integrated transceivers. The Wireless Subsystem block is comprised of on-chip Zigbee and Bluetooth 5.0 BBP/MAC, shared RF transceiver and Radio Arbiter. This section provides key features of the on-chip Wireless modules.

With integrated Ultra Low Power 2.4 GHz ISM band single transceiver, dual modem and dual MAC, the Radio supports dual Zigbee and Bluetooth 5.2 link protocols. An onboard intelligent Radio Arbiter HW module establishes both the links simultaneously using a single Radio Transmit/Receive with programmable QoS. The RF transceiver includes dual Power Amplifiers architecture and TR Switch. Therefore, medium to high power application use cases are supported without external FEM.

Arbitration between Application, Bluetooth link stack, Zigbee link stack and miscellaneous maintenance tasks are handled on the Cortex M4F (w/ DSP, FPU) CPU using available on-chip system memory resources via RTOS.

Note: Unique Bluetooth and 802.15.4 MAC address are available in OTP memory region. The software SDK and operational stacks provided by Microchip provide API's to access these addresses.

42.2 Features

2.4 GHz RF Transceiver

- Integrated 2.4 GHz Ultra Low Power RF Transceiver shared between Bluetooth and Zigbee Modems and Link (MAC) Controllers
- Integrated 16 MHz ±20 ppm Crystal Oscillator (External Low Cost Crystal)
- Two PA Design Architecture (LPA (+4 dBm) and MPA (+12 dBm)) to improve TX power efficiency
- Low RBOM Two-port TRX RFFE Architecture
  - Integrated balun (single-ended RF output) and TRX Switch
- Hardware Radio Arbiter with programmable QoS:
  - Resolution: up to per packet level
  - Time-division coexistence between Bluetooth and 802.15.4
  - Based on shared transceiver and antenna
  - Maintains connections of 802.15.4 and Bluetooth simultaneously

Bluetooth

- Bluetooth Low Energy 5.2 Certified
- Up to +12 dBm Programmable Transmit Output Power
- Typical Receiver Power Sensitivity:
  - -95 dBm for Bluetooth Low Energy 1 Mbps
  - -92 dBm for Bluetooth Low Energy 2 Mbps
  - -102 dBm for Bluetooth Low Energy 125 Kbps
  - -99 dBm for Bluetooth Low Energy 500 Kbps
  - Digital RSSI indicator (-90 dBm ~ -30 dBm)
- Bluetooth Supported Features:
  - 2M uncoded PHY
  - Long range (Coded PHY)
  - Channel selection algorithm #2
  - Advertising extensions, offloads CPU with hardware-based scheduler
  - High duty cycle non-connectible advertising
  - Data length extensions

  - Secure connections
  - Privacy upgrades (with hardware white-list support)
- ECDH P256 Hardware Engine for Link Key Generation when Bluetooth Pairing
- AES128 Hardware Module for Real-Time Bluetooth Payload Data Encryption
- HCI Interface via UART
- Bluetooth Low Energy Profiles:
  - Bluetooth Low Energy peripheral and central roles
  - Bluetooth Low Energy APIs for application layer to implement standard or customize GATT based profiles/ services
  - Microchip Transparent UART Service
  - Battery Service
  - Device Information Service
  - Multi-link and multi-role
- Bluetooth Low Energy Services:
  - Provisioning
  - Over-the-Air (OTA) update (also known as DFU)
  - Advertisement/Beacon
  - Personalized configuration
  - Alert notification service

802.15.4/Zigbee

- 802.15.4/Zigbee PSDU data rate: 250 Kbps
- Programmable RX Mode:
  - -103 dBm RX sensitivity in the Continuous mode
  - -98 dBm sensitivity in the RPC mode
  - RPC mode provides lower power consumption in RX mode to support California Green Energy Specification at the system level
- Hardware Assisted MAC:
  - Auto acknowledge
  - Auto retry
  - Channel access back-off
- SFD Detection, Spreading, De-spreading, Framing, CRC-16 Computation
- Independent TX/RX Buffers for improved CPU Offloading while Handling Zigbee Data:
  - 128-byte TX and 128-byte RX frame buffer
- Hardware Security:
  - Advanced Encryption Standard (AES)
  - True Random Number Generator (TRNG)
- Zigbee Stack Support:
  - Zigbee 3.0 ready
  - Zigbee Pro 2015
  - Zigbee green power support (proxy, sink and multi-sensor)

Proprietary(1)

- 500 kbps and 1 Mbps are 2.4 GHz Proprietary with DSSS
- 2 Mpbs are 2.4 GHz Proprietary without DSSS
- TX Output Power up to +12 dBm
- Receiver Sensitivity up to -96 dBm
- Hardware Assisted MAC:
  - Auto acknowledge
  - Auto retry

  - Channel access back-off
- SFD Detection, Spreading, De-spreading, Framing, CRC-16 Computation
- Independent TX/RX Buffers for improved CPU Offloading while Handling Zigbee Data:
  - 128-byte TX and 128-byte RX frame buffer
- Hardware Security:
  - Advanced Encryption Standard (AES)
  - True Random Number Generator (TRNG)

Note:

1. Proprietary modes are compatible to AT86RF233 modes of similar data rate.

42.3 Wireless Subsystem Top Level Diagram

Figure 42-1. Wireless Subsystem Top Level Diagram

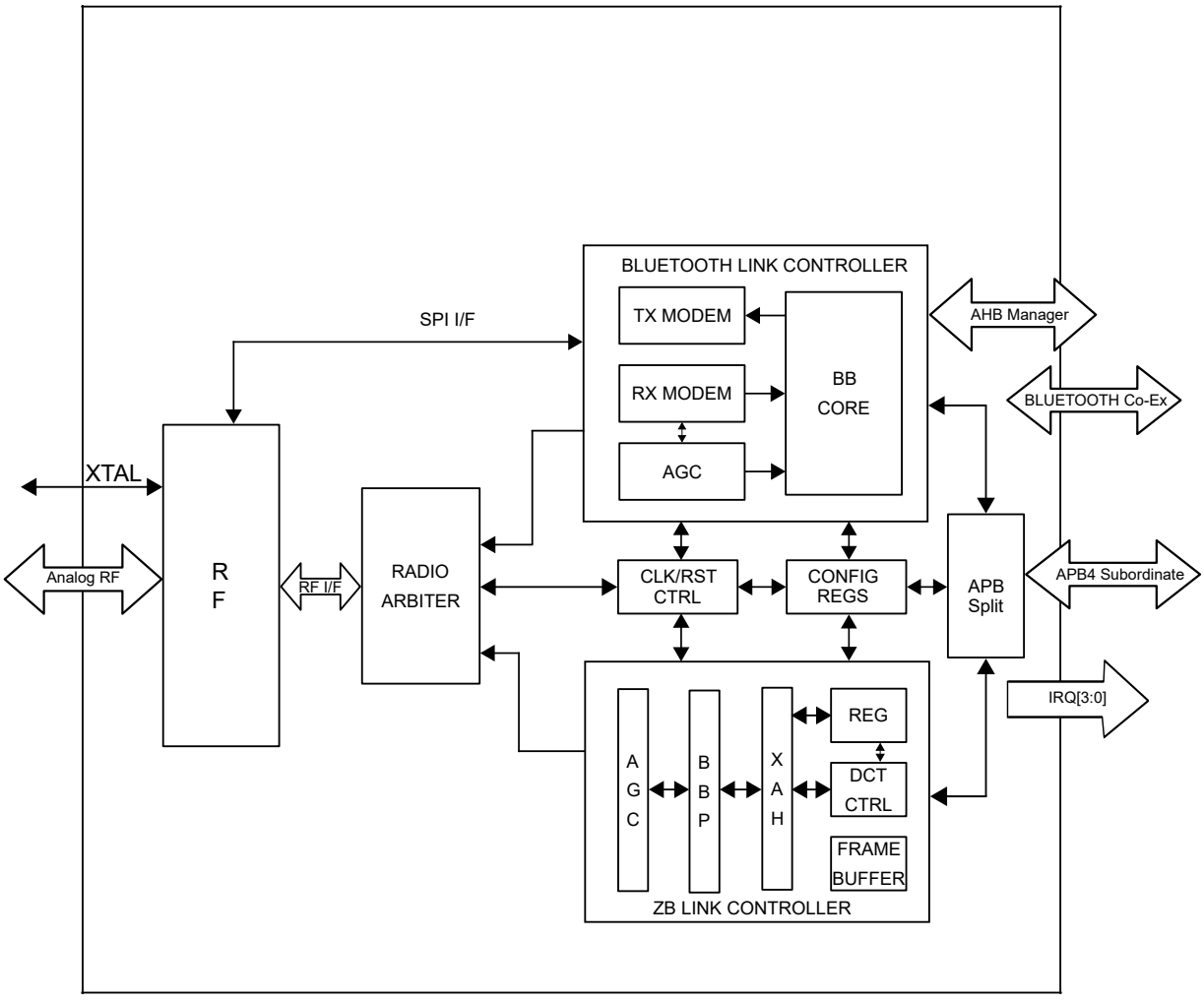
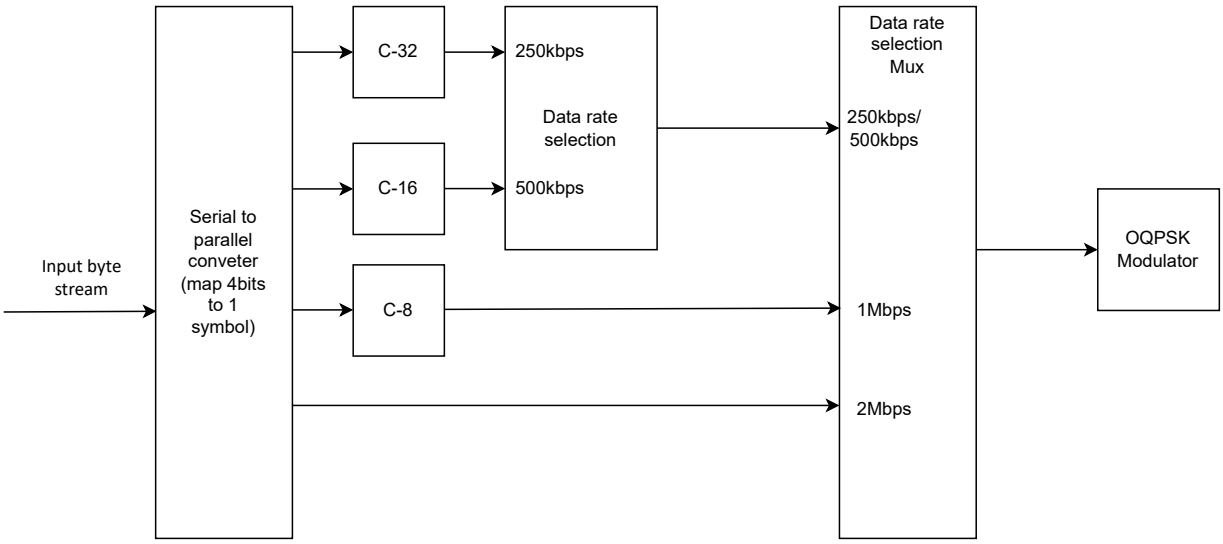


Figure 42-2. Zigbee Baseband Processor Block Diagram



42.4 Bluetooth Link Controller

The APB interface provides access to internal register banks of the macro. These registers are programmed by the host (firmware) to set various configurations, trigger commands, read status, service interrupts and other functions.

All Bluetooth operations are carried out as transmit or receive tasks within the Link controller. There are several task controllers:

- Firmware – Firmware can trigger tasks by writing the task controller registers.
- Hardware Schedule Controller – This is Bluetooth 5.2 Bluetooth Low Energy advertisement scheduler controller (Adv. role).
- Hardware Scanner – This is Bluetooth 5.2 Bluetooth Low Energy advertisement scanner (Central role).
- Bluetooth Low Energy advertisement controller – This is Bluetooth 5.2 advertisement controller (Adv. role).

The requests from the task controllers are arbitrated and is carried out by the task controller.

42.4.1 Hardware Schedule Controller

Hardware Schedule controller is a hardware accelerator that offloads the Bluetooth 5.2 advertising task from the firmware and executes it in hardware completely.

42.4.2 Hardware Scanner

Hardware Scanner is another hardware accelerator to be used in the Bluetooth Low Energy/central mode. This offloads the Bluetooth Low Energy task of scanning for advertisements from the firmware. This enables fast scanning and quick connections.

42.4.3 Bluetooth Low Energy Advertisement Controller

Bluetooth Low Energy advertisement controller is a hardware controller that supports offloading of Bluetooth 5.2 advertisements. This is a simpler advertising scenario requiring much smaller memory.

The task controller manages overall TX/RX functionality and starts TX control FSM / RX control FSM in the baseband as per requirement to transmit/receive a packet. The task controller also manages the coexistence interface with Wi-Fi. The task controller can also be used directly by the firmware to measure the RSSI across various Bluetooth channels, which can, then, be used to generate an AFH channel map.

42.4.4 Bluetooth Clock Controller

The Bluetooth clock controller/Slot timer provides information on Bluetooth slots and the slot boundary. The Bluetooth clock synchronization and clock adjustment due to RX window uncertainty is also handled by this controller. It controls the timings for Bluetooth Low Energy non-TIFS tasks.

TX/RX control FSM (State machines) provide sequencing and control of various phases of a task. These control various elements in the TX and RX data path.

The Crypto engine is used for AES encryption and decryption of transmit and receive data. This performs inline encryption as the data passes through the data path in the BB link controller. This controller is also used to perform offline CMAC operation directly under application firmware control. This controller is also used by the Bluetooth Low Energy privacy controller for the generation and decoding of hash needed for resolving private addresses.

42.4.5 Bluetooth Low Energy Privacy Controller

Bluetooth Low Energy Privacy controller is to implement the Bluetooth Low Energy privacy feature. It can be used in offline mode by the firmware to generate a Resolvable Private Address (RPA) to be used for a transmission. And it is also used inline by the RX Control FSM to resolve the RPA received in the packet before implementing the device filtering using the white list.

The Channel Hop Controller implements the logic needed to compute the next RF channel frequency to be used.

A dedicated math unit for big number (128-bit) operations is available for firmware to handle large number operations. This is programmed and used by the firmware directly. There is no direct use of this block by any other hardware block.

For simple, secure pairing, the P256 ECC module is available to perform computationally intensive key-matching procedures in hardware.

The transmit memory read controller reads the data from the common memory and feeds them to the transmit data path for Bluetooth PDU creations and transmit. The receive memory write controller receives data from the RX data path and writes them to the common memory.

42.5 Zigbee/Proprietary Data Rate Link Controller

The Zigbee Link Controller implements Zigbee 3.0 (802.15.4-2011) MAC and baseband functionality in hardware.

The CPU interface (APB Subordinate) provides access to internal registers of the macro. These registers are programmed by the host (firmware) to kick off transmit/receive functionality along with the programming of other features. The APB Subordinate contains shadow registers for the status registers in the design for single-cycle, contention-free read accesses. Interrupt controller logic also resides in the APB subordinate module.

Separate 128-byte frame buffers are provided for TX and RX.

The XAH module implements basic MAC functionality as well as a hardware accelerator for features such as automatic acknowledgement, CSMA-CA and retransmission, automatic FCS check and so on.

The BBP module implements Offset-QPSK PHY with 250 kb/s PSDU data rate. It also supports proprietary 500 kbps OQPSK PHY, 1 Mbps MSK PHY, and a 2 Mbps MSK non-spreading PHY.

42.5.1 Transmit Operation

A frame transmission comprises of two actions: a write to Frame Buffer and the transmission of its contents. Both actions can be run in parallel if required by critical protocol timing.

42.5.2 Receive Operation

A frame reception comprises of two actions: the transceiver listens for, receives and demodulates the frame to the Frame Buffer and signals the reception to the microcontroller. After that process, the microcontroller can read the available frame data from the Frame Buffer via the APB interface.

42.6 Radio Arbiter

The Radio Arbiter determines the ownership of the Radio between Zigbee and Bluetooth link controllers. It also generates control to select between Zigbee/Bluetooth to drive the controls and data to or from the RF.

By default, the Arbiter is IDLE and no Link controller has the ownership. The general intention of the design is to provide a radio arbiter that is flexible enough that the arbitration schemes can be tuned by the firmware with real application Bluetooth/Zigbee scenarios.

All design and intellectual property parts of this arbiter design are property of Microchip Technology Inc., released under appropriate NDA.

42.6.1 Arbiter Modes

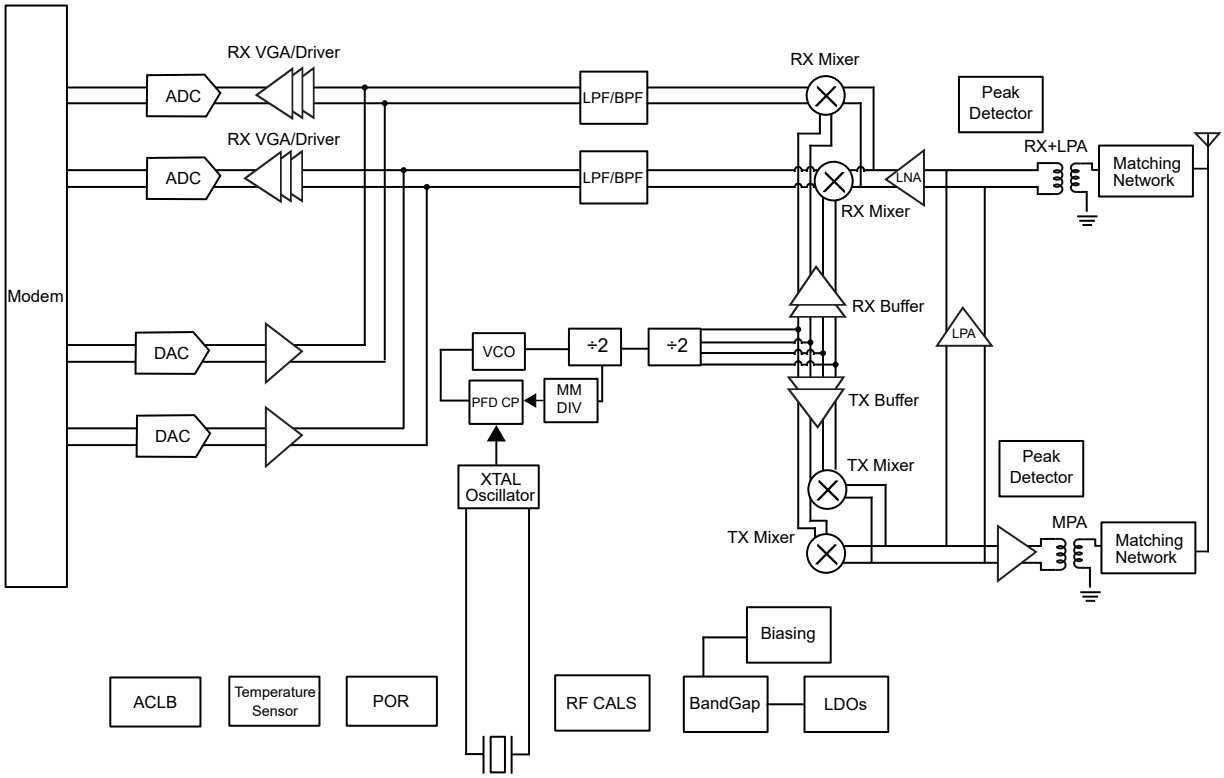
The following are the modes supported by the arbiter:

- Bluetooth static – Radio ownership is with the Bluetooth Link controller
- Zigbee static – Radio ownership is with the Zigbee Link controller
- Dynamic – Radio ownership decided dynamically at every arbitration event

42.7 RF Physical Layer

The top level block diagram of the transceiver architecture is as follows.

Figure 42-3. Transceiver Architecture



42.8 Frequency Synthesizer

The PIC32CX-BZ2 and the WBZ45 require 16 MHz for generating the local oscillator frequency between the 2.4 GHz to 2.48 GHz for the transmitter and receiver. The frequency synthesizer integrates all the passives for the loop filter inside the chip.

42.8.1 Transmit Mixer and Power Amplifier

The PIC32CX-BZ2 and WBZ45 implement a direct conversion I/Q transmitter. The I/Q from the low pass filter of the TX path is converted to the desired output frequency through an I/Q mixer that operates on the LO frequency from the Synthesizer. The VCO operates at 4x the LO frequency.

The supply for LPA and MPA is through separated power supply pins that can be connected to the output from the PMU with necessary decoupling.

The LPA is a single stage amplifier that provides ~6 dBm output power in standalone mode and ~4 dBm in LPA/MPA shared mode. Both LPA and MPA implement an on-chip BALUN. The LPA pin also acts as the input to the RX section.

MPA is a two-stage power amplifier designed to provide a maximum output power of ~12 dBm. MPA also implements an on-chip BALUN to get a single-ended TX output. MPA is NOT internally connected to the receiver.
It is mandatory that the LPA pin be used for the Receiver path. On a typical application, the LPA and MPA can be connected directly, eliminating the need for an external switch.

42.8.2 Receiver

The receiver path starts with the LPA pin, which is shared with the LPA path and the LNA. PIC32CX-BZ2 implements a low-IF differential receiver. The synthesizer generates the LO for down-conversion on the mixer, which goes through a BPF filter to the ADC. The Receiver implements an AGC to adjust the LNA gain depending upon the input signal level.

42.9 RFLDO

The RF section has multiple LDOs to power the various power domains of the RF subsystem, All these LDOs can be powered up from the MLDO/DC-DC mode of the PMU and can get 1.35V and generate 1.2V internally to feed the corresponding RF sections. These LDOs need a bypass capacitor on the output for their operation. See *Power Subsystem* from Related Links.

- RF-PLL
- BUCK-LPA
- BUCK-MPA
- BUCK-BB

Related Links

7. Power Subsystem

43. Electrical Characteristics

This chapter provides the Electrical Specification and Characteristic of PIC32CX-BZ2 and WBZ45 Module across the operating temperature range of the product.

43.1 Absolute Maximum Electrical Characteristics

Exposure to these maximum rating conditions for extended periods may affect device reliability. Functional operation of the device at these or any other conditions above the parameters indicated in the operation listings of this specification is not implied.

Table 43-1. Absolute Maximum Ratings

Parameter	Value
Ambient temperature under bias(Note)	-40°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on V_{DD} / V_{DDIO} with respect to GND	-0.3V to +4.0V
Voltage on any tolerant pins, with respect to GND	-0.3V to (V_{DD} + 0.3V)
Maximum current out of GND pins	300 mA
Maximum current into V_{DD} pins(2)	300 mA
Maximum output current sourced/sunk by any Low Current Mode I/O pin (4x drive strength)	10 mA
Maximum output current sourced/sunk by any High Current Mode I/O pin (8x drive strength)	15 mA
Maximum current sink by all ports	120 mA
Maximum current sourced by all ports(2)	120 mA
ESD Qualification	
Human Body Model (HBM) per JESD22-A114	2000V
Charged Device Model (CDM) (ANSI/ESD STM 5.3.1)…(All pins / Corner pins)	+500V/- 500V

Notes:

1. Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.
2. Maximum allowable current is a function of device maximum power dissipation (See the *Thermal Operating Conditions* table in the *Thermal Specifications* from Related Links).

Related Links

43.3. Thermal Specifications

43.2 DC Electrical Characteristics

Table 43-2. Operating Frequency VS. Voltage

Param. No.	V_{DDIO}, V_{DDANA} Range	Temp. Range (in °C)	Max. MCU Frequency	Comments
DC_5	1.9V to 3.6V	-40°C to +85°C	64 MHz	Industrial

.....continued				
Param. No.	**V_{DDIO}, V_{DDANA} Range**	**Temp. Range (in °C)**	**Max. MCU Frequency**	**Comments**
DC_7	1.9V to 3.6V	-40°C to +125°C	64 MHz	Extended

Note: The same voltage must be applied to V_{DDIN} and V_{DDANA}.

43.3 Thermal Specifications

Table 43-3. Thermal Operating Conditions

Rating	Symbol	Min.	Typ	Max.	Unit
Industrial Temperature Devices:					
Operating ambient temperature range	T_A	-40	—	+85	°C
Operating junction temperature range	T_J	-40	—	+105	°C
V-Temp Temperature Devices:					
Operating ambient temperature range	T_A	-40	—	+105	°C
Operating junction temperature range	T_J	-40	—	+125	°C
Extended Temperature Range:					
Operating ambient temperature range	T_A	-40	—	+125	°C
Operating junction temperature range	T_J	-40	—	+135	°C
Maximum allowed power dissipation	P_{DMAX}	$(T_J - T_A)/\Theta_{JA}$			W

43.4 Active Current Consumption DC Electrical Specifications

Table 43-4. Active Current Consumption DC Electrical Specifications

DC Characteristics				**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**			
Param. No.	**Symbol**	**Characteristics**	**Clock/Freq**	**Typ (1)**	**Max.**	**Units**	**Conditions**
APWR_1	I_{DD_ACTIVE} (2,3)	MCU I_{DD} in Active mode w/LDO mode selected	PLL 64 MHz	131	—	µA/MHz	V_{DD} = V_{DDANA} = 3.3V
APWR_2			FRC 8 MHz	443			
APWR_3		MCU I_{DD} in Active mode w/ BUCK mode selected	PLL 64 MHz	76.8			
APWR_4			FRC 8 MHz	300			

Figure 43-1. Run Mode Current Consumption in Buck Mode at PLL 64 MHz

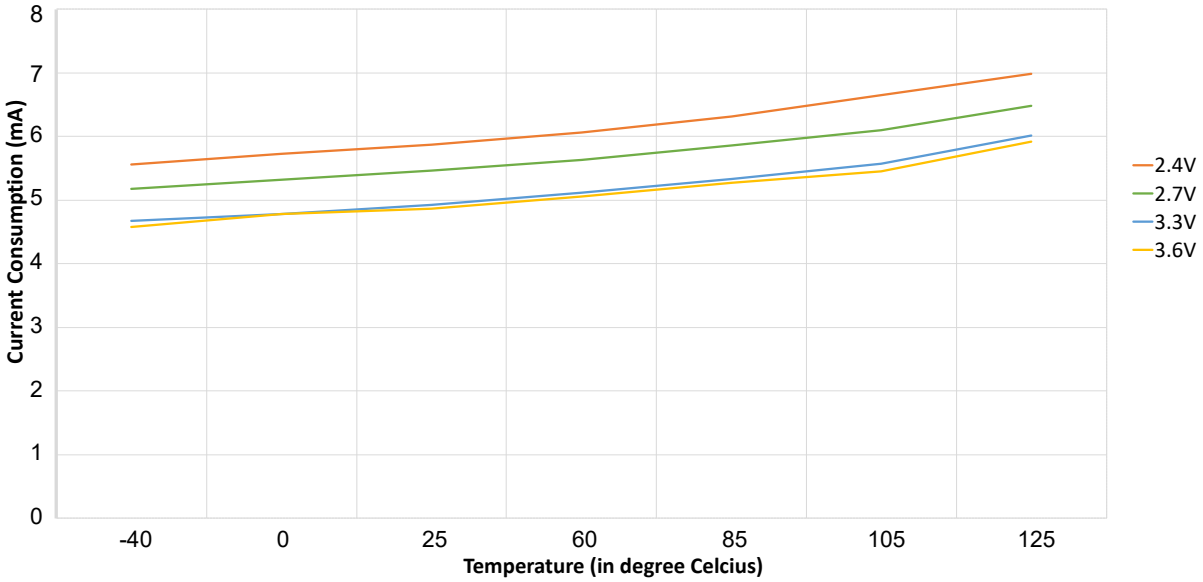
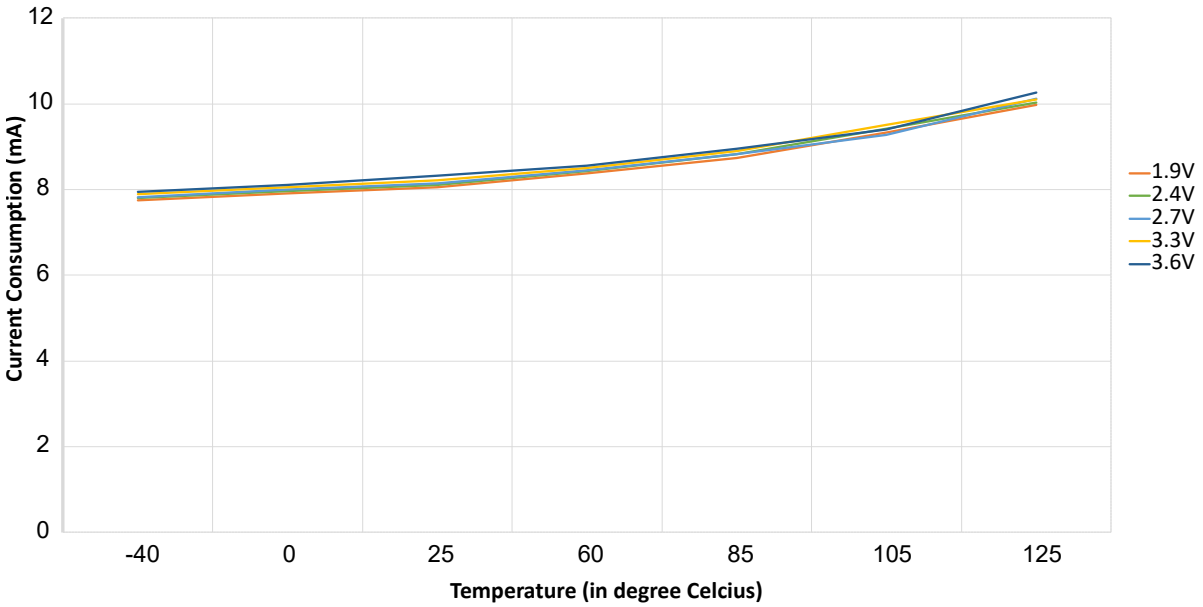


Figure 43-2. Run Mode Current Consumption in MLDO Mode at PLL 64 MHz



Notes:

1. Typical value measured at 3.3V and 25℃.
2. The test conditions are as follows:
   – All GPIO except RPA3 pulled up. RPA3 is pulled down.
   – RF system in IDLE state. Not transmitting or receiving packets.
   – All peripherals are disabled.
   – All PB clocks are divided by 16.
   – LPRC is set as LPCLK.
   – SOSC is disabled.
   – PMU 1 MHz clock is derived from FRC. FRC is divided by 8.
   – Cache is enabled and configured to wait states PFMWS[3:0] as 0xF.
   – Backup RAM in the Retention mode.
   – DSU is disconnected.
3. MCU running while(1) loop with 50 NOP instructions.

43.5 Idle Current Consumption DC Electrical Specifications

Table 43-5. Idle Current Consumption DC Electrical Specifications

DC Characteristics				Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp			
Param. No.	**Symbol**	**Characteristics**	**Clock/Freq**	**Typ(1)**	**Max.**	**Units**	**Conditions (3)**
IPWR_1	I_{DD_IDLE} (2)	MCU I_{DD} in IDLE mode w/LDO mode selected	PLL 64 MHz	83	—	µA/MHz	V_{DDIO} = V_{DDANA} = 3.3V
IPWR_2			FRC 8 MHz	336			
IPWR_3		MCU I_{DD} in IDLE mode w/BUCK mode selected	PLL 64 MHz	52.8	—		
IPWR_4			FRC 8 MHz	282			

Figure 43-3. Idle Mode Current Consumption in Buck Mode at PLL 64 MHz

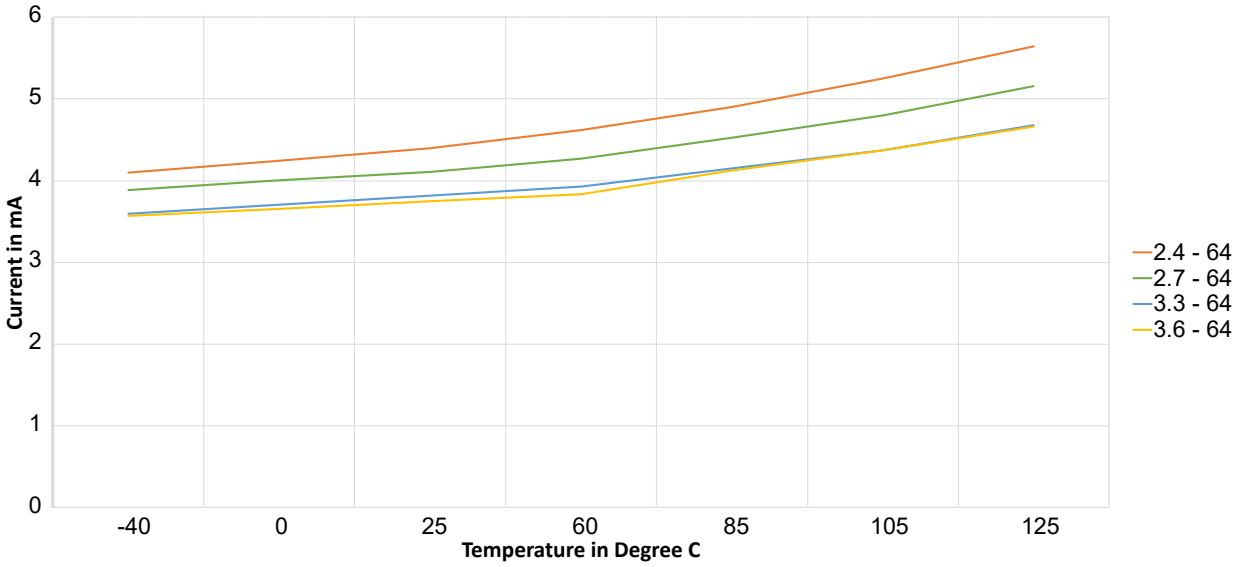
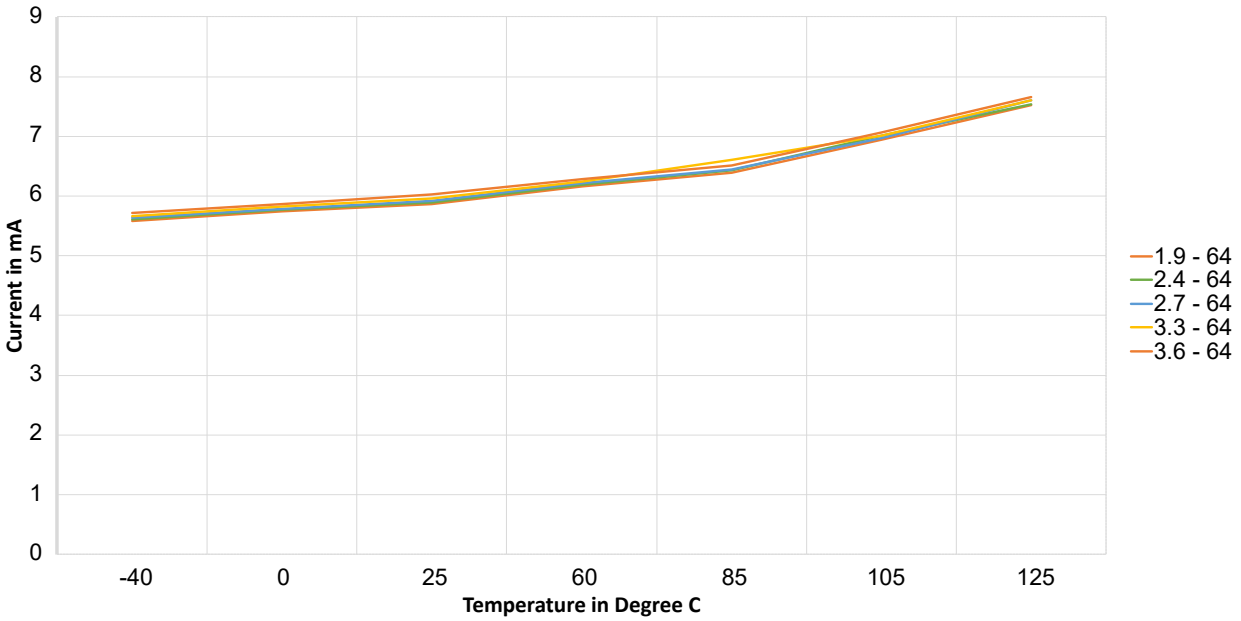


Figure 43-4. Idle Mode Current Consumption in MLDO Mode at PLL 64 MHz



Notes:

1. Typical value measured during characterization at 3.3V and 25℃.
2. The test conditions are as follows:
   - All GPIO except RPA3 are pulled up. RPA3 is pulled down.
   - All peripherals are disabled.
   - All PB clocks are divided by 16.
   - LPRC is set as LPCLK.
   - SOSC is disabled.
   - PMU 1 MHz clock is derived from FRC. FRC is divided by 8.
   - Cache is enabled and configured to wait states PFMWS[3:0] as 0xF.
   - Backup RAM in the Retention mode.
   - DSU is disconnected.
   - RF system clock is gated.
   - Entry to the Sleep mode is disabled and WFI instruction is executed.
   - No active RF transmission or reception during current measurement.

43.6 Sleep Current Consumption DC Electrical Specifications

Table 43-6. Sleep Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	Symbol	Characteristics	V_{DDIO}	Typ(1)	Max.	Units	Conditions
SPWR_1	I_{DD_SLEEP}	MCU I_{DD} in Sleep mode w/LDO mode selected	3.3V	950	—	µA	XTAL ON
			3.3V	620	—	µA	XTAL OFF
SPWR_29		MCU I_{DD} in Sleep mode w/ BUCK mode selected	3.3V	470	—	µA	XTAL ON
			3.3V	450	—	µA	XTAL OFF

Figure 43-5. Sleep Current with XTAL_OFF_MLDO

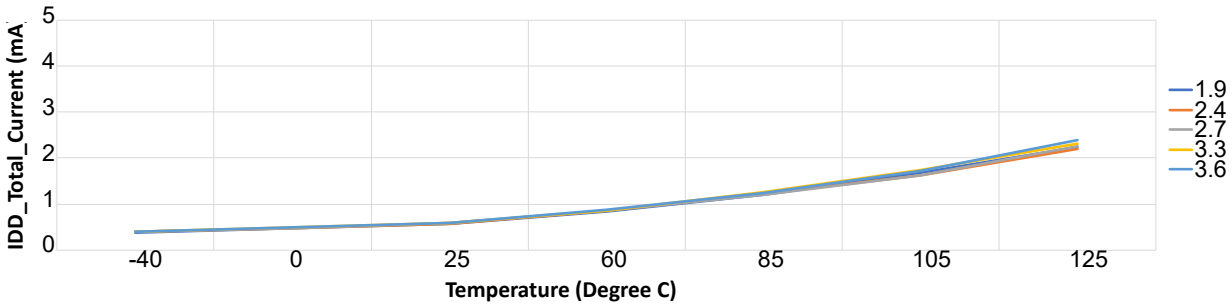


Figure 43-6. Sleep Current with XTAL_OFF_DC_DC

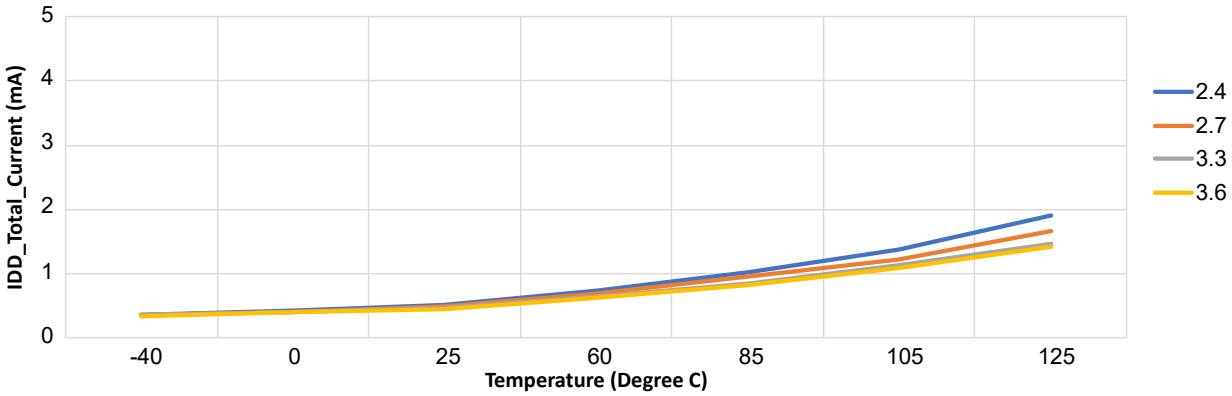


Figure 43-7. Sleep Current with XTAL_ON_MLDO

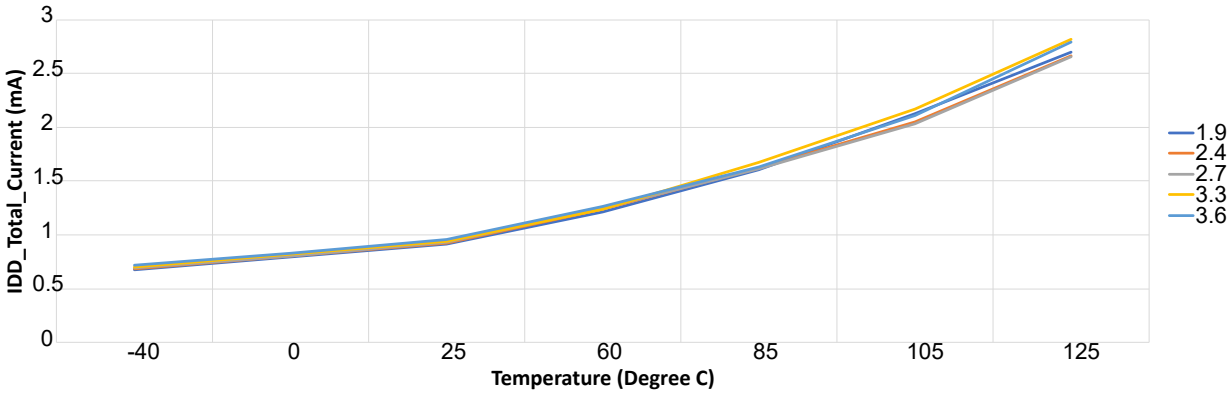
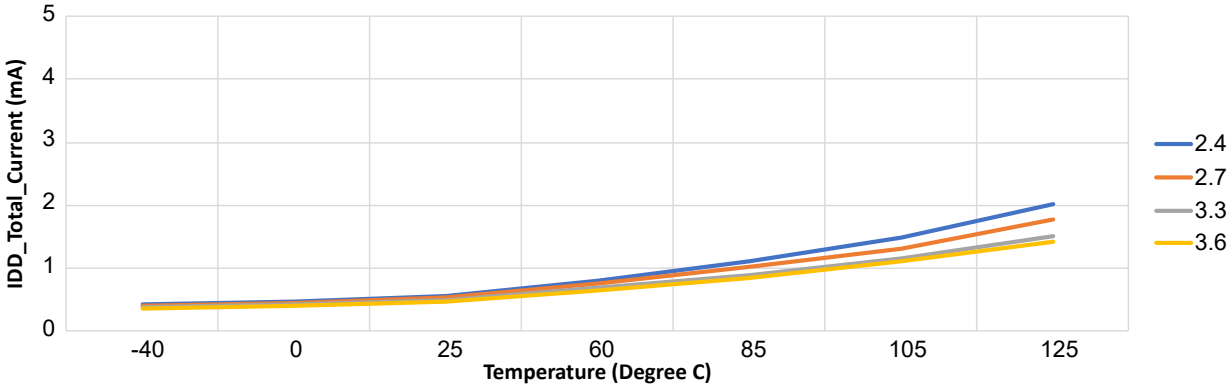


Figure 43-8. Sleep Current with XTAL_ON_DC_DC



Notes:

1. Typical value measured during characterization at 3.3V and 25℃.
2. The test conditions are as follows:
   - All GPIO except RPA3 are pulled up. RPA3 is pulled down.
   - All peripherals are disabled (except EIC).
   - All PB clocks are divided by 16.
   - LPRC is set as LPCLK.
   - SOSC is disabled.
   - CLDO is configured at lowest possible voltage (VREG Trim = 0x07).
   - PMU is configured to the Buck PSM mode on the Sleep Mode Entry with current sense is disabled.
   - Cache is enabled and configured to wait states PFMWS[3:0] as 0xF.
   - Backup RAM in the Retention mode.
   - DSU is disconnected.
   - RF system is in low power configuration.
   - Entry to the Sleep mode is enabled and WFI instruction is executed.
   - In XTAL ON mode - PMU Buck clock is derived from POSC 16 MHz and scaled to 1 MHz. FRC is OFF.
   - In XTAL OFF mode - PMU is clocked from FRC and XTAL 16 MHz clock is disabled and clock configuration in CRU changed to FRC.

43.7 Deep Sleep Current Consumption DC Electrical Specifications

Table 43-7. Deep Sleep Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	Symbol	Characteristics	V_{DDIO}	Typ(1)	Max.	Units	Conditions
BPWR_1	I_{DD_BACKUP}(2)	MCU I_{DD} in Deep Sleep mode powered from V_{DDIO}	3.3V	1.2	—	µA	No backup RAM retained
BPWR_2			3.3V	1.5	—	µA	8 KB backup RAM retained

Figure 43-9. Deep Sleep Current RAM ON

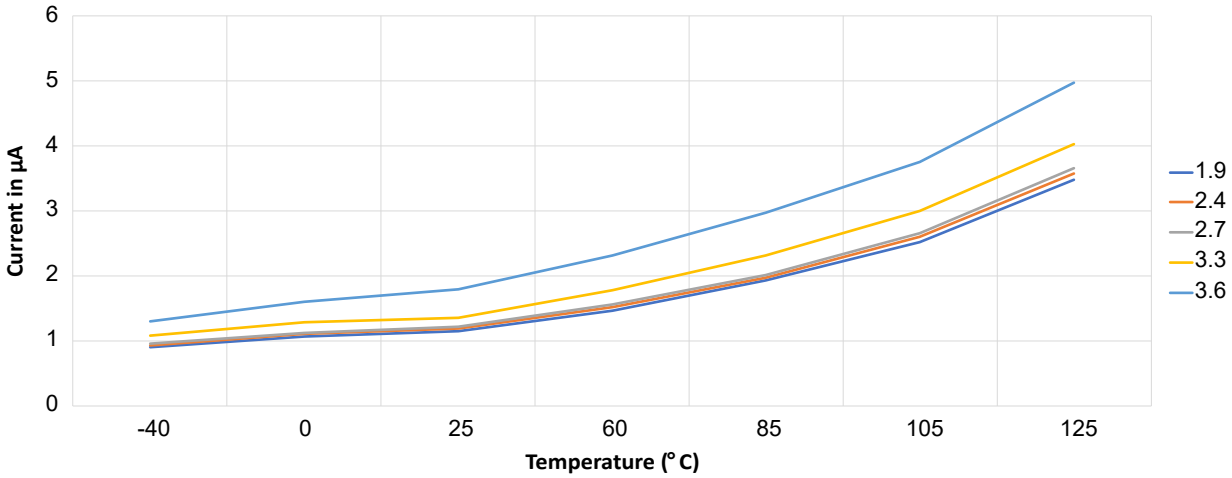
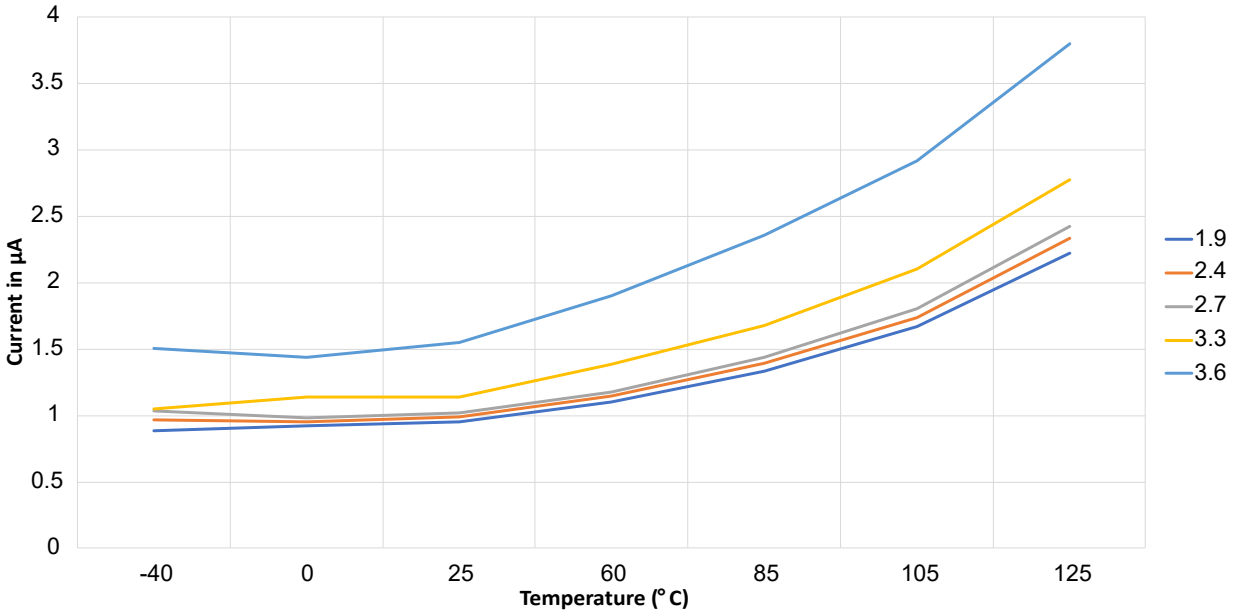


Figure 43-10. Deep Sleep Current RAM OFF



Notes:

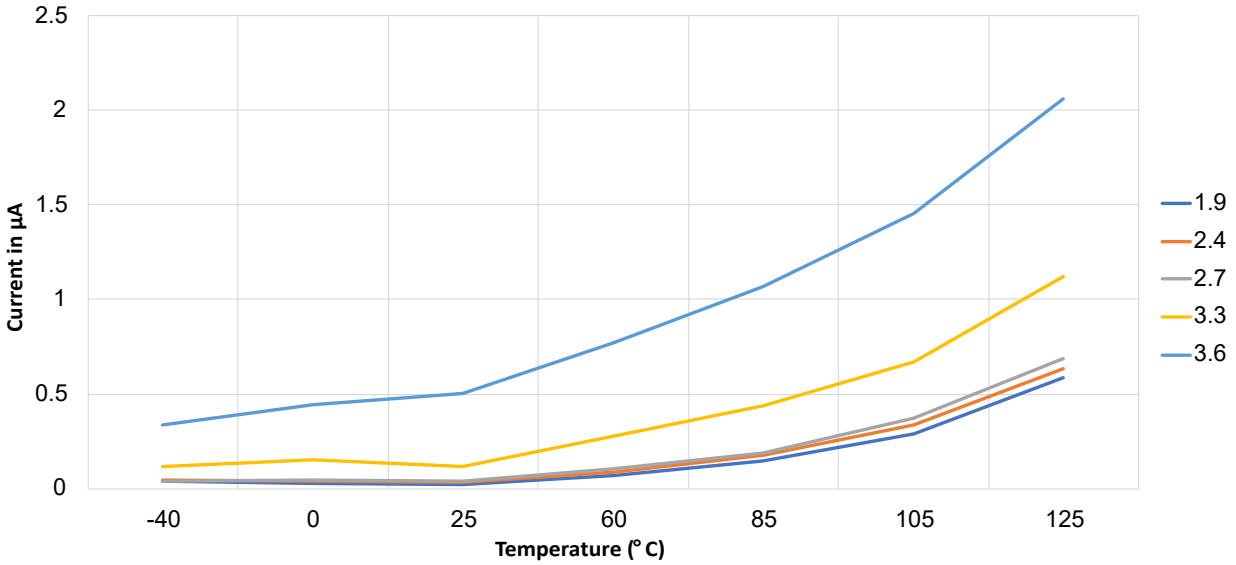
1. Typical value measured during characterization at 3.3V and 25℃.
2. The test conditions are as follows:
   - All GPIO except RPA3 are pulled up. RPA3 is pulled down.
   - All peripherals are disabled.
   - All PB clocks are divided by 16.
   - LPRC is set as LPCLK.
   - SOSC and POSC is disabled.
   - DSU is disconnected.
   - RF system is in low power configuration.
   - DSWDT is enabled and configured for wake-up.
   - Deep sleep entry is configured and WFI instruction is executed.

43.8 XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications

Table 43-8. XDS (Extreme Deep Sleep) Current Consumption DC Electrical Specifications

DC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**V_{DDIO}**	**Typ(1)**	**Max.**	**Units**	**Conditions**
OPWR_1	I_{DD_OFF}(2)	MCU I_{DD} in OFF mode powered from V_{DDIOx}	3.3V	0.12	—	µA	In OFF mode, the device is entirely powered-off. **Note:** This mode is left by pulling the RESET pin low, or when a power Reset is done.

Figure 43-11. Extreme Deep Sleep Current



Notes:

1. Typical value measured during characterization at 3.3V and 25℃.
2. The test conditions are as follows:
   - All GPIO except RPA3 are pulled up. RPA3 is pulled down.
   - All peripherals are disabled.
   - DSU is disconnected.
   - RF system is in low power configuration.
   - DSWDT is disabled.
   - RTCC and POSC is disabled.
   - Deep sleep entry is configured and WFI instruction is executed.

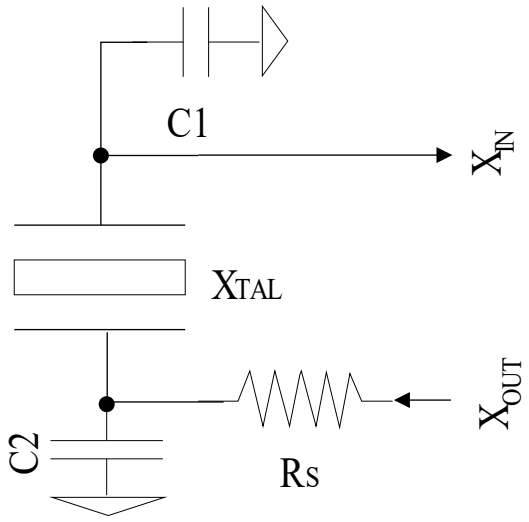
43.9 External XTAL and Clock AC Electrical Specifications

Table 43-9. External XTAL and Clock AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ**	**Max.**	**Units**	**Conditions(1)**
XOSC_1	FOSC_XOSC	XOSC crystal frequency	—	16	—	MHz	—
XOSC_1A	TOSC	TOSC = 1/ FOSC_XOSC	—	0.0625	—	ns	See parameter XOSC1 for FOSC_XOSC value
XOSC_2	XOSC_ST(2)	XOSC crystal start-up time	—	—	2.5	ms	Crystal stabilization time only not oscillator ready

.....continued							
AC Characteristics			**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ**	**Max.**	**Units**	**Conditions(1)**
XOSC_3	CXIN	XOSC XIN parasitic pin capacitance	—	0.35	—	pF	With default crystal trim settings(2)
XOSC_5	CXOUT	XOSC XOUT parasitic pin capacitance	—	0.35	—	pF	With default crystal trim settings
XOSC_11	CLOAD(3)	XOSC crystal FOSC = 16 MHz	—	9	—	pF	—
XOSC_21	ESR	XOSC crystal FOSC = 16 MHz	—	—	100	Ω	—
XOSC_33	DLEVEL	MCU crystal oscillator power drive level	—	—	100	µW	—
XOSC_34	FREQ ERROR	Center frequency error	-30	—	+30	ppm	Across temperature

Figure 43-12. External XTAL and Clock Diagram



Notes:

1. $V_{DDIO} = V_{DDANA} = 3.3V$.
2. Refer RDP for information related to crystal trimming.
3. This is for guidance only. A major component of crystal start-up time is based on the second party crystal MFG parasitic that are outside the scope of this specification.
4. The test conditions for crystal load capacitor calculation are as follows:
   – Standard PCB trace capacitance = 1.5 pF per 12.5 mm (0.5 inches) (in other words, PCB STD TRACE W = 0.175 mm, H = 36 µm, T = 113 µm).
   – Xtal PCB capacitance typical; therefore, ~= 2.5 pF for a tight PCB xtal layout
   – For CXIN and CXOUT within 4 pF of each other, assume CXTAL_EFF = ((CXIN+CXOUT) / 2).
   – **Note:** Averaging CXIN and CXOUT will affect the final calculated CLOAD value by less than 0.25 pF.

Equation 43-1. Equation 1:

$$MFG\ \ CLOAD\ Spec = \{([CXIN + C1] * [CXOUT + C2])/[CXIN + C1 + C2 + CXOUT]\} + estimated\ oscillator\ PCB\ stray\ capacitance$$

Assuming C1 = C2 and CXin ~= CXout, the formula can be further simplified and restated to solve for C1 and C2 by:

Equation 43-2. Equation 2 (In other words: Simplified Equation 1)

$$C1 = C2 = ((2 * MFG\ \ CLOAD\ Spec) - CXTAL\_EFF - (2 * PCB\ \ capacitance))$$

Example:

- XTAL Mfg CLOAD Data Sheet Spec = 12 pF
- PCB XTAL trace Capacitance = 2.5 pF
- CXIN pin = 6.5 pF, CXOUT pin = 4.5 pF; therefore, CXTAL_EFF = ((CXIN+CXOUT) / 2)

CXTAL_EFF = ((6.5 + 4.5)/2) = 5.5 pF

C1 = C2 = ((2 * MFG Cload spec) - CXTAL_EFF - (2 * PCB capacitance))

C1 = C2 = (24 - 5.5 - (2 * 2.5))

C1 = C2 = 13.5 pF (Always rounded down)

C1 = C2 = 13 pF (in other words, for hypothetical example crystal external load capacitors)

User C1 = C2 = 13 pF CLOAD (max) spec

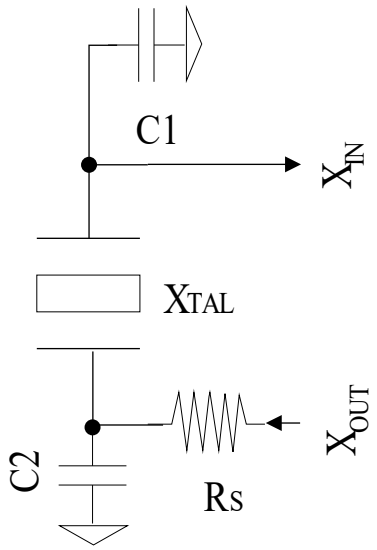
43.10 XOSC32 AC Electrical Specifications

Table 43-10. XOSC32 AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions(1)**
XOSC32_1	FOSC_XOSC32	XOSC32 oscillator crystal frequency	—	32.768	—	kHz	XIN32, XOUT32 secondary oscillator
XOSC32_3	CXIN32	XOSC32 XIN32 parasitic pin capacitance	—	0.4 2.4	—	pF	At the SOC pins on the Module
XOSC32_5	CXOUT32	XOSC32 XOUT32 parasitic pin capacitance on PIC32CX-BZ2	—	0.4 2.4	—	pF	At the SOC pins on the Module

.....continued							
AC Characteristics			**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions(1)**
XOSC32_11	CLOAD_X32(3)	32.768 kHz crystal load capacitance	—	11	—	pF	—
XOSC32_13	ESR_X32	32.768 kHz crystal ESR	—	75	—	KΩ	—
XOSC32_15	TOSC32	TOSC32 = 1/ FOSC_XOSC32	—	30.518	—	µs	See parameter XOSC32_1 for FOSC_XOSC32 value
XOSC32_17	XOSC32_ST(2)	XOSC32 crystal start-up time	—	23	—	ms	This includes system delay and cannot be considered as the exact start-up time of SOSC clock as it is brought out on REFOx

Figure 43-13. XOSC32 Block Diagram



Notes:

1. V_{DDIO} = V_{DDANA} = 3.3V.
2. This is for guidance only. A major component of crystal start-up time is based on the second party crystal MFG parasitic that is outside the scope of this specification. If this is a major concern, the customer might need to characterize this based on their design choices.
3. The test conditions for crystal load capacitor calculation are as follows:
   – Standard PCB trace capacitance = 1.5 pF per 12.5 mm (0.5 inches) (in other words, PCB STD TRACE W = 0.175 mm, H = 36 µm, T = 113 µm)
   – Xtal PCB capacitance typical; therefore, ~= 2.5 pF for a tight PCB xtal layout
   – For CXIN and CXOUT within 4 pF of each other, assume CXTAL_EFF = ((CXIN / 2)
   – **Note:** Averaging CXIN and CXOUT will affect the final calculated CLOAD value by less than the tolerance of the capacitor selection.
4. **Equation 43-3. Equation 1:**

   $$MFG\ \ CLOAD\ Spec = \{([CXIN + C1] * [CXOUT + C2])/[CXIN + C1 + C2 + CXOUT]\} + estimated\ oscillator\ PCB\ stray\ capacitance$$

   Assuming C1 = C2 and CXin ~= CXout, the formula can be further simplified and restated to solve for C1 and C2 by:

   Equation 43-4. Equation 2 (In other words: Simplified Equation 1)

   $$C1 = C2 = ((2 * MFG\ \ CLOAD\ Spec) - CXTAL\_EFF - (2 * PCB\ \ capacitance))$$

   Example:

   - XTAL Mfg CLOAD Data Sheet Spec = 12 pF
   - PCB XTAL trace Capacitance = 2.5 pF
   - CXIN pin = 6.5 pF, CXOUT pin = 4.5 pF therefore CXTAL_EFF = ((CXIN / 2)

   CXTAL_EFF = ((6.5 + 4.5)/2) = 5.5 pF

   C1 = C2 = ((2 * MFG Cload spec) - CXTAL_EFF - (2 * PCB capacitance))

   C1 = C2 = (24 - 5.5 - (2 * 2.5))

   C1 = C2 = (24 - 5.5 - 5)

   C1 = C2 = 13.5 pF (Always rounded down)

   C1 = C2 = 13 pF (in other words, for hypothetical example crystal external load capacitors)

   User C1 = C2 = 13 pF ≤ CLOAD_X32 (max.) spec

43.11 Low Power Internal 32 kHz RC Oscillator AC Electrical Specifications

Table 43-11. Low Power Internal 32 kHz RC Oscillator AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
LP32K_1	FOSC_LPRC32K	Output frequency	—	32.7	—	kHz	Factory default calibration
LP32K_3	LPRC32K_ACC	Accuracy	—	—	—	kHz	—
LP32K_9	RC32K_Duty	LPRC32K OSC duty cycle	—	50	—	%	—

43.12 FRC AC Electrical Specifications

Table 43-12. FRC AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
FRC1	FINTFREQ	Internal FRC frequency	—	8	—	MHz	Factory calibrated and user trim bits set to zero
FRC2	TSUFRC	Start-up time of internal FRC	—	15	—	µs	Not characterized
FRC3	FACCU	FRC accuracy	—	—	—	%	—
FRC5	DUTY_CYCLE	FRC duty cycle	—	50	—	%	—
FRC6	C_USE	FRC user tune frequency drift	—	—	—	%	Frequency drift = [(Maximum frequency measured - Minimum frequency measured)/ (Default frequency of 8 MHz)] * 100; Maximum frequency drift possible by using the frequency trim bits
FRC7	C_USER_STEP_SIZE	FRC user tune step size	—	—	—	%	Step size = (% Drift across Osc tune)/(Total number of trim bit combinations); Change in frequency with incremental trim bit

43.13 Frequency AC Electrical Specifications

Table 43-13. Frequency AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
MCU_1	FCY	MCU frequency of operation	—	—	64	MHz	$V_{DDIO(min)}$ to $V_{DDIO(max)}$
MCU_3	TCY	MCU clock period	1/FCY			µs	

PLL (Phase Locked Loop) AC Electrical Specifications

Table 43-14. PLL MHz (Phase Locked Loop)

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
PLL_1	PLL_FIN	PLL input frequency	—	16	—	MHz	—
PLL_3	PLL_FOUT	PLL output clock frequency	—	96	—	MHz	—
PLL_4	PLL_VCO_FREQ	PLL VCO frequency (Fixed)	—	480	—	MHz	—

Note:

1. PLL input clock is 16 MHz XOSC.

43.14 QSPI Module Electrical Specifications

Table 43-15. QSPI Module Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.(1)**	**Max.**	**Units**	**Conditions**
QSPI_1	FCLK	SQI serial clock frequency • SDR Host mode 0 and 2	—	—	32	MHz	—
QSPI_2	FCLK	SQI serial clock frequency • SDR Host mode 1 and 3	—	—	32	MHz	—
QSPI_3	TSCKH	Serial clock high time	—	8.9	—	ns	—
QSPI_5	TSCKL	Serial clock low time	—	7.8	—	ns	—
QSPI_7	TSCKR	Serial clock rise time	—	6.9	—	ns	—
QSPI_9	TSCKF	Serial clock fall time	—	7.6	—	ns	—
QSPI_11	TCSS	CS active setup time	—	7.3	—	ns	—
QSPI_13	TCSH	CS active hold time	—	10.4	—	ns	—
QSPI_19	TDIS	Data in setup time	—	7.4	—	ns	—

.....continued							
AC Characteristics			**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.(1)**	**Max.**	**Units**	**Conditions**
QSPI_21	TDIH	Data in hold time	—	19.8	—	ns	—
QSPI_23	TDOH	Data out hold	—	18	—	ns	—
QSPI_25	TDOV	Data out valid	—	2.4	—	ns	—

Note:

1. V_{DDIO} = 3.3V, C_{LOAD} = 30 pF (Typ.).

43.15 Power Supply DC Module Electrical Specifications

Table 43-16. Power Supply DC Electrical Specifications

AC Characteristics			**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
REG_1	VDDCORE_CIN	VDDCORE (CLDO_OUT) input bypass parallel capacitor pair(5)	—	1	—	µF	Bulk ceramic or solid tantalum with ESR <0.5Ω. Min and max represent absolute values including cap tolerances
			—	100	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature
REG_5	VDD33	VDD33 input bypass parallel capacitor pair(5)	—	10	—	µF	Bulk ceramic or solid tantalum with ESR <0.5Ω(6)
			—	100	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature on all VDDIO pins(6)
REG_6	PMU_VDDIO	Input bypass parallel capacitor pair for the PMU power section(5)	—	4.7	—	µF	Bulk Ceramic or solid Tantalum with ESR <0.5Ω(6)
			—	—	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature on all VDDIO pins(6)
REG_7	PMU_VDDP	Input bypass parallel capacitor pair for the PMU power section(5)	—	1	—	µF	Bulk ceramic or solid tantalum with ESR <0.5Ω(6)
			—	—	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature on all VDDIN pins(6)

.....continued

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
REG_9	VDDFLASH_C_{IN}	VDD_FLASH bypass parallel capacitor pair(5)	—	10	—	µF	Bulk ceramic or solid tantalum with ESR <0.5Ω(6)
			—	100	—	nF	Ceramic XR7/X5R with ESR <0.5Ω depending on temperature on all VDDFLASH pins(6)
REG_17	VDDANA_C_{IN}	VDDANA input bypass parallel capacitor pair(5)	—	10	—	µF	Bulk ceramic or solid tantalum with ESR <0.5Ω(6)
			—	0.1	—	nF	Ceramic XR7/X5R with ESR <0.5Ω
REG_18	VDDANA_LEXT	VDDANA series ferrite bead DCR (DC resistance)	—	—	0.1	Ω	≥600Ω at 100 MHz
REG_19		Ferrite bead current Rating	100	—	—	mA	—
REG_20	BUCK_PLL_CIN	VDD bypass capacitor on the BUCK_PLL input	—	1	—	µF	Ceramic XR7/X5R with ESR <0.5Ω
REG_21	BUCK_BB_CIN	VDD bypass capacitor on the BUCK_BB input	—	1	—	µF	Ceramic XR7 with ESR <0.5Ω
REG_22	BUCK_MPA_CIN	VDD bypass capacitor on the BUCK_LPA input	—	1	—	µF	Ceramic XR7 with ESR <0.5Ω
REG_23	BUCK_LPA_CIN	VDD bypass capacitor on the BUCK_MPA input	—	1	—	µF	Ceramic XR7 with ESR <0.5Ω
REG_24	BUCK_CLDO_CIN	VDD bypass capacitor on the BUCK_CLDO input	—	1	—	µF	Ceramic XR7 with ESR <0.5Ω
REG_25	R_EXT	Bias for the reference current generation	—	30	—	kΩ	—

.....continued

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
REG_27	VSW_L_{EXT}(2,3)	Buck Switch mode regulator inductor inductance	—	4.7	—	µH	Shielded inductor only
REG_29		Inductor DCR (DC resistance)	—	—	0.22	Ω	(7)
REG_31		Inductor I_{SAT} Rating(2)	250	—	—	mA	—
REG_32	VSW_CAP EXT	Buck Switch mode regulator bulk capacitor capacitance	10	—	—	µF	—
REG_32A		Buck Switch mode regulator filtering capacitor capacitance	100	—	—	nF	—
REG_36	VDDCORE	VDDCORE voltage range	—	1.2	—	V	MCU Active, cache and prefetch disabled while executing from Flash(1)
REG_37	VDD33(4)	VDD33 input voltage range	1.9	3.3	3.6	V	—
REG_39	VDDANA(4)	VDDANA input voltage range	1.9	3.3	3.6	V	—
REG_40	VDDREG	VDDREG input voltage range	—	1.35	—	V	PMU output voltage
REG_43	SVDDIO_R	VDDIO rise ramp rate to ensure internal Power-on Reset signal	—	0.1	—	V/µs	Failure to meet this specification may lead to start-up or unexpected behaviors
REG_44	SVDDIO_F	VDDIO falling ramp rate to ensure internal Power-on Reset signal	—	—	—	V/µs	Failure to meet this specification may cause the device to not detect reset
REG_45	VP0R+	Power-on Reset	—	1.5	—	V	VDDIO power up/Down (See Param REG43, VDDIO Ramp Rate)
REG_45_A	VP0R-	Power-on Reset	—	1.5	—	V	VDDIO Power up/Down (See Param REG43, VDDIO Ramp Rate)
REG_47	VBOR33(4)	VDDIO BOD (All modes)(4)	—	1.7	—	V	—

.....continued

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
REG_48	VBOR12	BOR of the 1.2V regulator	—	1.0	—	V	—
REG_48A	VZBPOR33	Zero power BOR	—	1.8	—	V	—
REG_53	TRST(6)	External RESET valid active pulse width	—	11	—	µs	Minimum Reset active time to guarantee MCU Reset for the module. Reset filter circuit inside Module
			—	2.5	—	µs	Minimum Reset active time to guarantee MCU Reset for SoC with no Reset filter circuit
REG_54	MLDO_DELAY	PMU MLDO and CLDO power-up delay	—	260	—	µs	—
REG_55	SYS_DELAY	System delay before fetching first instruction	—	—	—	µs	—
REG_56	CLK_DELAY	Crystal start-up delay	—	XOSC_2	—	µs	Refer to XOSC_2 parameter from the crystal oscillator section
REG_57	POWER_ON_DELAY	Power-up period	—	—	—	µs	—
REG_58	BOR_DELAY	Width of the BOR event	—	488	—	µs	Includes system delay since this is measured with a CPU event

Figure 43-14. Power-On Reset Timing Characteristics

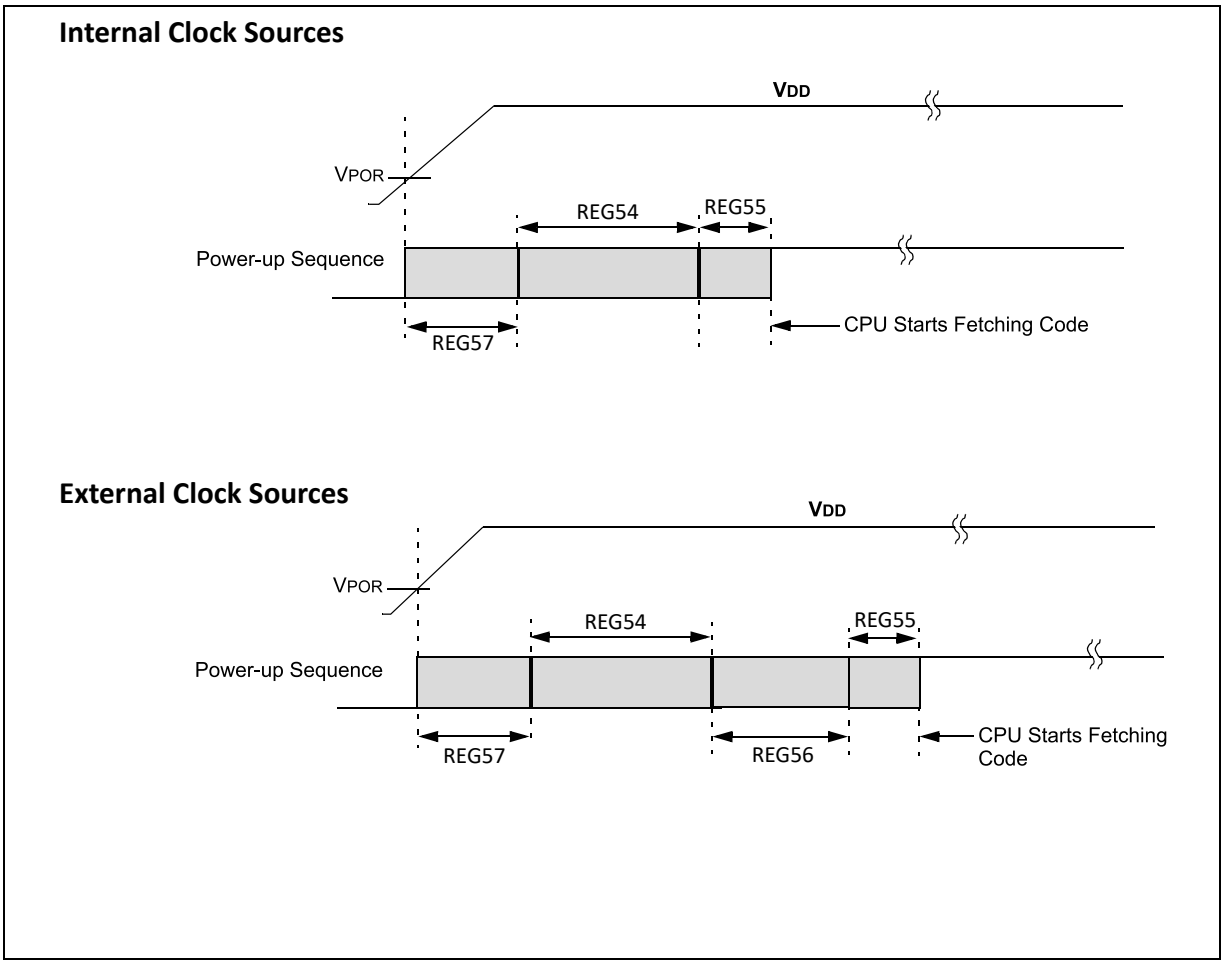
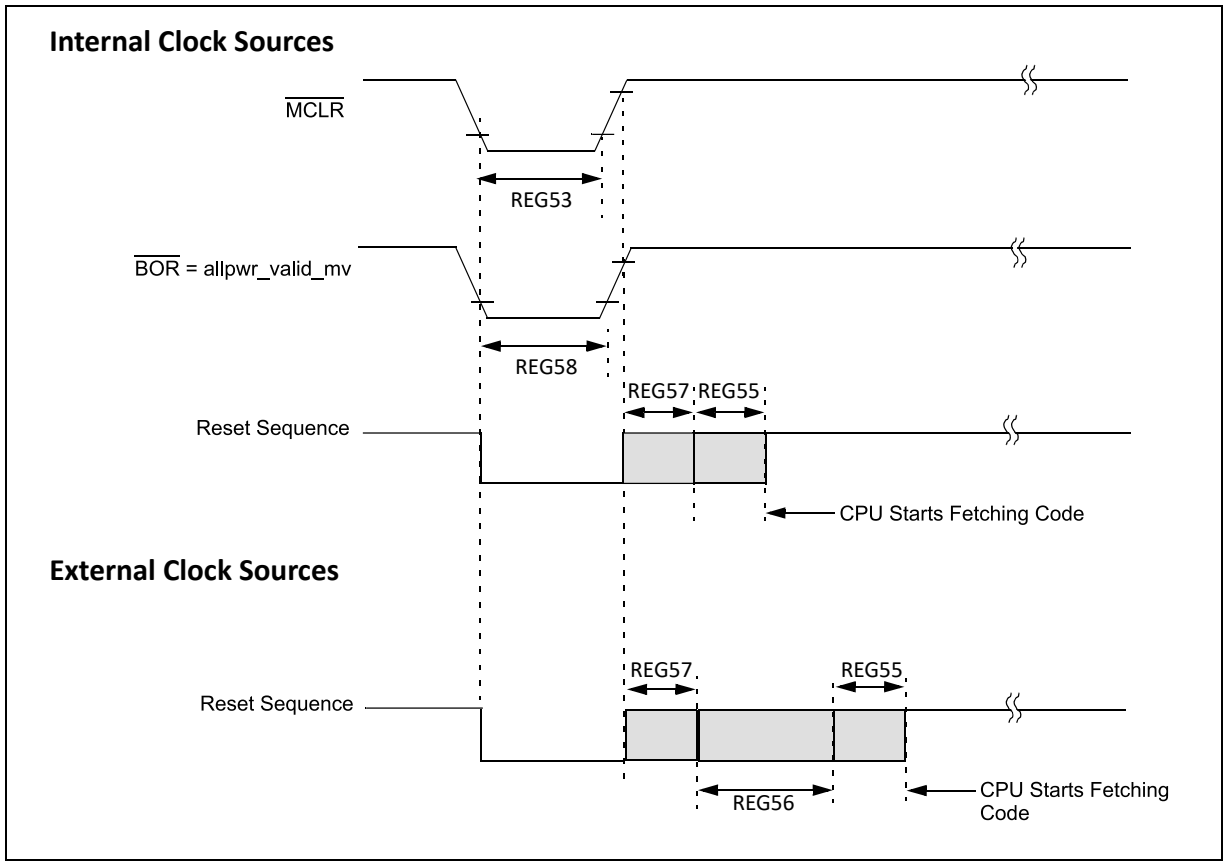


Figure 43-15. External Reset Timing Characteristics



Notes:

1. Ferrite Bead ISAT(min) ≥ (IDDANA(max) * 1.15).
2. Buck Inductor ISAT(min) ≥ ((ICAPACITOR + IVDDCORE_MAX) * 1.2) when the BUCK mode is enabled (shielded inductor only).
3. User must select either LDO or BUCK Mode. The modes are exclusive to each other.
4. VDD33 and VDDANA must be at the same voltage level.
5. All bypass caps must be located immediately adjacent to pin(s) and on the same side of the PCB as the MCU. Each primary power supply group VDDIO, VDDANA, VDDCORE must have one bulk capacitor and all power pins with a 100 nF bypass cap.
6. The RESET pulse width is the minimum pulse width required on the I/O pin after any filtering on the NMCLR pin.
7. Keep the DCR as low as possible to improve efficiency.

43.16 I/O PIN AC/DC Electrical Specifications

Table 43-17. I/O PIN AC/DC Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.(1)**	**Max.**	**Units**	**Conditions**
DI_1	VOL	Output voltage low (Drive strength, 8x)	—	0.1	—	V	VDDIO = 3.3V at IOL= 10 mA
		Output voltage low (Drive strength, 4x)		0.2	—		VDDIO = 3.3V at IOL= 10 mA
DI_2	VOL	Output voltage low (Drive strength, 8x)	—	0.2	—	V	VDDIO = 3.3V at IOL= 13 mA
		Output voltage low (Drive strength, 4x)		0.3	—		VDDIO = 3.3V at IOL= 13 mA
DI_3	VOL	Output voltage low (Drive strength, 8x)	—	0.2	—	V	VDDIO = 3.3V at IOL= 15 mA
DI_4	VOH	Output voltage low (Drive strength, 8x)	—	2.3	—	V	VDDIO = 3.3V at IOH= 5 mA
		Output voltage low (Drive strength, 4x)	—	2.1	—		VDDIO = 3.3V at IOH= 5 mA
DI_5	VOH	Output voltage low (Drive strength, 8x)	—	2.2	—	V	VDDIO = 3.3V at IOH= 7 mA
		Output voltage low (Drive strength, 4x)	—	2.0	—		VDDIO = 3.3V at IOH= 7 mA
DI_6	VOH	Output voltage High (Drive strength, 8x)	—	2.1	—	V	VDDIO = 3.3V at IOH= 10 mA
DI_7	VIL	Input voltage low (Drive strength, 8x)	—	0.7	—	V	VDDIO = 3.3V
		Input voltage low (Drive strength, 4x)	—	0.5	—		VDDIO = 3.3V

.....continued

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.(1)**	**Max.**	**Units**	**Conditions**
DI_8	VIH	Input voltage high (Drive strength, 8x)	—	2.9	—	V	VDDIO = 3.3V
		Input voltage high (Drive strength, 4x)	—	2.9	—		VDDIO = 3.3V
DI_13	IIL	Input pin leakage current	—	71.8	—	nA	—
DI_15(5)	RPDWN	Internal pull-down (Drive strength, 8x)	—	21.4	—	kΩ	VDDIO = 3.3V
		Internal pull-down (Drive strength, 4x)	—	3.9	—	kΩ	
DI_17	RPUP	Internal pull-up (Drive strength, 8x)	—	18.6	—	kΩ	
		Internal pull-up (Drive strength, 4x)	—	3.3	—	kΩ	
DI_25	TRISE	I/O pin rise time (High drive strength, high load)	—	1.97	—	ns	VDDIO = 3.3V
		I/O pin rise time (Low drive strength, high load)	—	10.7	—	ns	—
		I/O pin rise time (High drive strength, standard load)	—	2.0	—	ns	—
		I/O pin rise time (Low drive strength, standard load)	—	7.6	—	ns	—

.....continued							
AC Characteristics			**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.(1)**	**Max.**	**Units**	**Conditions**
DI_27	TFALL	I/O pin fall time (High drive strength, high load)	—	1.6	—	ns	—
		I/O pin fall time (Low drive strength, high load)	—	8.1	—	ns	—
		I/O pin fall time (High drive strength, standard load)	—	1.6	—	ns	—
		I/O pin fall time (Low drive strength, standard load)	—	5.1	—	ns	—

Notes:

1. All measurements done at 3.3V at 25℃.
2. VIL source < (GND – 0.3). Characterized but not tested.
3. Any number and/or combination of I/O pins not excluded under IICL or IICH conditions are permitted provided the absolute instantaneous sum of the input injection currents from all pins do not exceed the specified ∑IICT limit. To limit the injection current, the user must insert a resistor in series RSERIES (in other words, RS), between the input source voltage and device pin. The resistor value is calculated according to:
   - For negative Input voltages less than (GND – 0.3): RS ≥ absolute value of | ((VIL source – (GND – 0.3)) / IICL) |
   - For positive input voltages greater than (VDDIO + 0.3): RS ≥ ((VIH source – (VDDIO +0.3))/ IICH)
   - For Vpin voltages greater than VDDIO + 0.3 and less than GND – 0.3: RS = the larger of the values calculated above
4. High load = 50 pF and Standard load = 30 pF.
5. The drive strength of pads are as follows:
   - 8x pads - PA0, PA1, PA3, PA4, PA5, PA6, PA7, PA8, PA9, PA10, PA13, PA14, PB10, PB11, PB12, PB13
   - 4x pads - PA2, PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7, PB8, PB9

43.17 I²C Module Electrical Specifications

Note: Traditional Serial Communication Interface documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

Table 43-18. I²C Module Host Mode Electrical Specifications

AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial				
Param. No.	**Symbol**	**Characteristics**		**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
I2CM_1	TL0:SCL	Host clock low time	100 kHz mode	—	5020	—	ns	VDDIOx = 3.3V,
			400 kHz mode	—	1280	—	ns	
			1 MHz mode	—	712	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pf
I2CM_5	TF:SCL	SDAx and SCLx fall time	100 kHz mode	—	19	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	5	—	ns	
			1 MHz mode	—	5	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CM_7	TR:SCL	SDAx and SCLx rise time	100 kHz mode	—	166	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	154	—	ns	
			1 MHz mode	—	154	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CM_9	TSU:DAT	Data input setup time	100 kHz mode	—	4900	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	1200	—	ns	
			1 MHz mode	—	592	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CM_11	THD:DAT	Data input hold time	100 kHz mode	—	159	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	93	—	ns	
			1 MHz mode	—	140	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF

.....continued

AC Characteristics				Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial				
Param. No.	**Symbol**	**Characteristics**		**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
I2CM_13	TSU:STA	Start condition setup time	100 kHz mode	—	5180	—	µs	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	1450	—	ns	
			1 MHz mode	—	867	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CM_15	THD:STA	Start condition hold time	100 kHz mode	—	4990	—	µs	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	1250	—	µs	
			1 MHz mode	—	666	—	µs	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CM_21	TAA:SCL	Output valid from clock	100 kHz mode	—	—	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	—	—	ns	
			1 MHz mode	—	—	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CM_23	TBF:SDA	Bus free time(1)	100 kHz mode	—	TL0:SCL	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	TL0:SCL	—	ns	
			1 MHz mode	—	TL0:SCL	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF

Note:

1. The amount of time the bus must be free before a new transmission can start (STOP condition to START condition).

Figure 43-16. I²C Start/Stop Bits Host Mode AC Timing Diagram

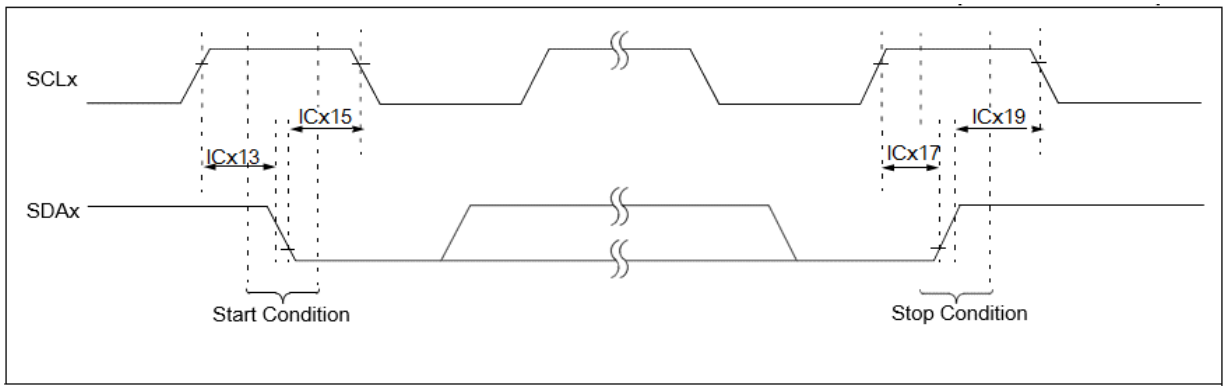


Figure 43-17. I²C Bus Data Host Mode AC Timing Diagram

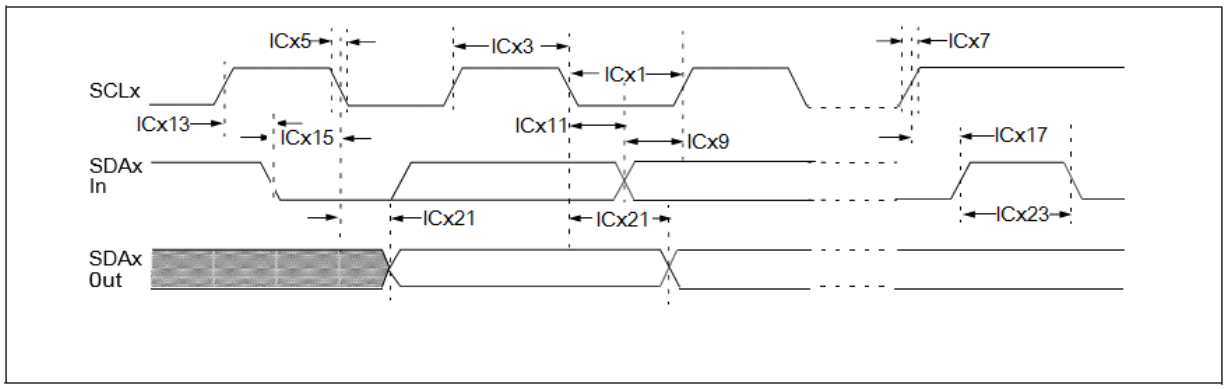


Table 43-19. I²C Module Client Mode Electrical Specifications

AC Characteristics				Standard Operating Conditions: $V_{DDIO} = V_{DDANA}$ 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended Temp				
Param. No.	**Symbol**	**Characteristics**		**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
I2CS_5	TF:SCL	SDAx and SCLx fall time	100 kHz mode	—	19	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	5	—	ns	
			1 MHz mode	—	5	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CS_7	TR:SCL	SDAx and SCLx rise time	100 kHz mode	—	166	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	154	—	ns	
			1 MHz mode	—	154	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF

.....continued								
AC Characteristics				**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended Temp**				
Param. No.	**Symbol**	**Characteristics**		**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
I2CS_9	TSU:DAT	Data input setup time	100 kHz mode	—	5580	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	1470	—	ns	
			1 MHz mode	—	—	—	ns	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF
I2CS_11	THD:DAT	Data input hold time	100 kHz mode	—	490	—	ns	VDDIOx = 3.3V, IPULL-UP = 3 mA, CLOAD = 400 pF
			400 kHz mode	—	490	—	ns	
			1 MHz mode	—	—	—	µs	VDDIOx = 3.3V, IPULL-UP = 20 mA, CLOAD = 550 pF

Note:

1. The amount of time the bus must be free before a new transmission can start (STOP condition to START condition).

Figure 43-18. I^2C Start/Stop Bits Client Mode AC Timing Diagram

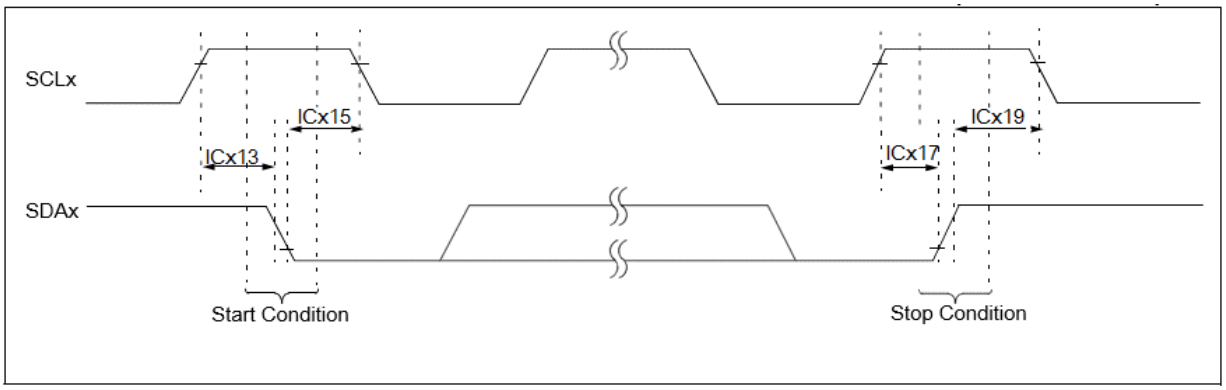
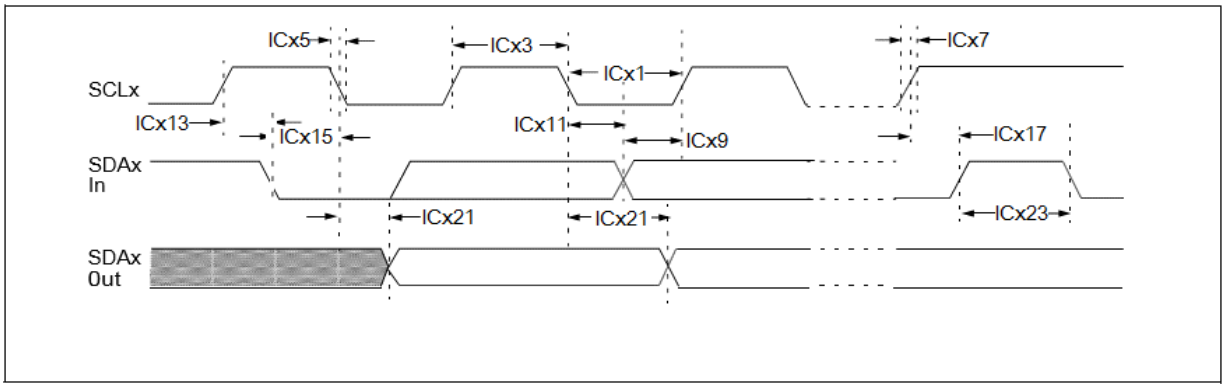


Figure 43-19. I²C Bus Data Client Mode AC Timing Diagram



43.18 SPI Module Electrical Specifications

Note: Traditional Serial Communication Interface documentation uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

Table 43-20. SPI Module Host Mode Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
MSP_1	FSCK	SCK frequency	—	32	—	MHz	Fixed pins
			—	24	—		Remappable pins
MSP_3	TSCL	SCK output low time	—	31.2	—	ns	—
MSP_5	TSCH	SCK output high time	—	31.2	—	ns	—
MSP_7	TSCF	SCK and MOSI output fall time	—	1.8	—	ns	See parameter DI27 I/O spec
MSP_9	TSCR	SCK and MOSI output rise time	—	1.8	—	ns	See parameter DI25 I/O spec
MSP_11	TMOV	MOSI Data output valid after SCK	—	11	—	ns	VDDIOx (Min.), CLOAD = 30 pF (Typ.)
MSP_13	TMOH	MOSI hold after SCK	—	15	—	ns	
MSP_15	TMIS	MISO setup time of data input to SCK	—	23	—	ns	
MSP_17	TMIH	MISO hold time of data input to SCK	—	8	—	ns	

Figure 43-20. SPI Host Module CPHA=0 Timing Diagrams

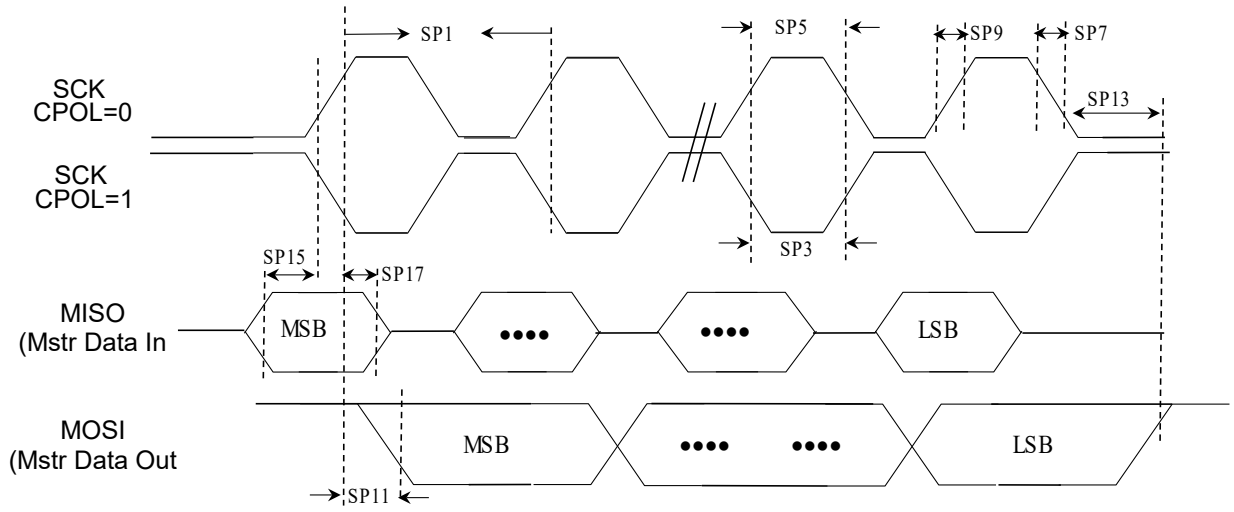
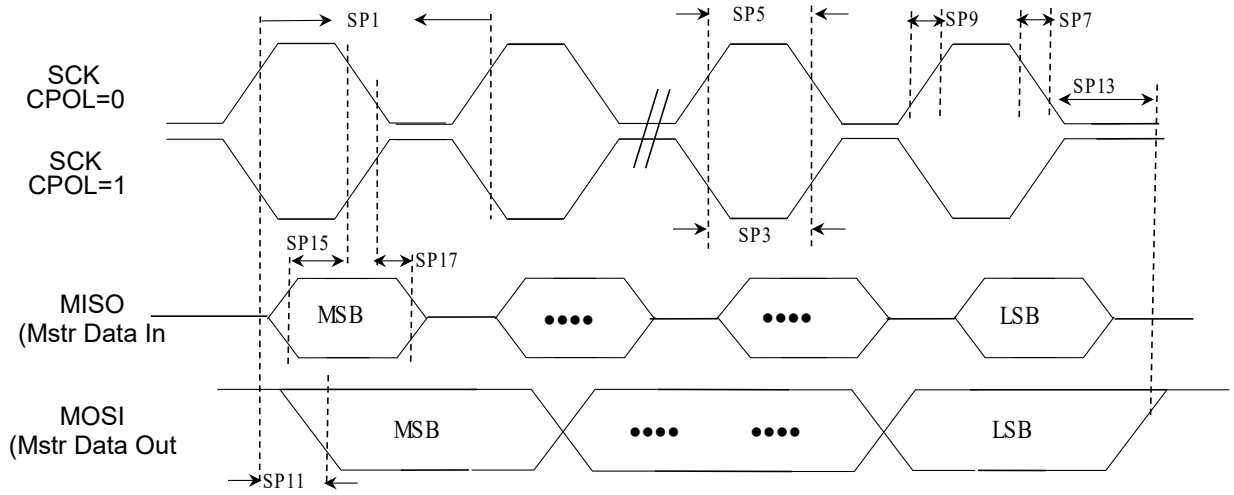


Figure 43-21. SPI Host Module CPHA=1 Timing Diagrams



Note:

1. Assumes VDDIOx(min) and 30 pF external load on all SPIx pins unless otherwise noted.

Table 43-21. SPI Module Client Mode Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ**	**Max.**	**Units**	**Conditions**
SSP_1	FSCK	SCK frequency	—	16	16	MHz	VDDIOx = 1.9V or VDDIO(min) whichever is greater, CLOAD = 30 pF (Min) fixed pins
			—	12	12		Remappable pins
SSP_3	TSCL	SCK output low time	—	62.5	—	ns	—

.....continued							
AC Characteristics			**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ**	**Max.**	**Units**	**Conditions**
SSP_5	TSCH	SCK output high time	—	62.5	—	ns	—
SSP_7	TSCF	SCK and MOSI output fall time	—	3	—	ns	See parameter DI27 I/O spec
SSP_9	TSCR	SCK and MOSI output rise time	—	26	—	ns	See parameter DI25 I/O spec
SSP_11	TSOV	MOSI data output valid after SCK	—	21	—	ns	VDDIOx = 3.3V, CLOAD = 30 pF (Min)
SSP_15	TSIS	MISO setup time of data input to SCK	—	23	—	ns	
SSP_17	TSIH	MISO hold time of data input to SCK	—	22	—	ns	

Figure 43-22. SPI Client Module CPHA=0 Timing Diagrams

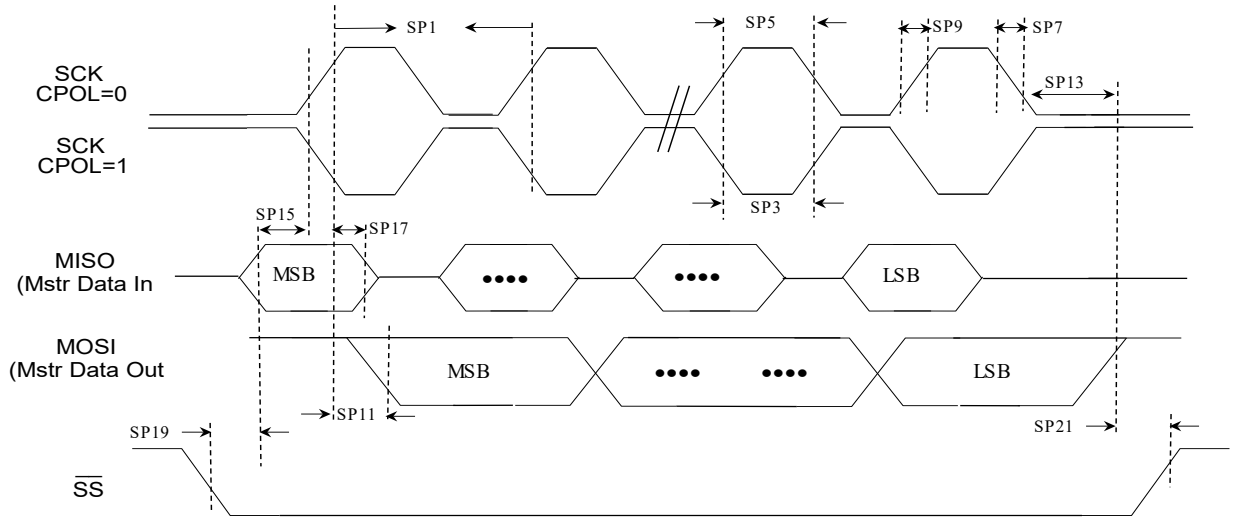
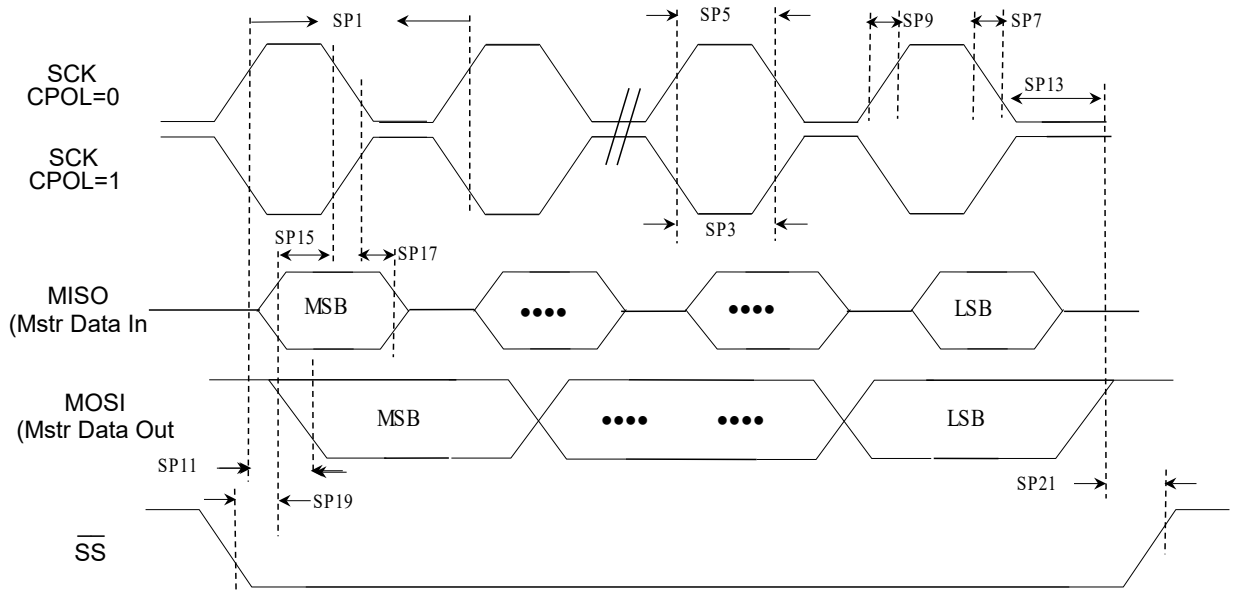


Figure 43-23. SPI Client Module CPHA=1 Timing Diagrams



43.19 ADC Electrical Specifications

Table 43-22. ADC AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
Device Supply							
ADC_1	V_{DDANA}	ADC supply	$V_{DDANA(min)}$	—	$V_{DDANA(max)}$	V	V_{DDIO} = V_{DDANA}
Reference Inputs							
ADC_3	V_{REF}(6)	ADC reference voltage(4)	—	—	V_{DDANA}	V	External reference voltage
Analog Input Range							
ADC_7	AFS	Full-scale analog input signal range (Single-ended)	0	—	V_{DDANA}	V	V_{REF} = $V_{DDANA(max)}$
ADC_9		Full-scale analog input signal range (Differential)	V_{DDANA}/2	—	V_{DDANA}/2	V	

Table 43-23. ADC Single-Ended Mode AC Electrical Specifications

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
Single-Ended Mode ADC Accuracy							
SADC_11	Res	Resolution	6	—	12	bits	Selectable 6, 8, 10, 12 bit resolution ranges
SADC_13	EN0B(3)	Effective number of bits	—	9.3	—	bits	1.6 Msps, Internal V_{REF}, V_{DDANA} = V_{DDIO} = 3.3V
SADC_19	INL(3)	Integral non linearity	—	-5.2 2.7	—	LSb	1.6 Msps, Internal V_{REF}, V_{DDANA} = V_{DDIO} = 3.3V
SADC_25	DNL(3)	Differential non linearity	—	-1 1.8	—	LSb	1.6 Msps, Internal V_{REF}, V_{DDANA} = V_{DDIO} = 3.3V
SADC_31	GERR(3)	Gain error	—	-0.7	—	LSb	1.6 Msps, Internal V_{REF}, V_{DDANA} = V_{DDIO} = 3.3V
SADC_37	E0FF(3)	Offset error	—	3.2	—	LSb	Internal V_{REF}, V_{DDANA} = V_{DDIO} = 3.3V
Single-Ended Mode ADC Dynamic Performance							
SADC_49	SINAD(1,2,3)	Signal to noise and distortion	—	57.8	—	dB	V_{REF} = V_{DDANA} = V_{DDIO} = 3.3V at 12-bit resolution, max sampling rate(1,2)
SADC_51	SNR(1,2,3))	Signal to noise ratio	—	58.2	—		
SADC_53	SFDR(1,2,3)	Spurious free dynamic range	—	66.2	—		
SADC_55	THD(1,2,3,5)	Total harmonic distortion	—	-71.3	—		

Notes:

1. Characterized with an analog input sine wave = (FTP(maximum)/100). Example: FTP(maxium) = 1 Msps/100 = 10 KHz sine wave.
2. Sinewave peak amplitude = 96% ADC_ Full Scale amplitude input with 12-bit resolution.
3. Spec values collected under the following additional conditions:
   a. 12-bit resolution mode.
   b. All registers at reset default value otherwise not mentioned.
4. ADC Measurements done with 3.3V V_{REF} Voltage.
5. Value taken over 7 harmonics.
6. Referred to as AVDD in the pinout.

Table 43-24. ADC Conversion AC Electrical Requirements

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
ADC_ Clock Requirements							
ADC_57	TAD	ADC clock period	—	20.8	—	ns	V_{REF} = V_{DDANA} = 3.3V
ADC Single-Ended Throughput Rates							
ADC_59	FTPR (Single-ended Mode)	Throughput rate(2) (Single-ended)	—	2	—	Msps	12-bit resolution, DIV_SHR = 2
			—	0.7	—		12-bit resolution, DIV_SHR = 4

Notes:

1. Conversion_time = (SAMC_SHR+15)*TAD.
2. FTPR = 1/((SAMC_SHR + Resolution + 1) * (1/(ControlClk/DIV_SHR))).

Table 43-25. ADC Sample AC Electrical Requirements

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
ADC_65	TCNV	Conversion time(1) (Single-ended Mode)		14		TAD	12-bit resolution
				12			10-bit resolution
				10			8-bit resolution
ADC_67		Conversion time(1) (Differential Mode)		14		TAD	12-bit resolution
				12			10-bit resolution
				10			8-bit resolution
ADC_69	CSAMPLE	ADC internal sample cap	—	5	—	pf	—
ADC_71	RSAMPLE	ADC internal impedance	—	—	200	Ω	—

Note:

1. ADC Throughput Rate FTP = ((1/((TSAMP + TCNV) * TAD))/(# of user active analog inputs in use on specific target ADC module)).

43.20 Bluetooth Low Energy RF Characteristics

Table 43-26. Bluetooth Low Energy RF Characteristics

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ(1)**	**Max.**	**Units**	**Conditions**
BTG1	FREQ	Frequency range of operation	2400	—	2480	MHz	—
BTTX1	TXPWR:MPA	Bluetooth transmit power MPA	—	11.5	—	dBm	—
BTTX2	TXPWR:LPA	Bluetooth transmit power LPA	—	4.0	—	dBm	—
BTX3	TXIB:1MBPS	In-band emission for FTX ± -2 MHz	—	-32	—	dBm	—
		In-band emission for FTX ± -(3+N) MHz	—	-45	—	dBm	—
BTX4	TXIB:2MBPS	In-band emission for FTX ± -4 MHz	—	-43	—	dBm	—
		In-band emission for FTX ± -5 MHz	—	-48	—	dBm	—
		In-band emission for FTX ± -(6+N) MHz	—	-51	—	dBm	—
BTRX1	RXSENSE	Receiver sensitivity at 1 Mbps	—	-95.5	—	dBm	(1)
		Receiver sensitivity at 2 Mbps	—	-92.5	—	dBm	(5)
		Receiver sensitivity at 500 kbps	—	-98.5	—	dBm	(5)
		Receiver sensitivity at 125 kbps	—	-102	—	dBm	(5)
BTRX2	MAXINSIG	Maximum input signal level at 1 Mbps	—	0	—	dBm	—
		Maximum input signal level at 2 Mbps	—	0	—	dBm	
		Maximum input signal level at 500 kbps	—	0	—	dBm	
		Maximum input signal level at 125 kbps	—	0	—	dBm	

.....continued

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ(1)**	**Max.**	**Units**	**Conditions**
BTRX3(4)	CI1M:COCH	C/I Co channel rejection	—	13	—	dB	—
	CI1M: ± -1 MHz	C/I adjacent channel rejection	—	13	—	dB	—
	CI1M: ± -2 MHz	C/I adjacent channel rejection	—	13	—	dB	—
	CI1M:ADJ(3+n)	C/I alternate channel rejection	—	15	—	dB	—
	CI1M:IMG	C/I image frequency rejection	—	15	—	dB	—
	CI1M:IMG ± -1 MHz	C/I adjacent channel to image freq rejection	—	14	—	dB	—
BTRX4(4)	CIS2:COCH	C/I Co channel rejection	—	11	—	dB	—
	CIS2: ± -1 MHz	C/I adjacent channel rejection	—	17	—	dB	—
	CIS2: ± -2 MHz	C/I adjacent channel rejection	—	18	—	dB	—
	CIS2:ADJ(3+n)	C/I alternate channel rejection	—	17	—	dB	—
	CIS2:IMG	C/I image frequency rejection	—	14	—	dB	—
	CIS2:IMG ± -1 MHz	C/I adjacent channel to image freq rejection	—	18	—	dB	—
BTRX5(4)	CIS8:COCH	C/I Co channel rejection	—	6	—	dB	—
	CIS8: ± -1 MHz	C/I adjacent channel rejection	—	13	—	dB	—
	CIS8: ± -2 MHz	C/I adjacent channel rejection	—	13	—	dB	—
	CIS8:ADJ(3+n)	C/I alternate channel rejection	—	14	—	dB	—
	CIS8:IMG	C/I image frequency rejection	—	8	—	dB	—
	CIS2:IMG ± -1 MHz	C/I adjacent channel to image freq rejection	—	16	—	dB	—

.....continued

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ(1)**	**Max.**	**Units**	**Conditions**
BTRX6(4)	CI2M:COCH	C/I Co channel rejection	—	13	—	dB	—
	CI2M: ± -2 MHz	C/I adjacent channel rejection	—	16	—	dB	—
	CI2M: ± -4 MHz	C/I adjacent channel rejection	—	19	—	dB	—
	CI2M:ADJ(6+2n)	C/I alternate channel rejection	—	16	—	dB	—
	CI2M:IMG	C/I image frequency rejection	—	13	—	dB	—
	CI2M:IMG ± -2 MHz	C/I adjacent channel to image freq rejection	—	19	—	dB	—
BTRX7(4)	BLOCK1M:<2 GHZ	Blocking performance from 30-2 GHz	—	20	—	dB	—
	BLOCK1M:2 GHZ<SIG<2399 MHz	Blocking performance from 2003-2399 MHz	—	14	—	dB	—
	BLOCK1M:2484 MHZ<SIG<2977 MHz	Blocking performance between 2484-2997 MHz	—	20	—	dB	—
	BLOCK1M:3 GHZ<SIG<12.75 GHz	Blocking performance between 3-12.5 GHz	—	20	—	dB	—
BTRX8(4)	BLE1M:INTERMOD	Inter modulation performance for BLEM	—	13.5	—	dB	—
	BLE2M:INTERMOD	Inter modulation performance for BLEM	—	19.5	—	dB	—

Notes:

1. Measured at 25℃, averaged across all voltages and channels.
2. Measured on a board with the reference schematic.
3. All measurement across voltage based on the SIG specifications.
4. The specified value is the limit above the SIG specifications.
5. PDU length = 37, channels = 2402/2426/2440/2480 MHz.

Table 43-27. Bluetooth Low Energy RF Current Characteristics

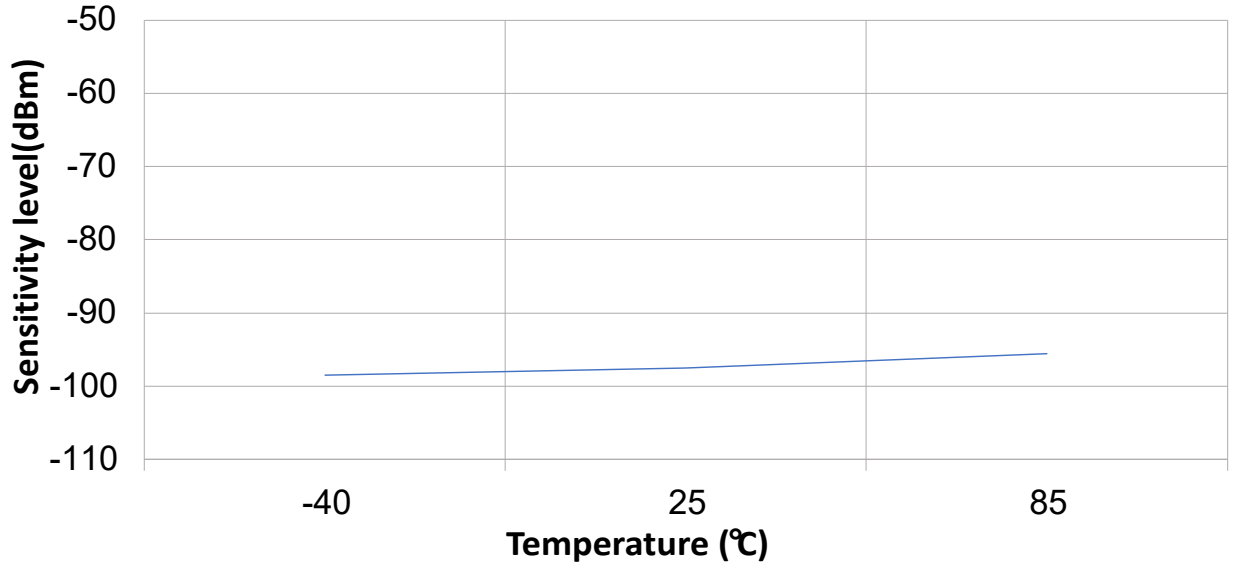
AC Characteristics					Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**RF Power**	**CPU Frequency**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
IBLETX1	IDDTXMPA	Current consumption with output power in DC-DC mode 1 Mbps	+12 dBm	64 MHz	—	42.8	—	mA	—
IBLETX2			+12 dBm	32 MHz	—	40.5	—	mA	—
IBLETX3			+12 dBm	8 MHz	—	39.0	—	mA	—
IBLETX4		Current consumption at +12 dBm output power in MLDO mode	+12 dBm	64 MHz	—	96.7	—	mA	—
IBLETX5			+12 dBm	32 MHz	—	91.8	—	mA	—
IBLETX6			+12 dBm	8 MHz	—	88.1	—	mA	—
IBLETX7	IDDTXLPA	Current consumption at +4 dBm output power in DC-DC mode 1 Mbps	4 dBm	64 MHz	—	24.9	—	mA	—
IBLETX8			4 dBm	32 MHz	—	22.9	—	mA	—
IBLETX9			4 dBm	8 MHz	—	21.1	—	mA	—
IBLETX10		Current consumption at +4 dBm output power in MLDO mode	4 dBm	64 MHz	—	55.5	—	mA	—
IBLETX11			4 dBm	32 MHz	—	48.7	—	mA	—
IBLETX12			4 dBm	8 MHz	—	45.8	—	mA	—
IBLETX7	IDDTXLPA0	Current consumption at +0 dBm output power in DC-DC mode 1 Mbps	0 dBm	64 MHz	—	22.7	—	mA	—
IBLETX8			0 dBm	32 MHz	—	20.9	—	mA	—
IBLETX9			0 dBm	8 MHz	—	18.2	—	mA	—
IBLETX10		Current consumption at 0 dBm output power in MLDO mode	0 dBm	64 MHz	—	47.6	—	mA	—
IBLETX11			0 dBm	32 MHz	—	43.0	—	mA	—
IBLETX12			0 dBm	8 MHz	—	39.7	—	mA	—

.....continued									
AC Characteristics					**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9-3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**RF Power**	**CPU Frequency**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
IBLERX1	IDDRXBLE1M	Current consumption at RX signal level -80 dBm in DC-DC mode	-80 dBm	64 MHz	—	20.6	—	mA	—
IBLERX2			-80 dBm	32 MHz	—	18.2	—	mA	—
IBLERX3			-80 dBm	8 MHz	—	16.5	—	mA	—
IBLERX4		Current consumption at RX signal level -80 dBm in MLDO mode	-80 dBm	64 MHz	—	40.6	—	mA	—
IBLERX5			-80 dBm	32 MHz	—	35.1	—	mA	—
IBLERX6			-80 dBm	8 MHz	—	30.9	—	mA	—

Notes:

- Current consumption is measured on a board based upon the Microchip Technology Reference Design.
- Current consumption is for the entire SoC (including the MCU), measured at the input power rail.
- Current consumption is measured using HUT code.
- Current reported is the average of the current during the transmit or receive burst (exclude off cycle of the transmit/receive operation).

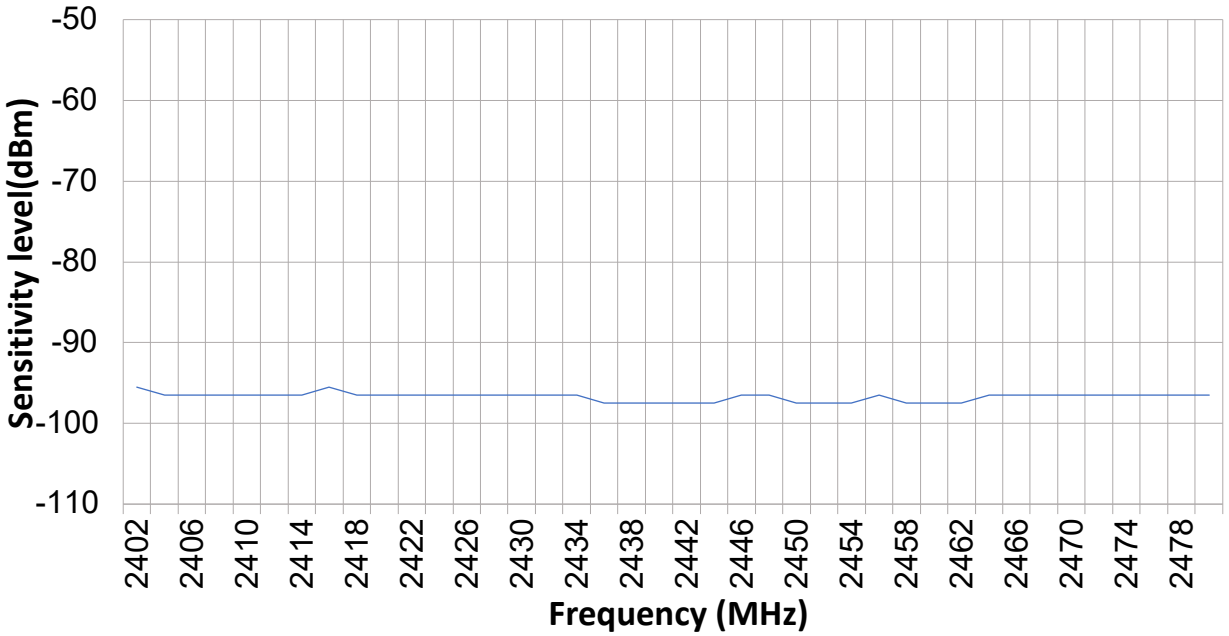
Figure 43-24. Module Bluetooth Low Energy Receive Sensitivity vs Temperature



Notes:

- Bluetooth Low Energy receive sensitivity is measured across temperature at 3.6V, 2440 MHz, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

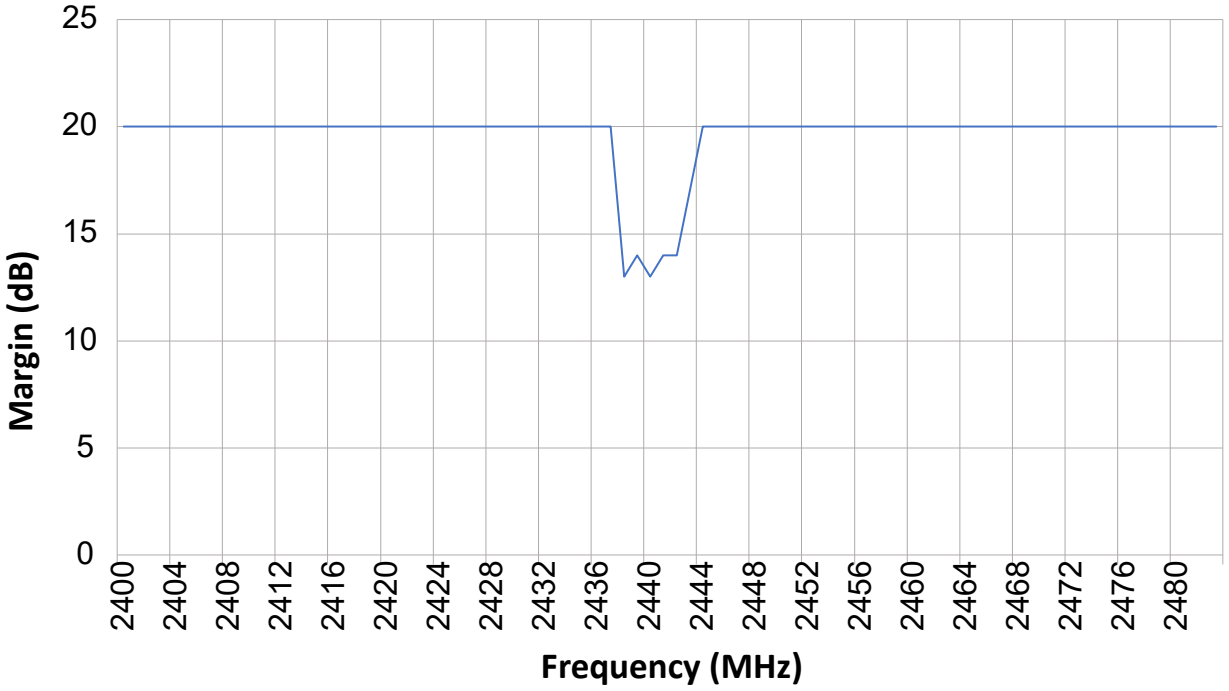
Figure 43-25. Module Bluetooth Low Energy Receive Sensitivity vs Frequency



Notes:

- Bluetooth Low Energy sensitivity is measured across channels at 3.6V at 25℃, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

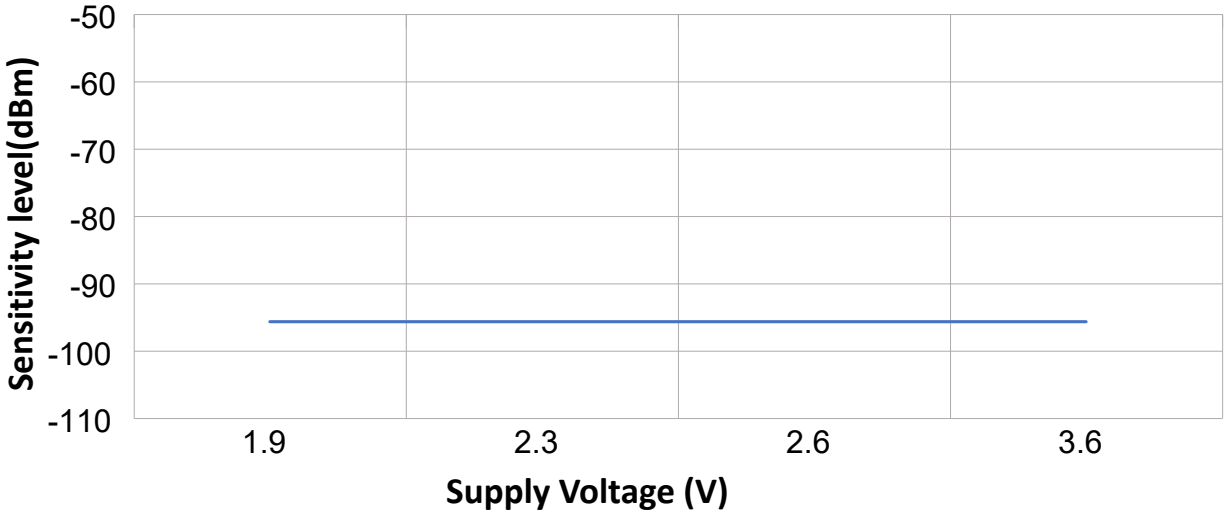
Figure 43-26. Bluetooth Low Energy 1M CI Margin



Notes:

- Bluetooth Low Energy 1M C/I Margin is measured at 2440 MHz at 25℃, 3.6V, uncoded data at 1 Ms/s.
- C/I test is done with HUT code based on the SIG specifications.
- Reported C/I margin is the margin above the C/I specifications from SIG.

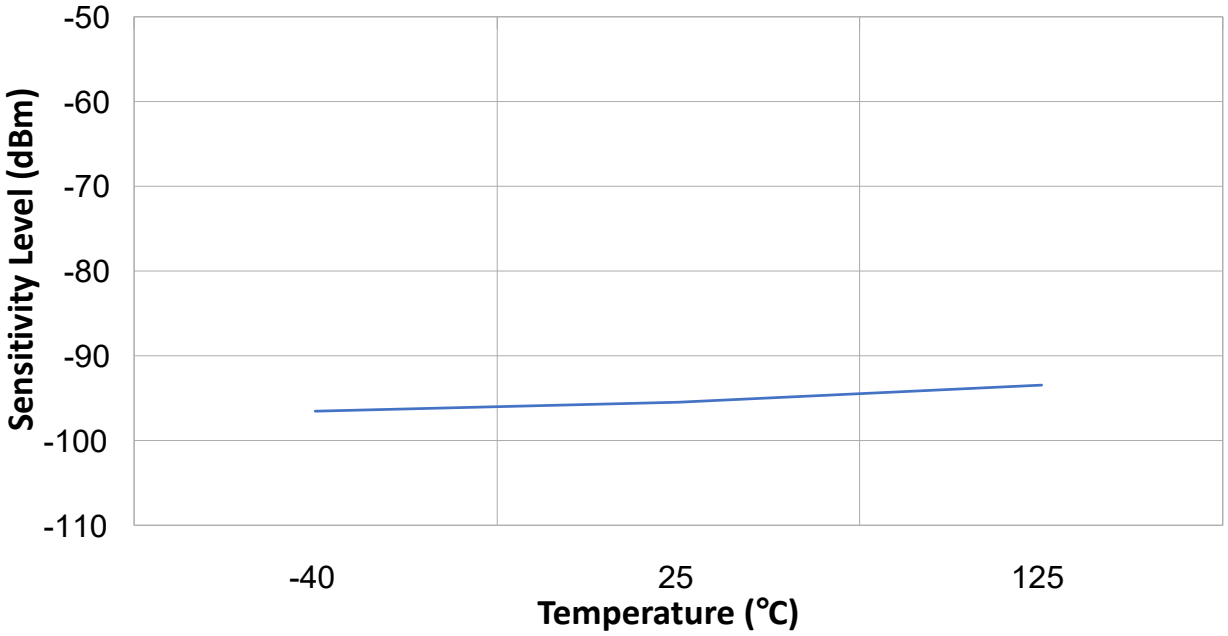
Figure 43-27. Bluetooth Low Energy Receive Sensitivity vs Voltage



Notes:

- Bluetooth Low Energy receive sensitivity is measured at 2440 MHz at 25℃, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

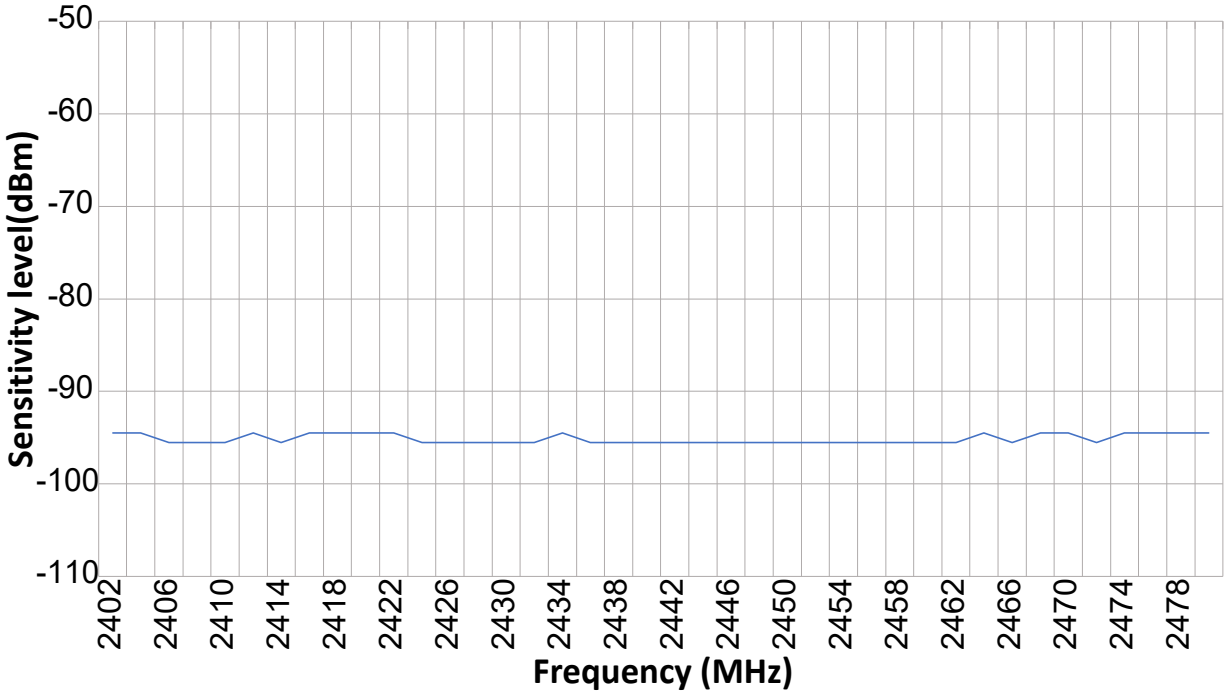
Figure 43-28. Bluetooth Low Energy Receive Sensitivity vs Temperature



Notes:

- Bluetooth Low Energy receive sensitivity is measured across channels at 3.6V, 2440 MHz, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

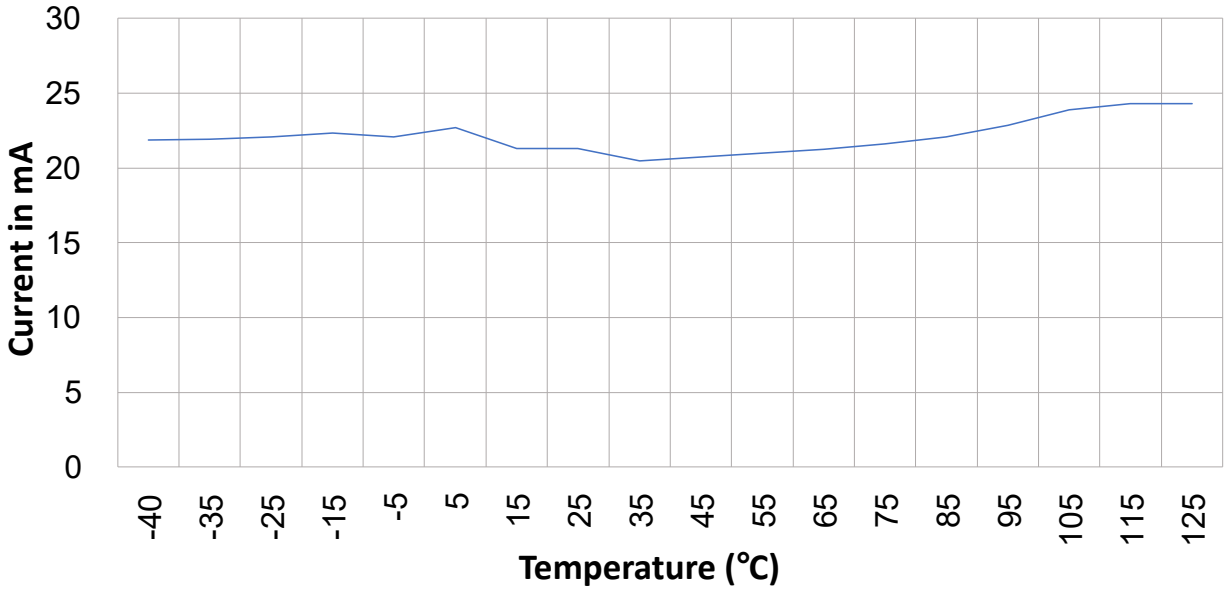
Figure 43-29. Bluetooth Low Energy Receive Sensitivity vs Frequency



Notes:

- Bluetooth Low Energy receiver sensitivity is measured across channels at 3.6V at 25℃, uncoded data at 1 Ms/s.
- PDU length = 37.
- Sensitivity is measured according to the SIG specifications.

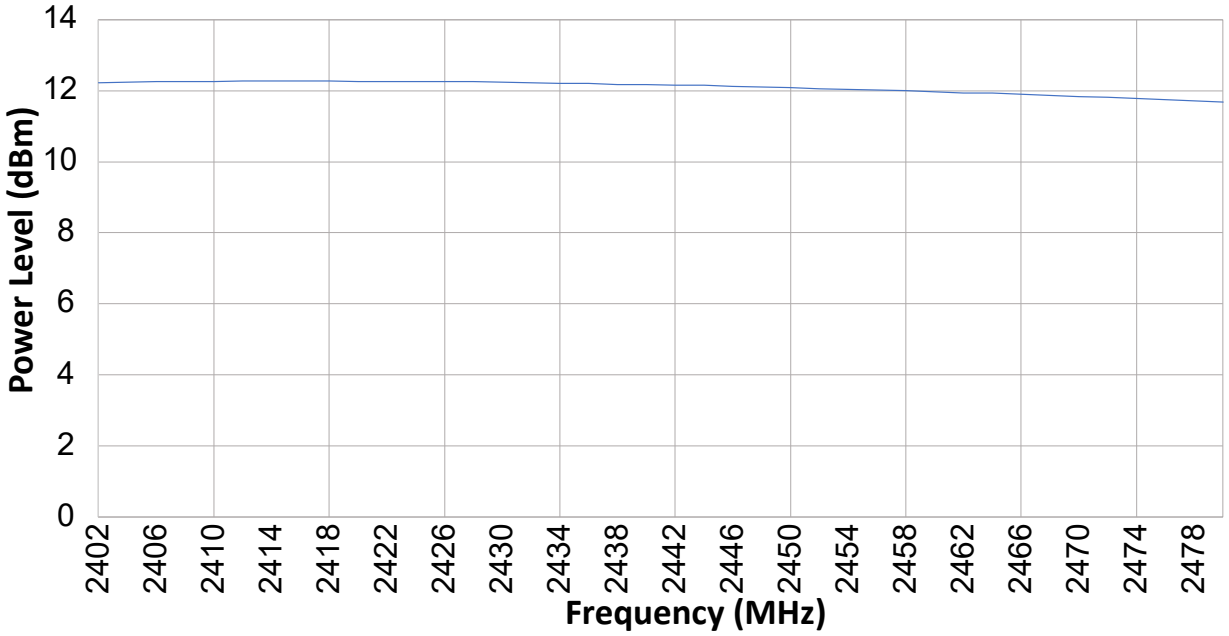
Figure 43-30. Bluetooth Low Energy Receive Current vs Temperature



Notes:

- Bluetooth Low Energy receive current is measured at 3.3V (Buck mode), uncoded data at 1 Ms/s with LNA configured at maximum gain.
- PDU length = 37.
- Current is measured on input power rail to SoC (includes processor current as well).
- Current is measured with HUT code.

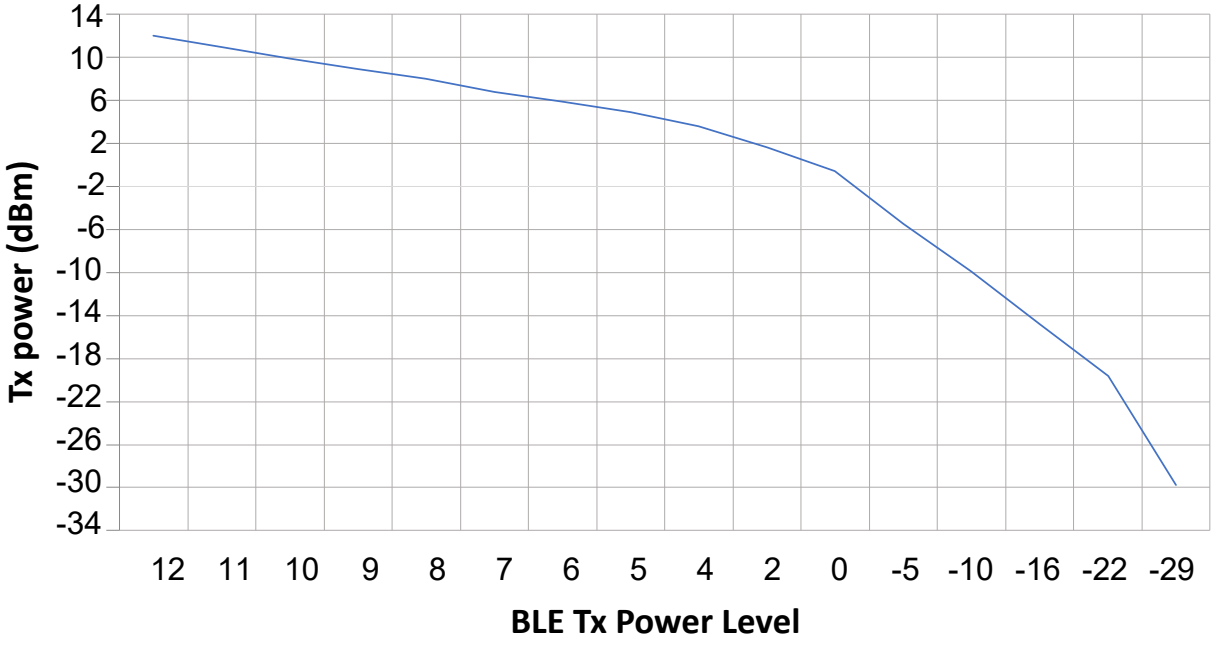
Figure 43-31. Bluetooth Low Energy Transmit Power vs Frequency



Notes:

- Bluetooth Low Energy transmit power is measured across frequency after transmit power calibration at 3.3V (Buck mode).
- Transmit power is measured with HUT code.
- Transmit power is measured after the PA matching and LPF.

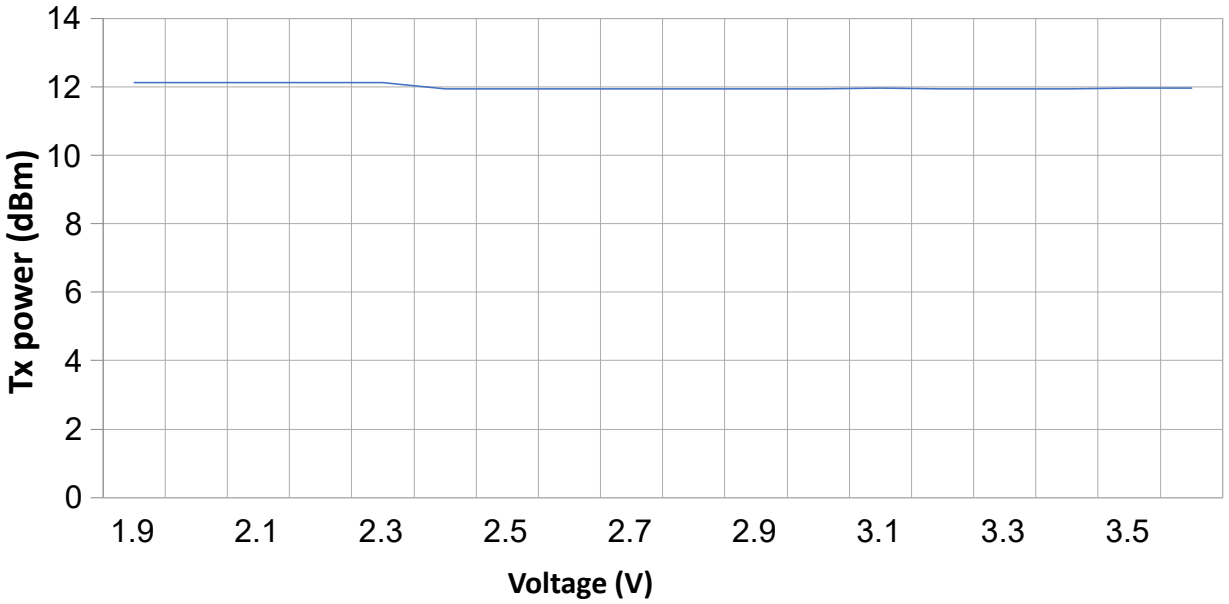
Figure 43-32. Bluetooth Low Energy Transmit Power vs Transmit Power Level



Notes:

- Bluetooth Low Energy transmit power is measured at 2440 MHz after transmit power calibration.
- Transmit power is measured on board based on Microchip Technology Reference Design.
- Transmit power is measured after PA match and LPF.
- Transmit power is measured with HUT code.
- Transmit power is controlled by transmit power settings on HUT code for measurement.

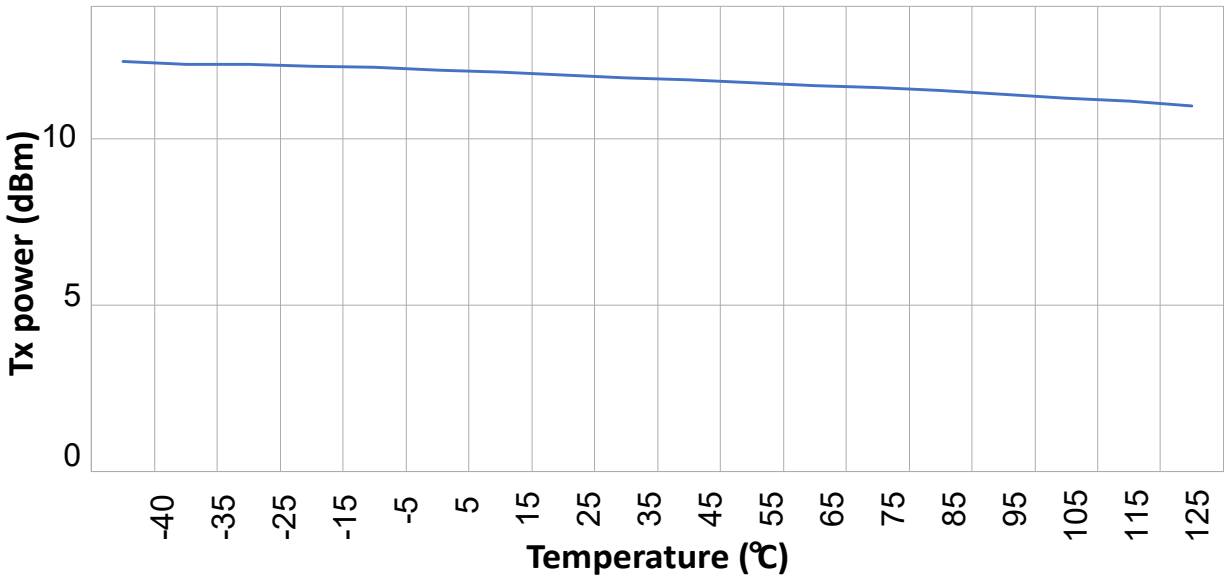
Figure 43-33. Bluetooth Low Energy Transmit Power vs VDD Supply Voltage



Notes:

- Bluetooth Low Energy transmit power is measured across voltage after transmit power calibration.
- Transmit power is measured after calibration at +12 dBm (± 0.5 dBm).
- Transmit power is measured on board based on the Microchip Reference Design.
- Transmit power is measured after the LPA and PA match section.
- Transmit power is measured with HUT code.

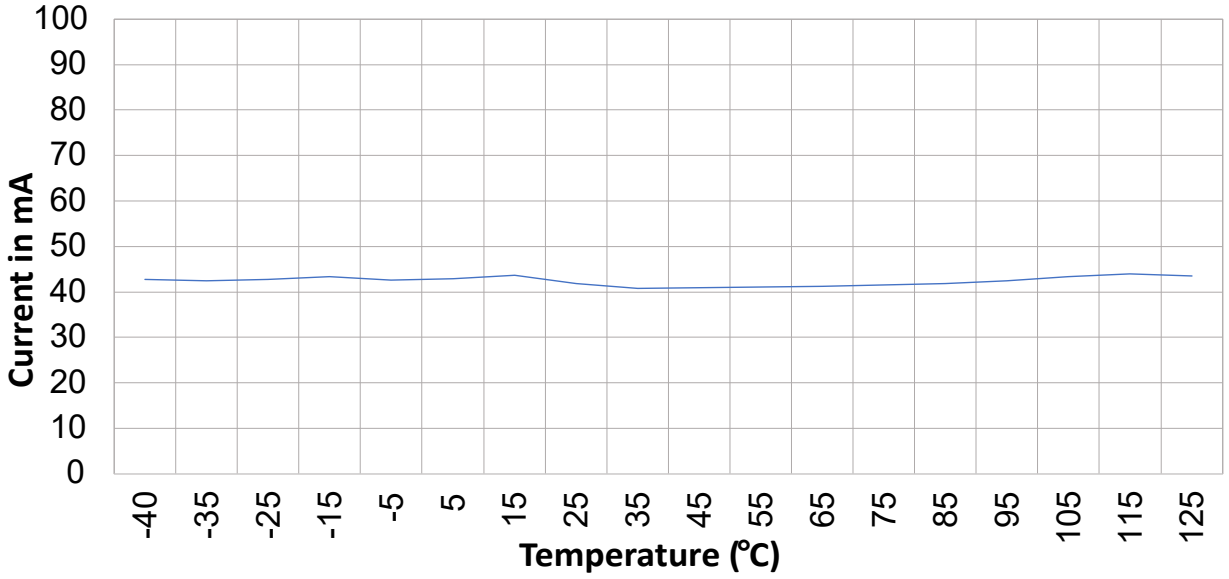
Figure 43-34. Bluetooth Low Energy Transmit Power vs. Temperature



Notes:

- Bluetooth Low Energy transmit power is measured across temperature after transmit power calibration at 3.6V and 2440 MHz.
- Transmit power is measured with HUT code.
- Temperature power compensation is triggered before power measurement.
- Transmit power is measured after the PA matching and LPF.

Figure 43-35. Bluetooth Low Energy Transmit Current vs Temperature



Notes:

- Bluetooth Low Energy transmit current is measured at 3.3V (Buck mode) at 2440 MHz across temperature.
- Transmit current is measured after calibration at +12 dBm (± 0.5 dBm).
- Current is measured on input power rail to SoC.
- Current is measured with HUT code.

43.21 Zigbee RF Characteristics

Table 43-28. Zigbee RF Characteristics

AC Characteristics			Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.(1,2)**	**Max.**	**Units**	**Conditions**
ZBG1	FREQ	Frequency range	2405	—	2480	MHz	—
ZBG2	FCH	Channel spacing	—	5	—	MHz	—
ZBG3	PSDU	Bit rate	—	250, 500, 1000, 2000	—	kbps	—
ZBT1	TXOPMPA	Transmit output power MPA	—	11.5	—	dBm	(4)
ZBT2	TXOPLPA	Transmit output power LPA	—	4.0	—	dBm	(4)
ZBT3	POWERRANGE	Output power range	—	26	—	dB	TX power on ZB power levels from (-14 to 12 dBm)
ZBT4	EVM	Error vector magnitude	—	10	—	%RMS	—
ZBRX1	SENS250	Receiver sensitivity in 250 kbps	—	-99	—	dBm	—
	SENS500	Receiver sensitivity in 500 kbps	—	-96	—	dBm	—
	SENS1M	Receiver sensitivity in 1 Mbps	—	-94	—	dBm	—
	SENS2M	Receiver sensitivity in 2 Mbps	—	-88	—	dBm	—
ZBRX2	PMAX	Maximum input level	—	0	—	dBm	—
ZBRX3(3)	PACRP	Adjacent channel rejection +5 MHz	—	35	—	dB	(6)
ZBRX4(3)	PACRN	Adjacent channel rejection -5 MHz	—	31	—	dB	
ZBRX5(3)	PALRP	Alternate channel rejection +10 MHz	—	17	—	dB	
ZBRX6(3)	PALRN	Alternate channel rejection -10 MHz	—	17	—	dB	
ZBRX7	LOLEAKLPA	LO leakage on LPA mode	—	-34	—	dB	(5)
ZBRX7A	LOLEAKMPA	LO leakage on MPA mode	—	-28	—	dB	(5)

.....continued							
AC Characteristics			**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**Min.**	**Typ.(1,2)**	**Max.**	**Units**	**Conditions**
ZBRX8	RSSIRANGE	Dynamic range of RSSI	—	40	—	dB	—

Notes:

1. Measured on a board based on the Microchip Technology reference design.
2. Measured across channels at 3.3V and according to the 802.15.4 standard specifications.
3. Specified value is the margin above the 802.15.4 standard limits.
4. Measured across channels and voltages.
5. LO leakage on LPA mode, measured across voltage.
6. All results are based on measurement conditions as per the 802.15.4 standards.

Table 43-29. Zigbee RF Current Characteristics

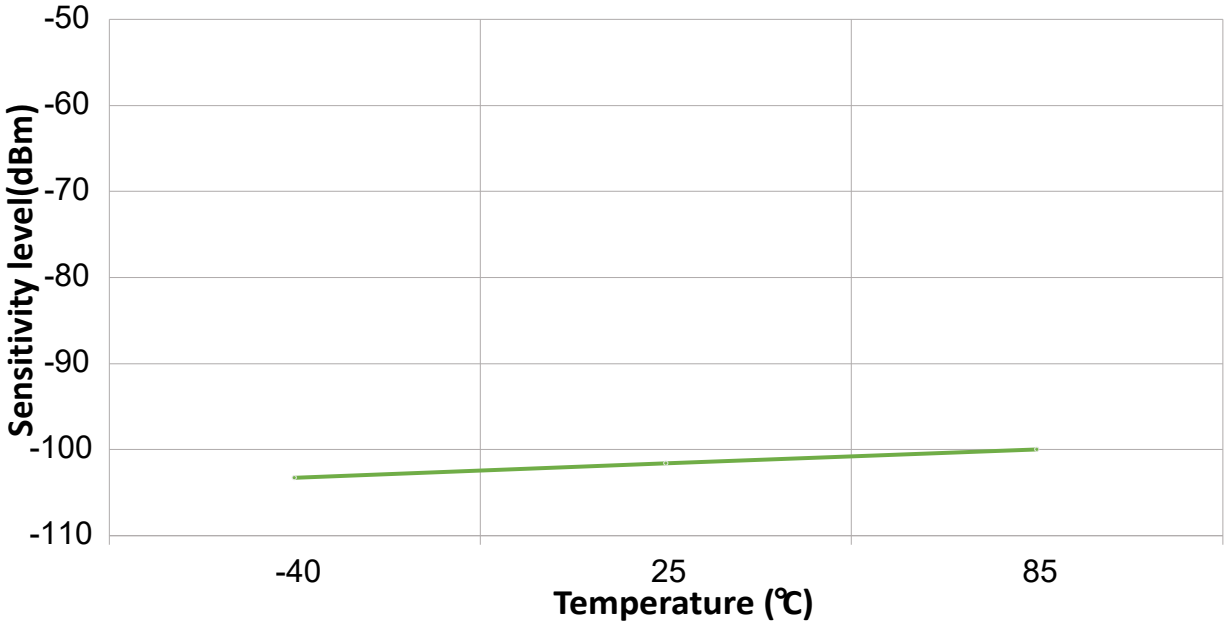
AC Characteristics				**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**CPU Frequency**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
IZBTX1	IDDTXMPA	Current consumption at +12 dBm output power in DC-DC mode 250 kbps	64 MHz	—	43.3	—	mA	—
IZBTX2			32 MHz	—	41.3	—	mA	—
IBLETX3			8 MHz	—	39.9	—	mA	—
IZBTX4		Current consumption at +12 dBm output power in MLDO mode	64 MHz	—	96.4	—	mA	—
IZBTX5			32 MHz	—	92.1	—	mA	—
IZBTX6			8 MHz	—	89.0	—	mA	—
IZBTX7	IDDTXLPA	Current consumption at +4 dBm output power in DC-DC mode 250 kbps	64 MHz	—	27.3	—	mA	—
IZBTX8			32 MHz	—	24.9	—	mA	—
IZBTX9			8 MHz	—	22.7	—	mA	—
IZBTX10		Current consumption at +4 dBm output power in MLDO mode	64 MHz	—	51.7	—	mA	—
IZBTX11			32 MHz	—	47.6	—	mA	—
IZBTX12			8 MHz	—	45.0	—	mA	—
IZBRX1	IDDRXZB	Current consumption at RX signal level -95 dBm in DC-DC mode	64 MHz	—	19.4	—	mA	—
IZBRX2			32 MHz	—	17.4	—	mA	—
IZBRX3			8 MHz	—	15	—	mA	—
IZBRX4		Current consumption at RX signal level -95 dBm in MLDO mode	64 MHz	—	38.5	—	mA	—
IZBRX5			32 MHz	—	33.5	—	mA	—
IZBRX6			8 MHz	—	30.3	—	mA	—

.....continued								
AC Characteristics				**Standard Operating Conditions: V_{DDIO} = V_{DDANA} 1.9V to 3.6V (unless otherwise stated) Operating Temperature: -40°C ≤ T_A ≤ +85°C for Industrial Temp**				
Param. No.	**Symbol**	**Characteristics**	**CPU Frequency**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
IZBRX1	IDDRXZBRPC	Current consumption at RX signal level -95 dBm in DC-DC mode	64 MHz	—	13.6	—	mA	—
IZBRX2			32 MHz	—	11.3	—	mA	—
IZBRX3			8 MHz	—	9.8	—	mA	—
IZBRX4		Current consumption at RX signal level -95 dBm in MLDO mode	64 MHz	—	29	—	mA	—
IZBRX5			32 MHz	—	24.2	—	mA	—
IZBRX6			8 MHz	—	20.7	—	mA	—

Notes:

- Current consumption is measured on a board based upon the Microchip Technology Reference Design.
- Current consumption is for the entire SoC (including the MCU).
- Current consumption is measured using the HUT code.
- Current reported is the average of the current during the transmit burst (exclude off cycle of the transmission).

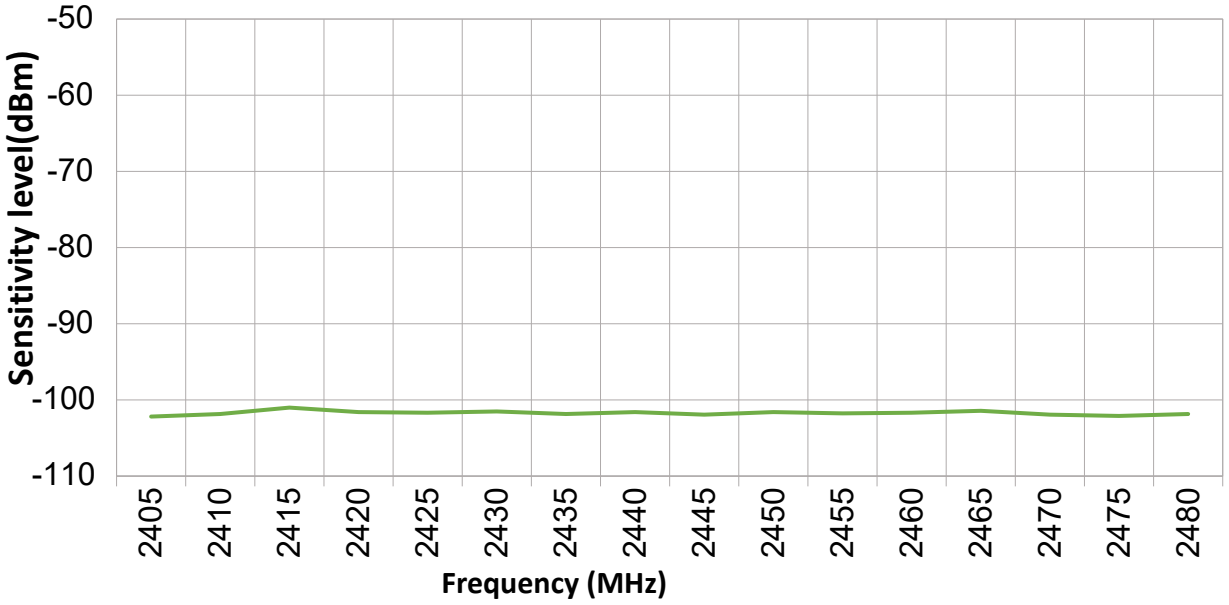
Figure 43-36. Module Zigbee Receive Sensitivity vs. Temperature



Notes:

- Receiver sensitivity is measured based on the 802.15.4 specifications.
- Receiver sensitivity is measured at 2440 MHz at 3.6V, 250 kbps.
- Measured after receiver calibration.

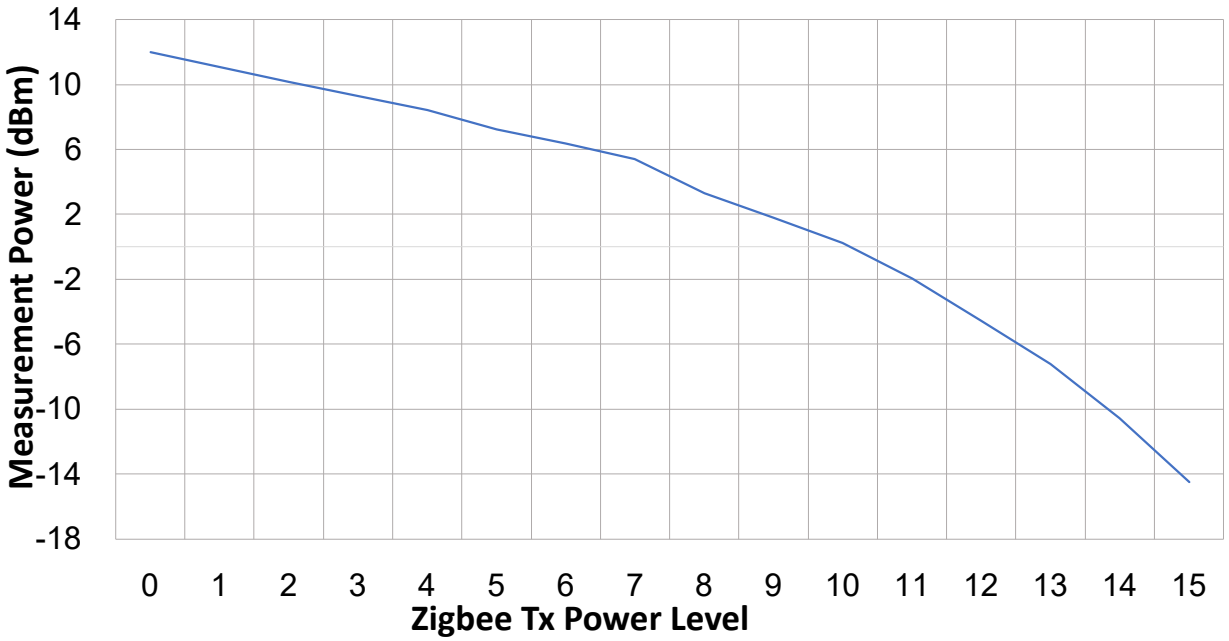
Figure 43-37. Module Zigbee Receive Sensitivity vs. Frequency



Notes:

- RX sensitivity across channels is measured at 3.6V at 25°C, 250 kbps.
- Sensitivity is measured according to the 802.15.4 specifications.

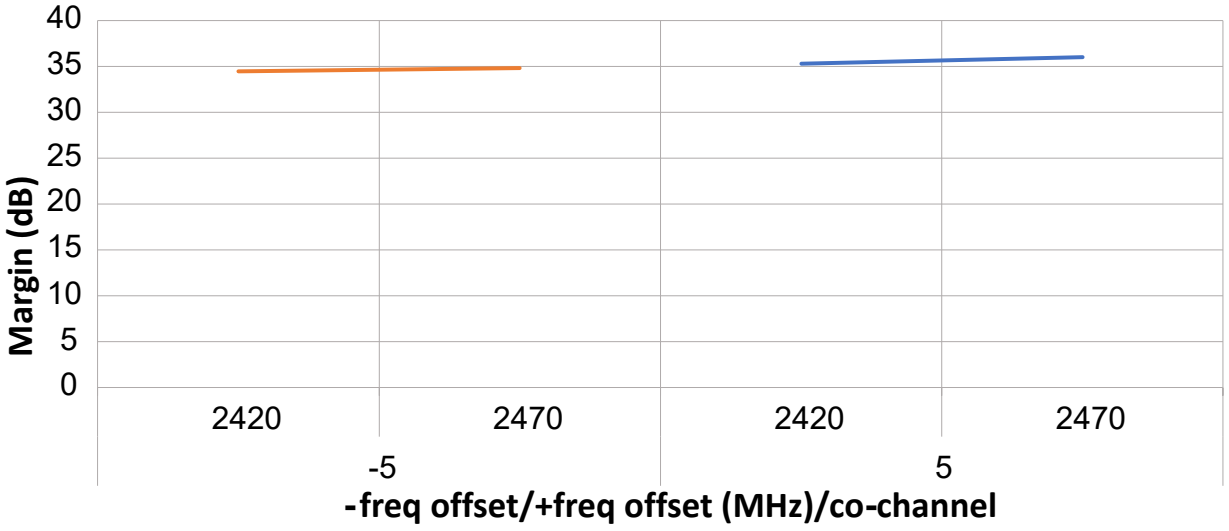
Figure 43-38. Zigbee TX Setting vs. Measurement Power



Notes:

- Transmit power is measured after calibration.
- Transmit power is measured across power levels at 2440 MHz at 3.6V, 25°C.
- Transmit power is measured after the PA matching and LPF.
- Transmit power is configured using the transmit power setting in HUT code.

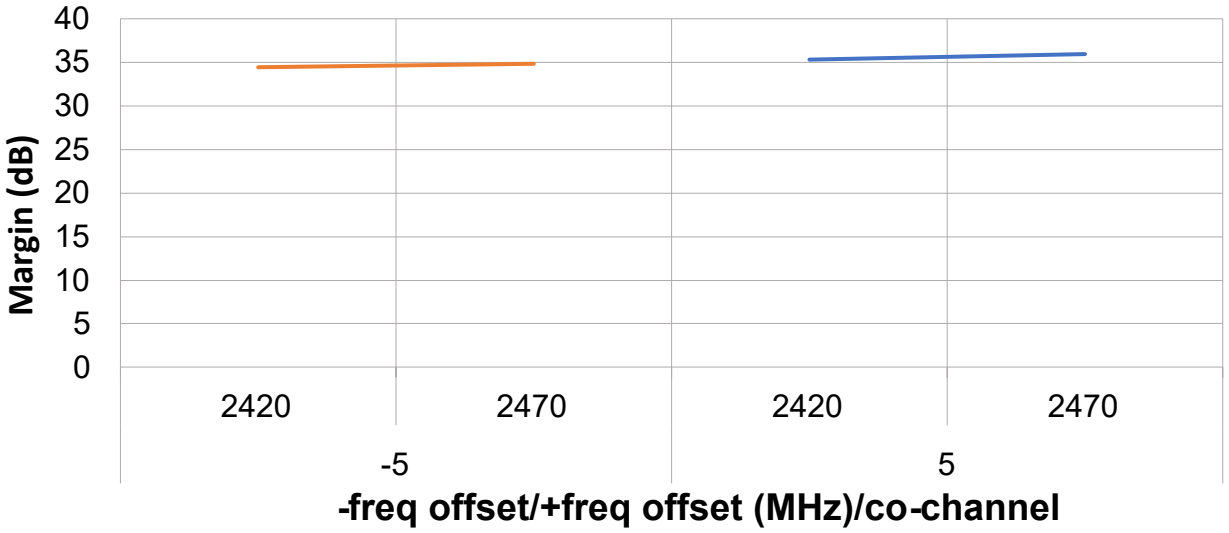
Figure 43-39. Zigbee ACR +-5M Margin



Notes:

- Adjacent channel rejection is measured at 3.6V at 25℃, 250 kbps.
- Measured based on the 802.15.4 adjacent channel relative jamming specification.
- Margin specified is the margin above the 802.15.4 specifications.
- Measured after receiver calibration.

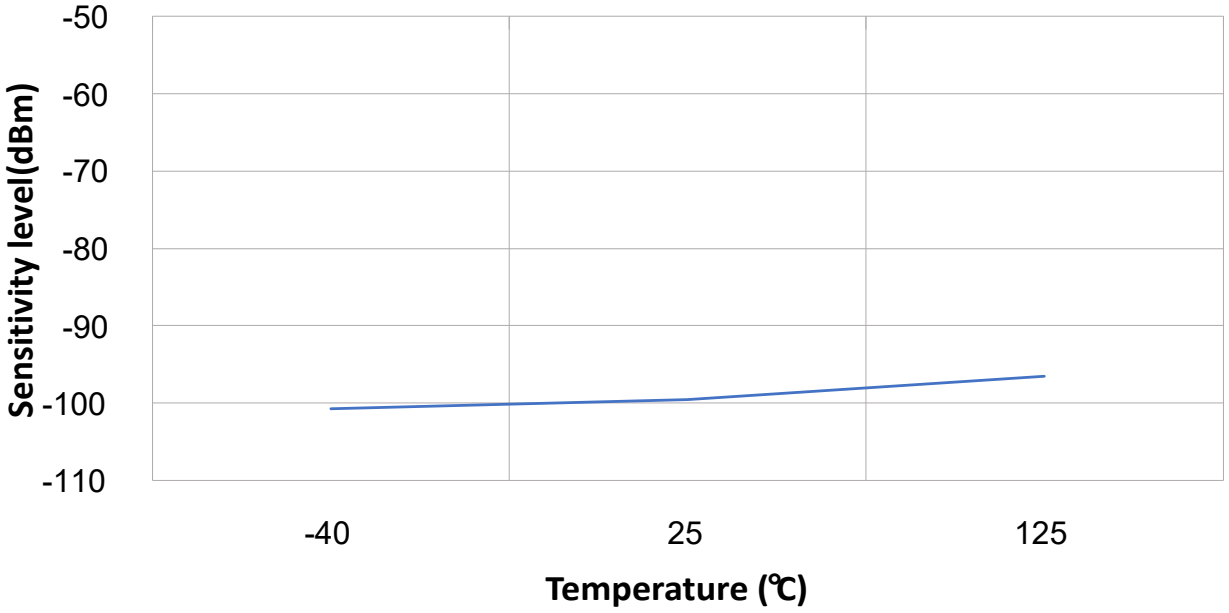
Figure 43-40. Zigbee ACR +-10M Margin



Notes:

- Adjacent channel rejection is measured at 3.6V at 25℃, 250 kbps.
- Measured based on the 802.15.4 adjacent channel relative jamming specification.
- Margin specified is the margin above the 802.15.4 specifications.
- Measured after receiver calibration.

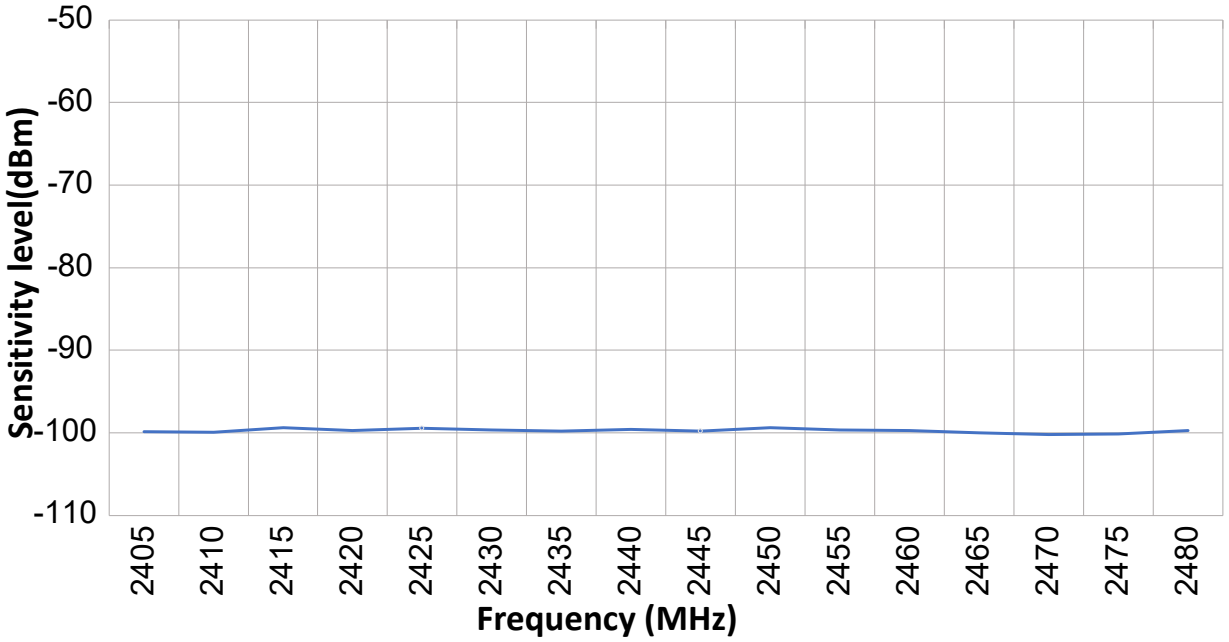
Figure 43-41. Zigbee Receive Sensitivity vs. Temperature



Notes:

- Receiver sensitivity is measured based on the 802.15.4 specifications.
- Sensitivity measured at 3.6V, 25℃ on 2440 MHz across temperature, 250 kbps.
- Measured after receiver calibration.

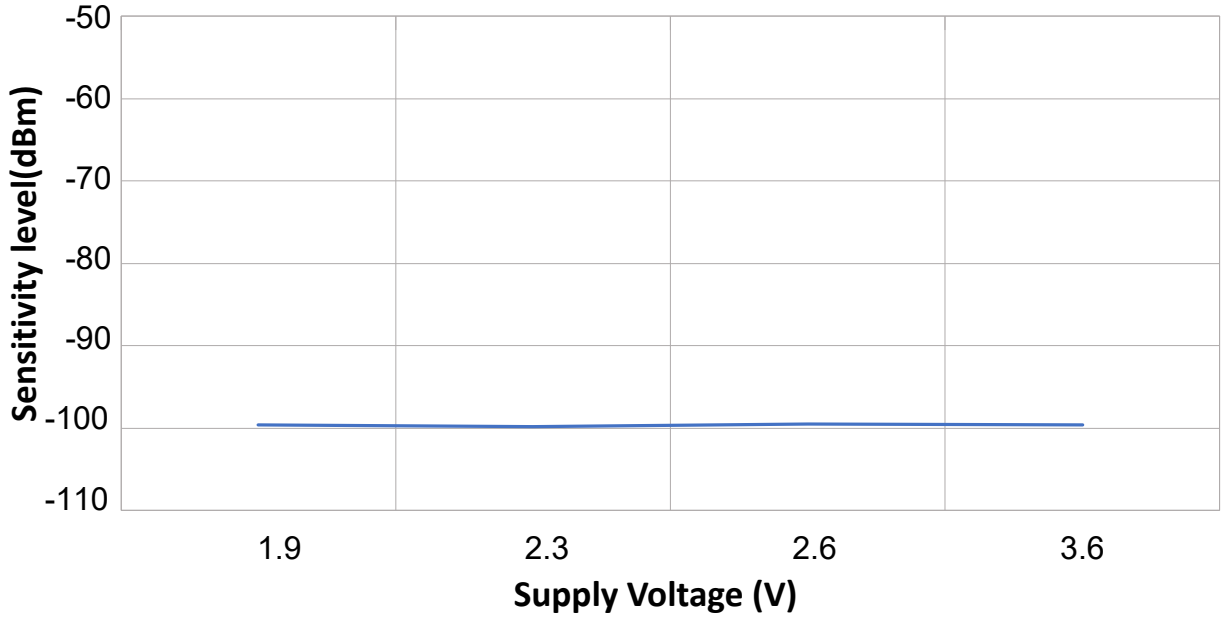
Figure 43-42. Zigbee Receive Sensitivity vs. Frequency



Notes:

- RX sensitivity is measured across channels at 3.6V at 25℃, 250 kbps.
- Sensitivity is measured according to the 802.15.4 specifications.
- Sensitivity is measured after RX calibration.

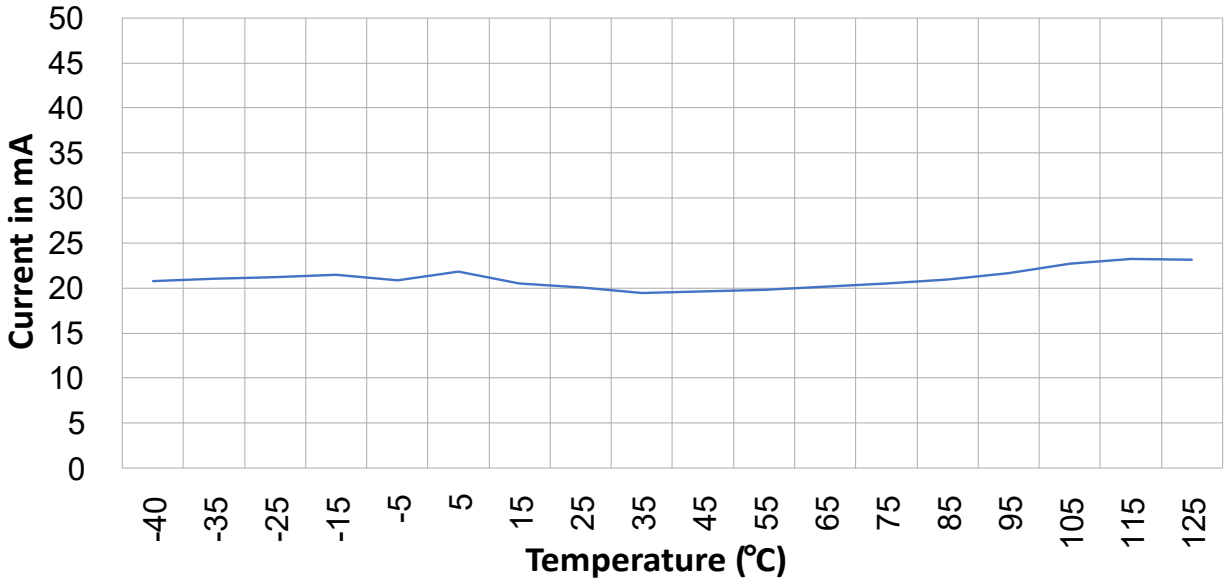
Figure 43-43. Zigbee Receive Sensitivity vs. VDD Supply Voltage



Notes:

- RX sensitivity is measured at 2440 MHz at 25℃ across voltage, 250 kbps.
- Sensitivity is measured according to the 802.15.4 specifications.
- Sensitivity is measured after RX calibration.

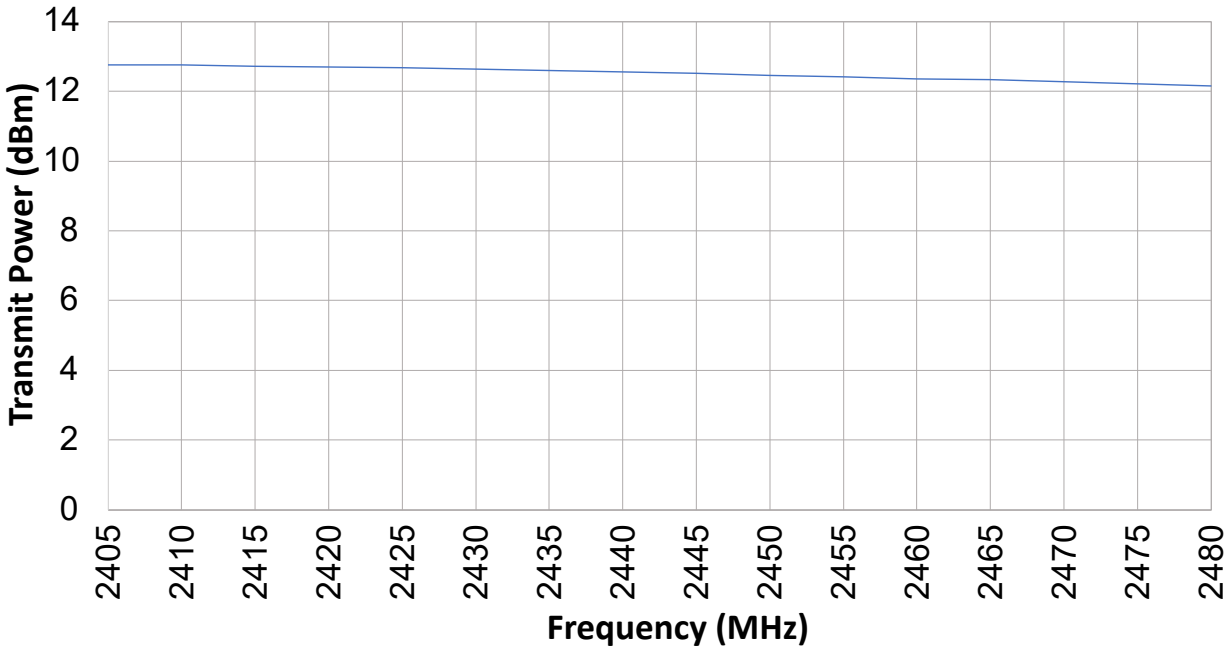
Figure 43-44. Zigbee Receive Current vs. Temperature



Notes:

- Receiver operating at 2440 MHz, 3.3V, 25℃ at maximum LNA gain.
- Measured after receiver calibration.

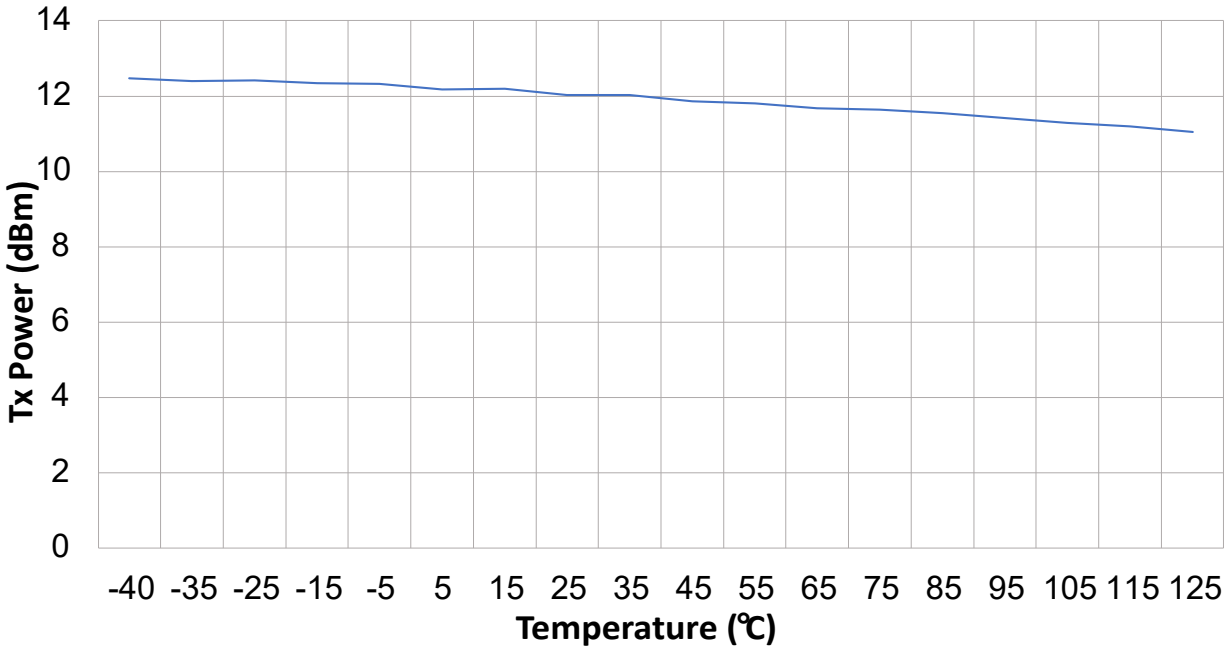
Figure 43-45. Zigbee Transmit Power vs Frequency



Notes:

- Transmit power is measured after calibration.
- Transmit power is measured across the channels at 3.6V at 25℃.
- Transmit power is measured after the PA matching and LPF.

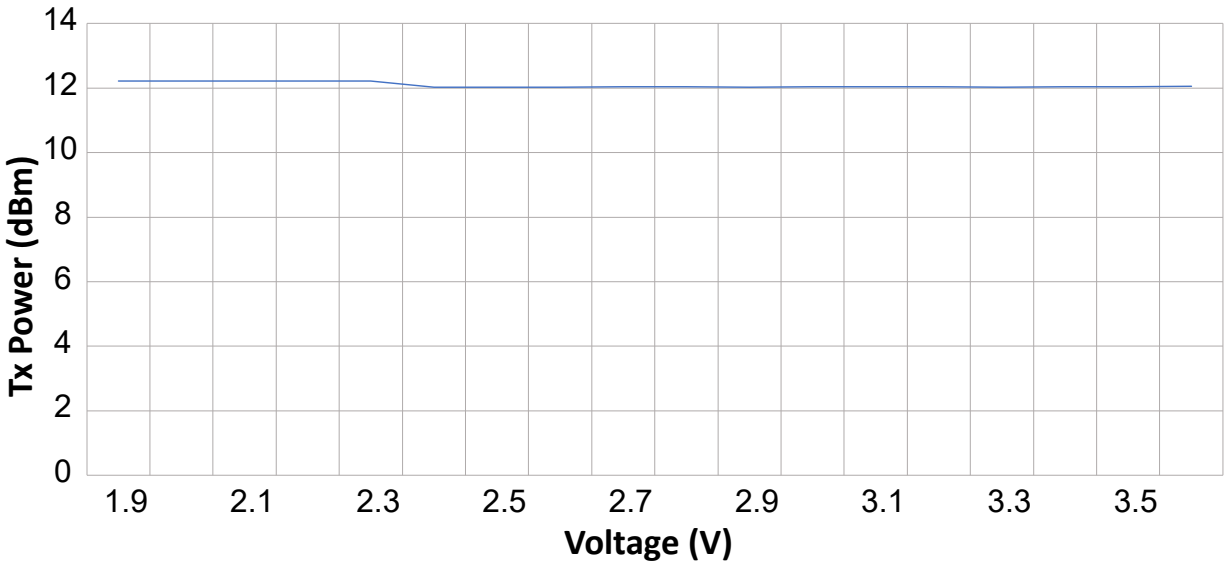
Figure 43-46. Zigbee Transmit Power vs Temperature



Notes:

- Transmit power is measured after calibration.
- Transmit power is measured at 2440 MHz at 3.6V across temperature.
- Transmit power is measured after the PA matching and LPF.
- Transmit power compensation is triggered before measurement across temperature.

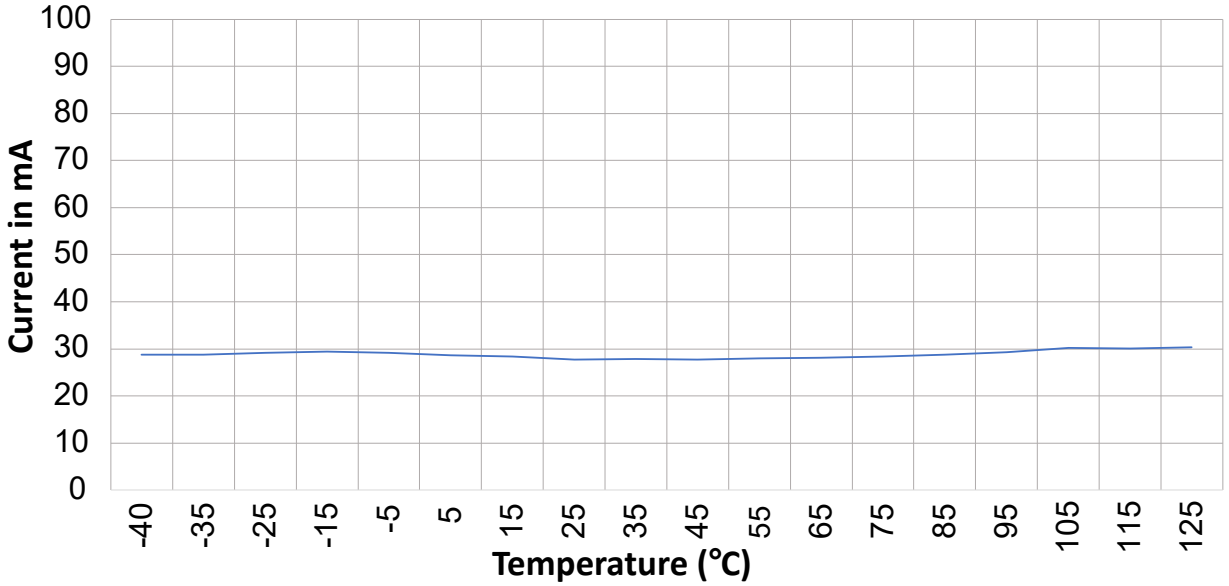
Figure 43-47. Zigbee Transmit Power vs. VDD Supply Voltage



Notes:

- Transmit power is measured after calibration.
- Transmit power is measured across voltage at 2440 MHz and 25℃.
- Transmit power is measured on reference board after the PA matching and LPF.
- Transmit power is configured using transmit power setting in HUT code.

Figure 43-48. Zigbee Transmit Current vs. Temperature



Notes:

- Transmit current is measured at input to SoC (includes SoC power consumption).
- Transmit current is measured at 2440 MHz at 3.3V (Buck mode).
- Transmit power is calibrated to +12 dBm (± 0.5 dBm on MPA mode).

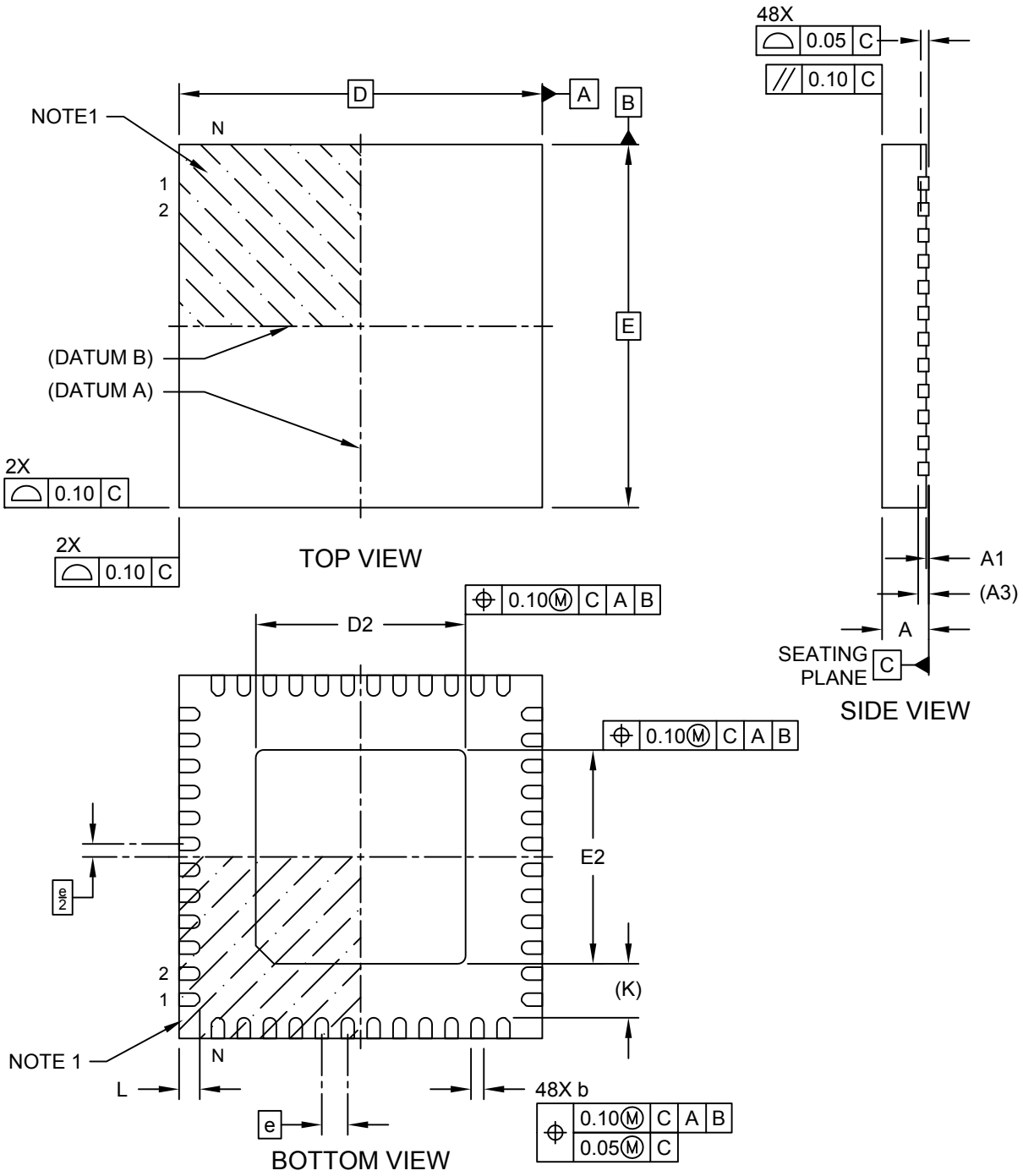
44. Packaging Information

This chapter provides the information on package markings, dimension and footprint of the PIC32CX-BZ2 and WBZ45 family.

44.1 PIC32CX-BZ2 SoC Packaging Information

48-Lead Very Thin Quad Flat, No Lead Package (MYX) 7x7x0.9 mm Body [VQFN] With 4.04x4.12 mm Exposed Pad

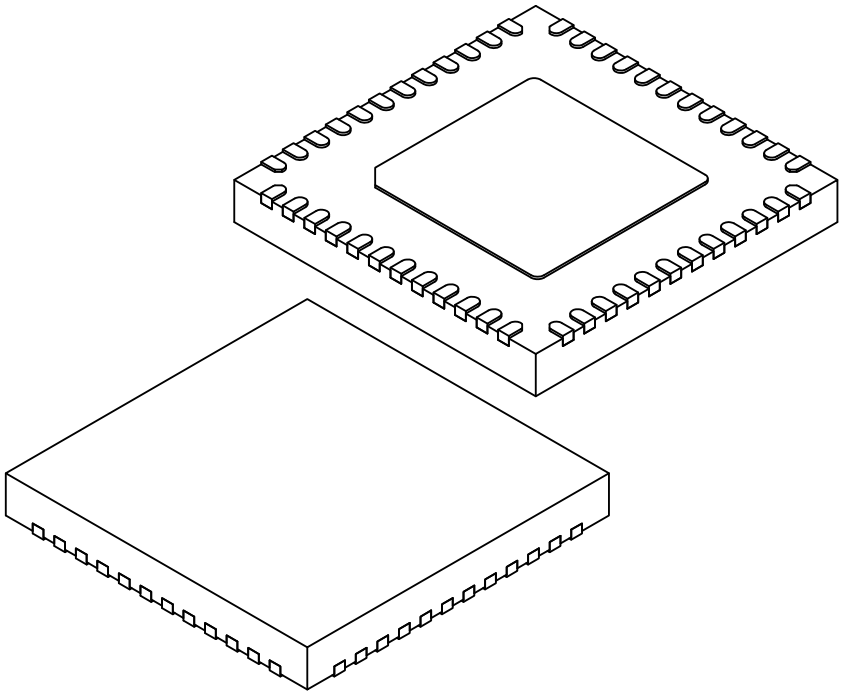
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-507 Rev A Sheet 1 of 2

48-Lead Very Thin Quad Flat, No Lead Package (MYX) 7x7x0.9 mm Body [VQFN] With 4.04x4.12 mm Exposed Pad

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		48	
Pitch	e		0.50 BSC	
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.20 REF	
Overall Length	D		7.00 BSC	
Exposed Pad Length	D2	3.94	4.04	4.14
Overall Width	E		7.00 BSC	
Exposed Pad Width	E2	4.02	4.12	4.22
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.35	0.40	0.45
Terminal-to-Exposed-Pad	K		1.04 REF	

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

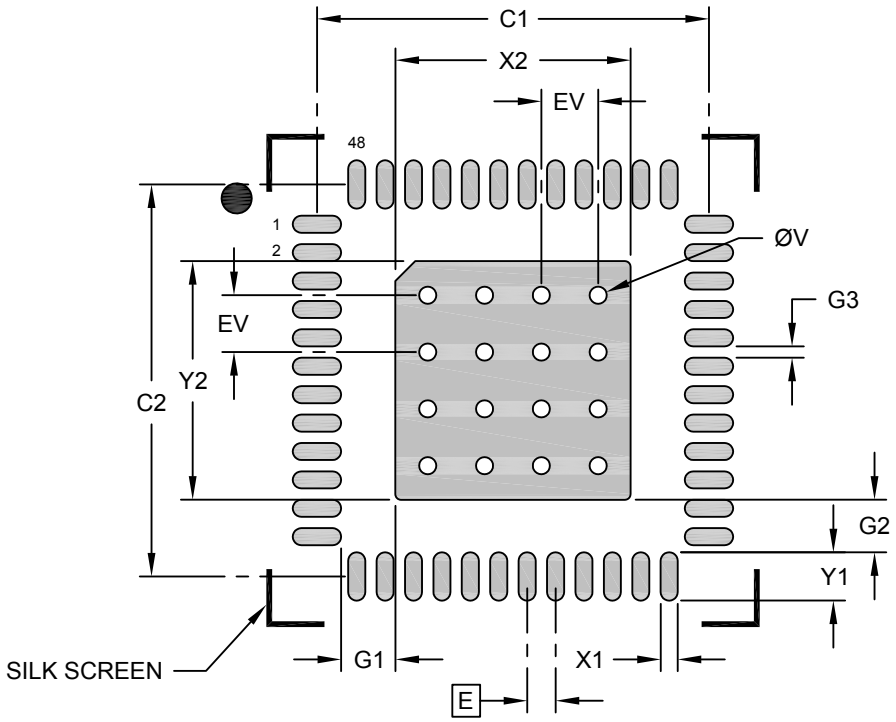
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-507 Rev A Sheet 2 of 2

48-Lead Very Thin Quad Flat, No Lead Package (MYX) 7x7x0.9 mm Body [VQFN] With 4.04x4.12 mm Exposed Pad

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	X2			4.14
Optional Center Pad Length	Y2			4.20
Contact Pad Spacing	C1		6.90	
Contact Pad Spacing	C2		6.90	
Contact Pad Width (X48)	X1			0.30
Contact Pad Length (X48)	Y1			0.85
Contact Pad to Center Pad (X24)	G1	0.96		
Contact Pad to Center Pad (X24)	G2	0.93		
Contact Pad to Contact Pad (X44)	G3	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

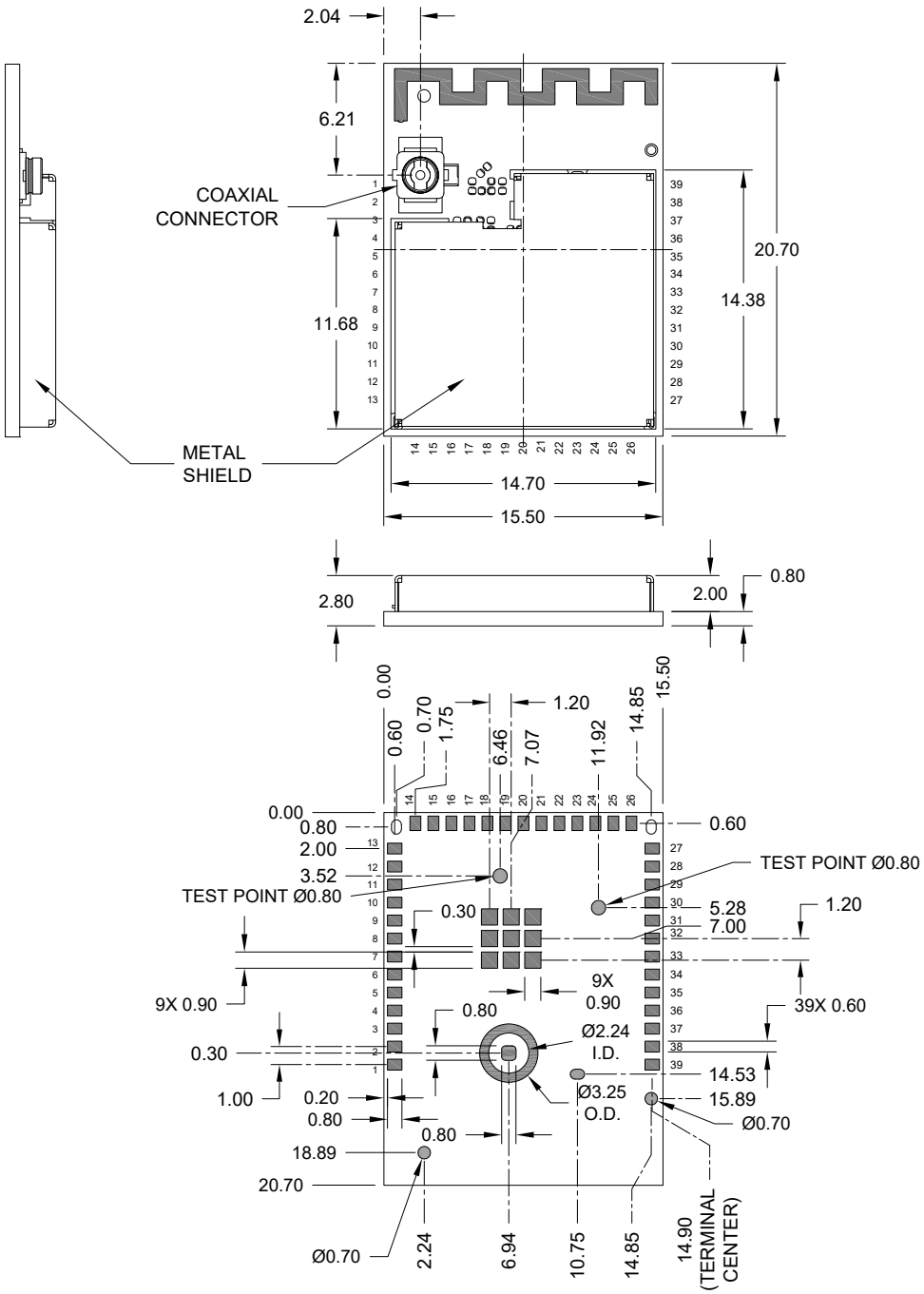
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2507 Rev A

44.2 WBZ451 Module Packaging Information

39-Lead PCB Module (ZSX) - 15.5x20.7x2.8 mm Body [MODULE] With Metal Shield and Coaxial Connector

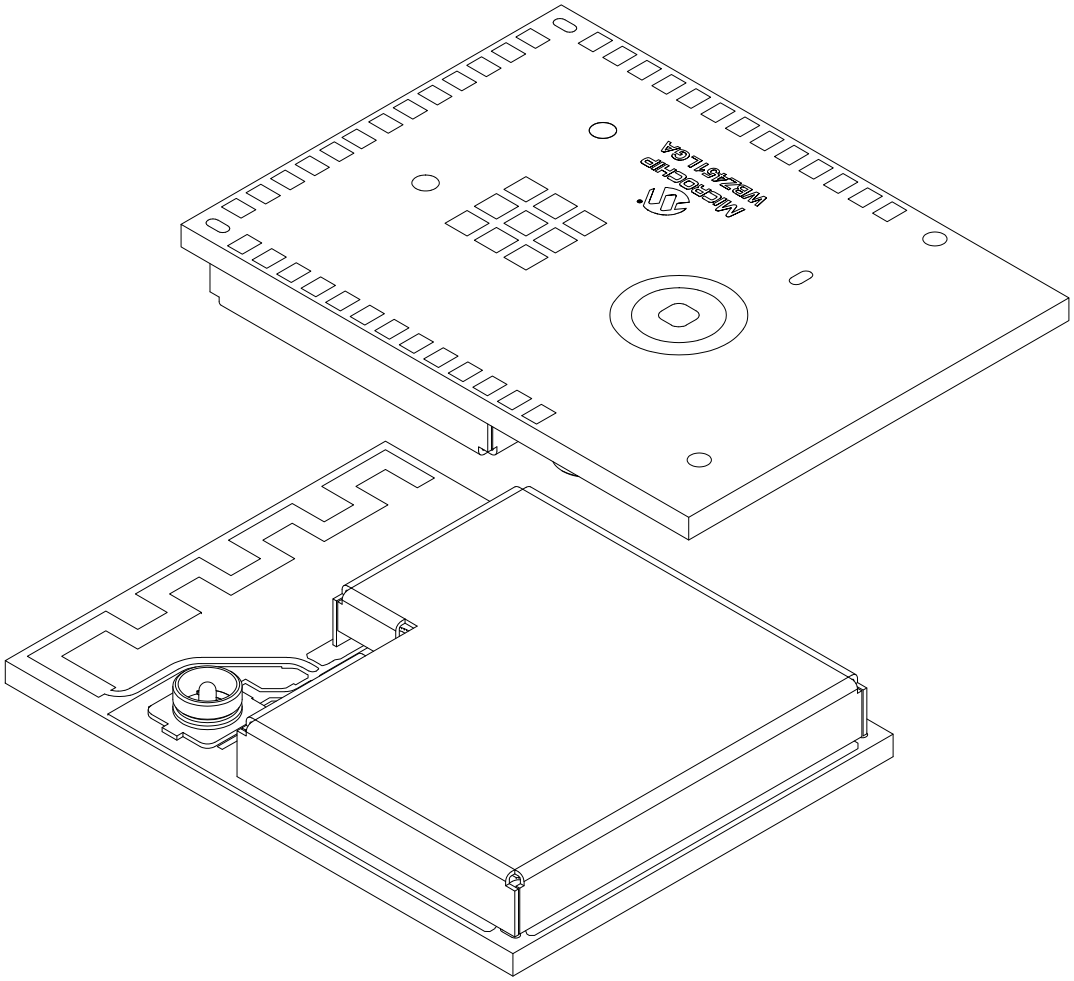
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-10052 Rev B Sheet 1 of 2

39-Lead PCB Module (ZSX) - 15.5x20.7x2.8 mm Body [MODULE]
With Metal Shield and Coaxial Connector

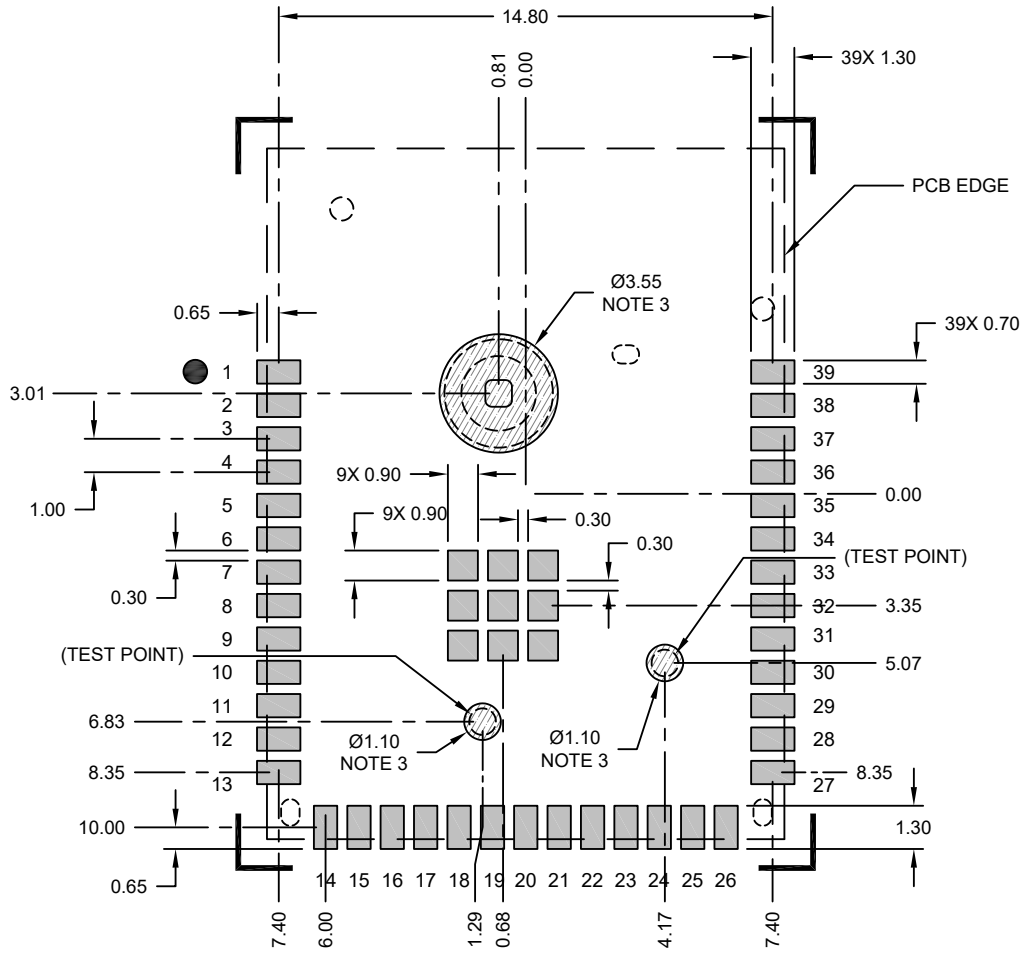
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-10052 Rev B Sheet 2 of 2

39-Lead PCB Module (ZSX) - 15.5x20.7x2.8 mm Body [MODULE]
With Metal Shield and Coaxial Connector

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



COPPER KEEPOUT ZONE

RECOMMENDED LAND PATTERN

Notes:

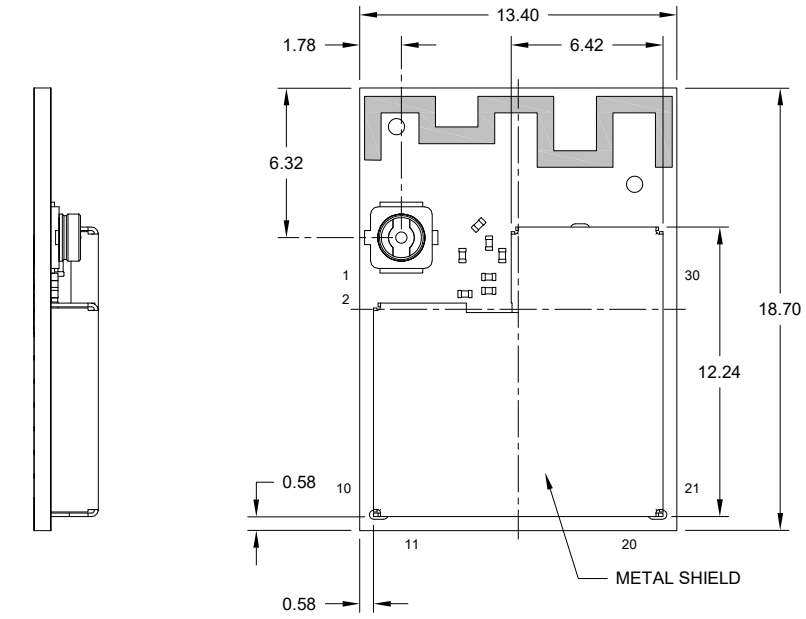
1. All dimensions are in millimeters.
2. Keep this area free from all metal, including ground flll.
3. Keep these areas free from routes and exposed copper. Ground fill with solder mask may be placed here.

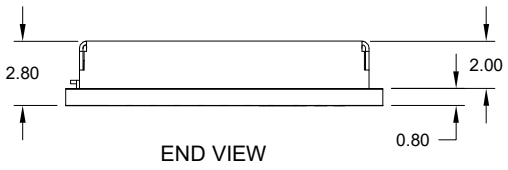
Microchip Technology Drawing C04-12052 Rev B

44.3 WBZ450 Module Packaging Information

30-Lead PCB Module (ZRX) -13.4x18.7x2.8 mm Body For WBZ450

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

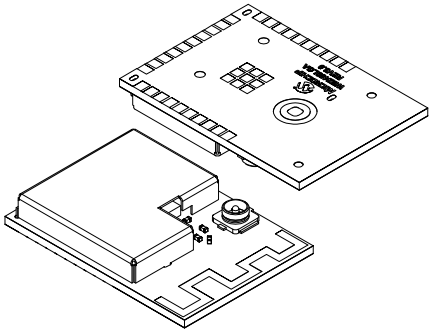


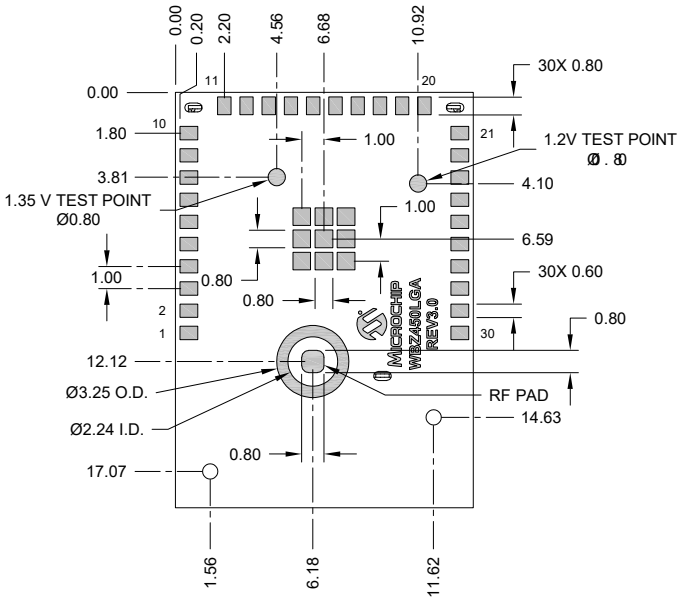


Microchip Technology Drawing C04-10051 Rev A Sheet 1 of 2

30-Lead PCB Module (ZRX) -13.4x18.7x2.8 mm Body For WBZ450

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging





BOTTOM VIEW

Microchip Technology Drawing C04-10051 Rev A

45. Appendix A: Regulatory Approval

The WBZ450 module has received regulatory approval for the following countries:

- Bluetooth Special Interest Group (SIG) QDID: D056857
- United States/FCC ID: 2ADHKWBZ450
- Canada/ISED:
  - IC: 20266-WBZ450
  - HVIN: WBZ450
- Europe/CE

45.1 United States

The WBZ450 module has received Federal Communications Commission (FCC) CFR47 Telecommunications, Part 15 Subpart C “Intentional Radiators” single-modular approval in accordance with Part 15.212 Modular Transmitter approval. Single-modular transmitter approval is defined as a complete RF transmission sub-assembly, designed to be incorporated into another device, that must demonstrate compliance with FCC rules and policies independent of any host. A transmitter with a modular grant can be installed in different end-use products (referred to as a host, host product or host device) by the grantee or other equipment manufacturer, then the host product may not require additional testing or equipment authorization for the transmitter function provided by that specific module or limited module device.

The user must comply with all of the instructions provided by the Grantee, which indicate installation and/or operating conditions necessary for compliance.

A host product itself is required to comply with all other applicable FCC equipment authorization regulations, requirements, and equipment functions that are not associated with the transmitter module portion. For example, compliance must be demonstrated: to regulations for other transmitter components within a host product; to requirements for unintentional radiators (Part 15 Subpart B), such as digital devices, computer peripherals, radio receivers, etc.; and to additional authorization requirements for the non-transmitter functions on the transmitter module (i.e., Suppliers Declaration of Conformity (SDoC) or certification) as appropriate (e.g., Bluetooth and Wi-Fi transmitter modules may also contain digital logic functions).

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

45.1.1 Labeling and User Information Requirements

The WBZ450 module has been labeled with its own FCC ID number, and if the FCC ID is not visible when the module is installed inside another device, then the outside of the finished product into which the module is installed must display a label referring to the enclosed module. This exterior label must use the following wording:

> Contains Transmitter Module FCC ID: 2ADHKWBZ450
>
> or
>
> Contains FCC ID: 2ADHKWBZ450
>
> **This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation**.

The user's manual for the finished product must include the following statement:

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy, and if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna
- Increase the separation between the equipment and receiver
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- Consult the dealer or an experienced radio/TV technician for help

Additional information on labeling and user information requirements for Part 15 devices can be found in KDB Publication 784748, which is available at the FCC Office of Engineering and Technology (OET) Laboratory Division Knowledge Database (KDB) apps.fcc.gov/oetcf/kdb/index.cfm.

45.1.2 RF Exposure

All transmitters regulated by FCC must comply with RF exposure requirements. KDB 447498 General RF Exposure Guidance provides guidance in determining whether proposed or existing transmitting facilities, operations or devices comply with limits for human exposure to Radio Frequency (RF) fields adopted by the Federal Communications Commission (FCC).

From the FCC Grant: Output power listed is conducted. This grant is valid only when the module is sold to OEM integrators and must be installed by the OEM or OEM integrators. This transmitter is restricted for use with the specific antenna(s) tested in this application for Certification and must not be co-located or operating in conjunction with any other antenna or transmitters within a host device, except in accordance with FCC multi-transmitter product procedures.

WBZ450: These modules are approved for installation into mobile host platforms atleast 20cm away from the human body.

45.1.3 Helpful Web Sites

- Federal Communications Commission (FCC): www.fcc.gov.
- FCC Office of Engineering and Technology (OET) Laboratory Division Knowledge Database (KDB) apps.fcc.gov/oetcf/kdb/index.cfm.

45.2 Canada

The WBZ450 module has been certified for use in Canada under Innovation, Science and Economic Development Canada (ISED, formerly Industry Canada) Radio Standards Procedure (RSP) RSP-100, Radio Standards Specification (RSS) RSS-Gen and RSS-247. Modular approval permits the installation of a module in a host device without the need to recertify the device.

45.2.1 Labeling and User Information Requirements

Labeling Requirements (from RSP-100 - Issue 12, Section 5): The host product shall be properly labeled to identify the module within the host device.

The Innovation, Science and Economic Development Canada certification label of a module shall be clearly visible at all times when installed in the host device; otherwise, the host product must be labeled to display the Innovation, Science and Economic Development Canada certification number of the module, preceded by the word “Contains” or similar wording expressing the same meaning, as follows:

Contains IC: 20266-WBZ450

User Manual Notice for License-Exempt Radio Apparatus (from Section 8.4 RSS-Gen, Issue 5, February 2021): User manuals for license-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both:

This device contains license-exempt transmitter(s)/receiver(s) that comply with Innovation, Science and Economic Development Canada's license-exempt RSS(s). Operation is subject to the following two conditions:

(1) This device may not cause interference;

(2) This device must accept any interference, including interference that may cause undesired operation of the device.

L'émetteur/récepteur exempt de licence contenu dans le présent appareil est conforme aux CNR d'Innovation, Sciences et Développement économique Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes:

1. L'appareil ne doit pas produire de brouillage;

2. L'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Transmitter Antenna (From Section 6.8 RSS-GEN, Issue 5, February 2021): User manuals, for transmitters shall display the following notice in a conspicuous location:

This radio transmitter [IC: 20266-WBZ450] has been approved by Innovation, Science and Economic Development Canada to operate with the antenna types listed below, with the maximum permissible gain indicated. Antenna types not included in this list that have a gain greater than the maximum gain indicated for any type listed are strictly prohibited for use with this device.

Le présent émetteur radio [IC: 20266-WBZ450] a été approuvé par Innovation, Sciences et Développement économique Canada pour fonctionner avec les types d'antenne énumérés cidessous et ayant un gain admissible maximal. Les types d'antenne non inclus dans cette liste, et dont le gain est supérieur au gain maximal indiqué pour tout type figurant sur la liste, sont strictement interdits pour l'exploitation de l'émetteur.

Immediately following the above notice, the manufacturer shall provide a list of all antenna types approved for use with the transmitter, indicating the maximum permissible antenna gain (in dBi) and required impedance for each.

45.2.2 RF Exposure

All transmitters regulated by Innovation, Science and Economic Development Canada (ISED) must comply with RF exposure requirements listed in RSS-102 - Radio Frequency (RF) Exposure Compliance of Radiocommunication Apparatus (All Frequency Bands).

This transmitter is restricted for use with a specific antenna tested in this application for certification, and must not be co-located or operating in conjunction with any other antenna or transmitters within a host device, except in accordance with Canada multi-transmitter product procedures.

WBZ450: The device operates at an output power level which is within the ISED SAR test exemption limits at any user distance > 20cm.

Tous les émetteurs réglementés par Innovation, Sciences et Développement économique Canada (ISDE) doivent être conformes aux normes RF exigences d'exposition répertoriées dans RSS-102 - Conformité à l'exposition aux radiofréquences (RF) des radiocommunications Appareil (toutes les bandes de fréquence).

Cet émetteur est limité à une utilisation avec une antenne spécifique testée dans cette application pour la certification, et ne doit pas être co-localisé ou fonctionner en conjonction avec toute autre antenne ou émetteur dans un dispositif hôte, sauf dans conformément aux procédures des produits multi-émetteurs du Canada.

WBZ450 : L'appareil fonctionne à un niveau de puissance de sortie qui se situe dans les limites d'exemption de test ISED SAR à tout moment. distance utilisateur > 20 cm.

45.2.3 Helpful Web Sites

Innovation, Science and Economic Development Canada (ISED): www.ic.gc.ca/.

45.3 Europe

The WBZ450 module has is/are a Radio Equipment Directive (RED) assessed radio module that is CE marked and has been manufactured and tested with the intention of being integrated into a final product.

The WBZ450 module has has/have been tested to RED 2014/53/EU Essential Requirements mentioned in the following European Compliance table.

Table 45-1. European Compliance Information

Certification	Standard	Article
Safety	EN 62368	3.1a
Health	EN 62311	
EMC	EN 301 489-1	3.1b
	EN 301 489-17	
Radio	EN 300 328	3.2

The ETSI provides guidance on modular devices in the "*Guide to the application of harmonised standards covering articles 3.1b and 3.2 of the RED 2014/53/EU (RED) to multi-radio and combined radio and non-radio equipment*" document available at http://www.etsi.org/deliver/etsi_eg/203300_203399/20 3367/01.01.01_60/eg_203367v010101p.pdf.

Note: To maintain conformance to the standards listed in the preceding European Compliance table, the module shall be installed in accordance with the installation instructions in this data sheet and shall not be modified. When integrating a radio module into a completed product, the integrator becomes the manufacturer of the final product and is therefore responsible for demonstrating compliance of the final product with the essential requirements against the RED.

45.3.1 Labeling and User Information Requirements

The label on the final product that contains the WBZ450 module has must follow CE marking requirements.

45.3.2 Conformity Assessment

From ETSI Guidance Note EG 203367, section 6.1, when non-radio products are combined with a radio product:

If the manufacturer of the combined equipment installs the radio product in a host non-radio product in equivalent assessment conditions (i.e. host equivalent to the one used for the assessment of the radio product) and according to the installation instructions for the radio product, then no additional assessment of the combined equipment against article 3.2 of the RED is required.

45.3.2.1 Simplified EU Declaration of Conformity

Hereby, Microchip Technology Inc. declares that the radio equipment type WBZ450 is in compliance with Directive 2014/53/EU.

The full text of the EU declaration of conformity, for this product, is available at www.microchip.com/design-centers/wireless-connectivity/.

45.3.3 Helpful Websites

A document that can be used as a starting point in understanding the use of Short Range Devices (SRD) in Europe is the European Radio Communications Committee (ERC) Recommendation 70-03 E, which can be downloaded from the European Communications Committee (ECC) at: http://www.ecodocdb.dk/.

Additional helpful web sites are:

- Radio Equipment Directive (2014/53/EU): https://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/red_en
- European Conference of Postal and Telecommunications Administrations (CEPT): http://www.cept.org

- European Telecommunications Standards Institute (ETSI): http://www.etsi.org
- The Radio Equipment Directive Compliance Association (REDCA): http://www.redca.eu/

45.4 Japan

The WBZ450 module has has/have received type certification and is required to be labeled with its own technical conformity mark and certification number as required to conform to the technical standards regulated by the Ministry of Internal Affairs and Communications (MIC) of Japan pursuant to the Radio Act of Japan.

Integration of this module into a final product does not require additional radio certification provided installation instructions are followed and no modifications of the module are allowed. Additional testing may be required:

- If the host product is subject to electrical appliance safety (for example, powered from an AC mains), the host product may require Product Safety Electrical Appliance and Material (PSE) testing. The integrator should contact their conformance laboratory to determine if this testing is required
- There is an voluntary Electromagnetic Compatibility (EMC) test for the host product administered by VCCI: www.vcci.jp/vcci_e/index.html

45.4.1 Labeling and User Information Requirements

The label on the final product which contains the WBZ450 module has must follow Japan marking requirements. The integrator of the module should refer to the labeling requirements for Japan available at the Ministry of Internal Affairs and Communications (MIC) website.

For the WBZ450 module, due to a limited module size, the technical conformity logo and ID is displayed in the data sheet and/or packaging and cannot be displayed on the module label. The final product in which this module is being used must have a label referring to the type certified module inside:

45.4.2 Helpful Web Sites

- Ministry of Internal Affairs and Communications (MIC): www.tele.soumu.go.jp/e/index.htm.
- Association of Radio Industries and Businesses (ARIB): www.arib.or.jp/english/.

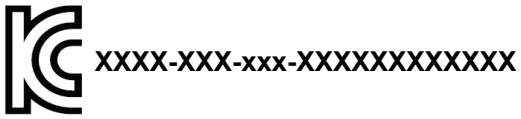
45.5 Korea

The WBZ450 module has has/have received certification of conformity in accordance with the Radio Waves Act. Integration of this module into a final product does not require additional radio certification provided installation instructions are followed and no modifications of the module are allowed.

45.5.1 Labeling and User Information Requirements

The label on the final product which contains the WBZ450 module has must follow KC marking requirements. The integrator of the module should refer to the labeling requirements for Korea available on the Korea Communications Commission (KCC) website.

The WBZ450 module is labeled with its own KC mark. The final product requires the KC mark and certificate number of the module:



45.5.2 Helpful Websites

- Korea Communications Commission (KCC): www.kcc.go.kr.
- National Radio Research Agency (RRA): rra.go.kr.

45.6 Taiwan

The WBZ450 module has has/have received compliance approval in accordance with the Telecommunications Act. Customers seeking to use the compliance approval in their product should contact Microchip Technology sales or distribution partners to obtain a Letter of Authority.

Integration of this module into a final product does not require additional radio certification provided installation instructions are followed and no modifications of the module are allowed.

45.6.1 Labeling and User Information Requirements

For the WBZ450 module, due to the limited module size, the NCC mark and ID are displayed in the data sheet only and cannot be displayed on the module label:

The user's manual should contain following warning (for RF device) in traditional Chinese:

根據 NCC LP0002 低功率射頻器材技術規範_章節 3.8.2：

取得審驗證明之低功率射頻器材，非經核准，公司、商號或使用者均不得擅自變更頻率、加大功率或變更原設計之特性及功能。

低功率射頻器材之使用不得影響飛航安全及干擾合法通信；經發現有干擾現象時，應立即停用，並改善至無干擾時方得繼續使用。

前述合法通信，指依電信管理法規定作業之無線電通信。

低功率射頻器材須忍受合法通信或工業、科學及醫療用電波輻射性電機設備之干擾。

45.6.2 Helpful Web Sites

National Communications Commission (NCC): www.ncc.gov.tw

45.7 Other Regulatory Information

- For information about other countries' jurisdictions not covered here, refer to the www.microchip.com/design-centers/wireless-connectivity/certifications.
- Should other regulatory jurisdiction certification be required by the customer, or the customer needs to recertify the module for other reasons, contact Microchip for the required utilities and documentation.

46. Document Revision History

Revision	Date	Section	Description
A	12/2022	Document	Initial revision

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded

by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet- Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, KoD, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN:

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820