Exhibit L: User Manual - Part 2 of 2

FCC ID: HN22011B-2

Code 128 Enumerations

```
typedef enum tagCode128Decoding
{
ITC_CODE128_NOTACTIVE = 0, // Default
ITC_CODE128_ACTIVE = 1,
ITC_CODE128_NO_CHANGE = 255
} ITC_CODE128_DECODING;
typedef enum tagEan128Identifier
{
ITC_EAN128_ID_REMOVE,
ITC_EAN128_ID_INCLUDE, // Default
ITC_EAN128_ID_NO_CHANGE = 255
} ITC_EAN128_IDENTIFIER;
typedef enum tagCode128Cip128
{
ITC_CODE128_CIP128_NOTACTIVE = 0, // Default
ITC_CODE128_CIP128_ACTIVE = 1,
ITC_CODE128_CIP128_NO_CHANGE = 255
} ITC_CODE128_CIP128;

#define ITC_CODE128_FNC1_NO_CHANGE 255.
This definition can be used when the Code128 FNC1 does not require any change.

#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar
code length does not require any change.
```

The table below shows what to be expected for EAN 128 labels for various symbology identifier transmit configurations and EAN 128 Identifier options.

Setup			**Application's Expected Result**	
EAN 128]C1 ID		**Symbology ID option**	**EAN 128 Label**	**Other Labels**
1	Include]C1	Disabled	<data>	<data>
2	Remove]C1	Disabled	<data>	<data>
3	Include]C1	AIM ID Transmitted	]C1<data>	]XY<data>
4	Remove]C1	AID ID Transmitted	]C1<data>	]XY<data>
5	Include]C1	Custom ID Transmitted	Z]C1<data>	Z<data>
6	Remove]C1	Custom ID Transmitted	Z<data>	Z<data>
where "X" is the symbology identifier, "Y" is the modifier character, and "Z" is the 1-byte symbology identifier.				

IS9CConfig::GetI2of5

This function retrieves the current settings of Interleaved 2 of 5.

Syntax

```
HRESULT IS9CConfig::GetI2of5( ITC_INTERLEAVED2OF5_DECODING*
peDecode, ITC_INTERLEAVED2OF5_CHECK_DIGIT* peCheck,
ITC_BARCODE_LENGTH_ID* peLengthId, BYTE rbgLengthBuff[],
DWORD* pdwNumBytes );
```

Parameters

peDecode	[out]	Pointer to the ITC_INTERLEAVED2OF5_DECODING location to receive the decoding for Interleaved 2 of 5 symbology.
peCheck	[out]	Pointer to the ITC_INTERLEAVED2OF5_CHECK_DIGIT location to receive the check digit.
peLengthId	[out]	Pointer to the ITC_BARCODE_LENGTH_ID location to receive an indicator of either ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.
rgbLengthBuff	[out,size_is(3)]	An array of bytes to receives 1 byte of data for ITC_BARCODE_LENGTH or 3 bytes of data for ITC_BARCODE_FIXED_LENGTH.
pdwNumBytes	[out]	Pointer to the DWORD location to receive a number indicating number of bytes in *rbgLengthBuff[]*: 1 byte for ITC_BARCODE_LENGTH or 3 bytes for ITC_BARCODE_FIXED_LENGTH.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig::SetI2of5

This function updates the Interleaved 2 of 5 settings with new values.

Syntax

```
HRESULT IS9CConfig::SetI2of5( ITC_INTERLEAVED2OF5_DECODING
eDecode, ITC_INTERLEAVED2OF5_CHECK_DIGIT eCheck,
ITC_BARCODE_LENGTH_ID eLengthId, BYTE rgbLengthBuff[], DWORD
dwNumBytes );
```

Parameters

eDecode	[in]	Identifies the decoding for Interleaved 2 of 5 symbology.
eCheck	[in]	Identifies the check digit.
eLengthId	[in]	Use ITC_BARCODE_LENGTH_NO_CHANGE to indicate no change for bar code length. Use ITC_BARCODE_LENGTH for any length and minimum length, and set *rgbLengthBuff[0]* to a valid length value. Use ITC_BARCODE_FIXED_LENGTH to compose 1 or 2 or 3 fixed lengths, and set 3 bytes: *rgbLengthBuff[0]*, *rgbLengthBuff[1]*, *rgbLengthBuff[2]* with valid values.
rgbLengthBuff	[in,size_is(dwNumBytes)]	Contains bar code lengths when *eLengthId* = Use ITC_BARCODE_LENGTH or Use ITC_BARCODE_FIXED_LENGTH.
dwNumBytes	[in]	Number of bytes in *rbgLengthBuff[]*. For S9C, this value is 1 when *eLengthId* = ITC_BARCODE_LENGTH or 3 when *eLengthId* = ITC_BARCODE_FIXED_LENGTH.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Interleaved 2 of 5 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_INTERLEAVED2OF5_DECODING
Check Digit	Not Used	ITC_INTERLEAVED2OF5_CHECK_DIGIT
Bar Code Length	Minimum Length = 6	0x00`0xFE ITC_BC_LENGTH_NO_CHANGE

Interleaved 2 of 5 Enumerations

```
typedef enum tagInterleaved2of5Decoding
{
ITC_INTERLEAVED2OF5_NOTACTIVE = 0,      // Default
ITC_INTERLEAVED2OF5_ACTIVE = 1,
ITC_INTERLEAVED2OF5_NO_CHANGE = 255
} ITC_INTERLEAVED2OF5_DECODING;
typedef enum tagInterleaved2of5CheckDigit
{
ITC_INTERLEAVED2OF5_CHECK_NOTUSED,      // Default
ITC_INTERLEAVED2OF5_CHECK_MOD10_XMIT,
ITC_INTERLEAVED2OF5_CHECK_MOD10_NOTXMIT,
ITC_INTERLEAVED2OF5_CHECK_FRENCH_CIP_XMIT,
ITC_INTERLEAVED2OF5_CHECK_FRENCH_CIP_NOTXMIT,
ITC_INTERLEAVED2OF5_CHECK_NO_CHANGE = 255
} ITC_INTERLEAVED2OF5_CHECK_DIGIT;
typedef enum tagBarcodeLengthId
{
ITC_BARCODE_LENGTH = 0,
ITC_BARCODE_FIXED_LENGTH,
ITC_BARCODE_LENGTH_NO_CHANGE = 255
} ITC_BARCODE_LENGTH_ID;
```

IS9CConfig::GetMatrix2of5

This function retrieves the current settings of Matrix 2 of 5.

Syntax

```
HRESULT IS9CConfig::GetMatrix2of5( ITC_MATRIX2OF5_DECODING*
peDecode, DWORD* pdwLength );
```

Parameters

peDecode	[out]	Pointer to the ITC_MATRIX2OF5_DECODING location to receive the decoding for Matrix 2 of 5 symbology.
pdwLength	[out]	Pointer to the DWORD location to receive a value for the bar code length.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig::SetMatrix2of5

This function updates the Matrix 2 of 5 settings with new values.

Syntax

HRESULT **IS9CConfig::SetMatrix2of5(** ITC_MATRIX2OF5_DECODING *eDecode*, DWORD *dwLength* **);**

Parameters

eDecode	[in]	Identifies the decoding for Matrix 2 of 5 symbology.
dwLength	[in]	Identifies the bar code length.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Matrix 2 of 5 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_MATRIX2OF5_DECODING
Bar Code Length	Minimum Length = 6	0x00`0xFE ITC_BC_LENGTH_NO_CHANGE

Matrix 2 of 5 Enumerations

```
typedef enum tagMatrix2of5Decoding
{
ITC_MATRIX2OF5_NOTACTIVE = 0, // Default
ITC_MATRIX2OF5_ACTIVE = 1,
ITC_MATRIX2OF5_NO_CHANGE = 255
} ITC_MATRIX2OF5_DECODING;
#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar
code length does not require any change.
```

IS9CConfig::GetMSI

This function retrieves the current MSI settings.

Syntax

```
HRESULT IS9CConfig::GetMSI( ITC_MSI_DECODING* peDecode,
ITC_MSI_CHECK_DIGIT* peCheck, DWORD* pdwLength );
```

Parameters

peDecode	[out]	Pointer to the ITC_MSI_DECODING location to receive the decoding for MSI symbology.
peCheck	[out]	Pointer to the ITC_MSI_CHECK_DIGIT location to receive the check digit.
pdwLength	[out]	Pointer to the DWORD location to receive the bar code length.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig::SetMSI

This function updates the MSI settings with new values.

Syntax

```
HRESULT IS9CConfig::SetMSI( ITC_MSI_DECODING eDecode,
ITC_MSI_CHECK_DIGIT eCheck, DWORD dwLength );
```

Parameters

eDecode	[in]	Identifies the decoding for MSI symbology.
eCheck	[in]	Identifies the check digit.
dwLength	[in]	Identifies the bar code length.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

MSI Default Settings

Parameter	Default	Valid Range	
Decoding	Not Active	ITC_MSI_DECODING	
Check Digit	MOD 10 checked and transmitted	ITC_MSI_CHECK_DIGIT	
Bar Code Length	Minimum Length = 6	0x00`0xFE	ITC_BC_LENGTH_NO_CHANGE

MSI Enumerations

```
typedef enum tagMsiDecoding
{
ITC_MSI_NOTACTIVE = 0, // Default
ITC_MSI_ACTIVE = 1,
ITC_MSI_NO_CHANGE = 255
} ITC_MSI_DECODING;
typedef enum tagMsiCheckDigit
{
ITC_MSI_CHECK_MOD10_XMIT, // Default
ITC_MSI_CHECK_MOD10_NOTXMIT,
ITC_MSI_CHECK_DOUBLEMOD10_XMIT,
ITC_MSI_CHECK_DOUBLEMOD10_NOTXMIT,
ITC_MSI_CHECK_NO_CHANGE = 255
} ITC_MSI_CHECK_DIGIT;
#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar
code length does not require any change.
```

IS9CConfig::GetPDF417

This function retrieves the current PDF417 settings.

Syntax

```
HRESULT IS9CConfig::GetPDF417( ITC_PDF417_DECODING*
pePdf417Decode, ITC_PDF417_MACRO_PDF* peMacroPdf,
ITC_PDF417_CTRL_HEADER* pePdfControlHeader,
ITC_PDF417_FILE_NAME* pePdfFileName,
ITC_PDF417_SEGMENT_COUNT* pePdfSegmentCount,
ITC_PDF417_TIME_STAMP* pePdfTimeStamp, ITC_PDF417_SENDER*
pePdfSender, ITC_PDF417_ADDRESSEE* pePdfAddressee,
ITC_PDF417_FILE_SIZE* pePdfFileSize, ITC_PDF417_CHECKSUM*
pePdfChecksum );
```

Parameters

pePdf417Decode	[out]	Pointer to the ITC_PDF417_DECODING location to receive the decoding for PDF417 symbology.
peMacroPdf	[out]	Pointer to the ITC_PDF417_MACRO_PDF location to receive the Macro PDF.
pePdfControlHeader	[out]	Pointer to the ITC_PDF417_CTRL_HEADER location to receive the control header.
pePdfFileName	[out]	Pointer to the ITC_PDF417_FILE_NAME location to receive the file name.
pePdfSegmentCount	[out]	Pointer to the ITC_PDF417_SEGMENT_COUNT location to receive the segment count.
pePdfTimeStamp	[out]	Pointer to the ITC_PDF417_TIME_STAMP location to receive the time stamp.

pePdfSender	[out]	Pointer to the ITC_PDF417_SENDER location to receive the sender.
pePdfAddressee	[out]	Pointer to the ITC_PDF417_ADDRESSEE location to receive the addressee.
pePdfFileSize	[out]	Pointer to the ITC_PDF417_FILE_SIZE location to receive the file size.
pePdfChecksum	[out]	Pointer to the ITC_PDF417_CHECKSUM location to receive the checksum.

Return Values
HRESULT that indicates success or failure.

Remarks
None.

See Also
None.

IS9CConfig::SetPDF417
This function updates the PDF417 settings with new values.

Syntax
```
HRESULT IS9CConfig::SetPDF417( ITC_PDF417_DECODING
ePdf417Decode, ITC_PDF417_MACRO_PDF eMacroPdf,
ITC_PDF417_CTRL_HEADER ePdfControlHeader,
ITC_PDF417_FILE_NAME ePdfFileName, ITC_PDF417_SEGMENT_COUNT
ePdfSegmentCount, ITC_PDF417_TIME_STAMP ePdfTimeStamp,
ITC_PDF417_SENDER ePdfSender, ITC_PDF417_ADDRESSEE
ePdfAddressee, ITC_PDF417_FILE_SIZE ePdfFileSize,
ITC_PDF417_CHECKSUM ePdfChecksum );
```

Parameters

ePdf417Decode	[in]	Identifies the decoding for PDF417 symbology.
eMacroPdf	[in]	Identifies the Macro PDF.
ePdfControlHeader	[in]	Identifies the control header.
ePdfFileName	[in]	Identifies the file name.
ePdfSegmentCount	[in]	Identifies the segment count.
ePdfTimeStamp	[in]	Identifies the time stamp.
ePdfSender	[in]	Identifies the sender.
ePdfAddressee	[in]	Identifies the addressee.
ePdfFileSize	[in]	Identifies the file size.
ePdfChecksum	[in]	Identifies the checksum.

Return Values
HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

PDF 417 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_PDF417_DECODING
Macro PDF	Macro PDF Buffered	ITC_PDF417_MACRO_PDF
Control Header	Not Transmitted	ITC_PDF417_CTRL_HEADER
*File Name	Not Transmitted	ITC_PDF417_FILE_NAME
*Segment Count	Not Transmitted	ITC_PDF417_SEGMENT_COUNT
*Time Stamp	Not Transmitted	ITC_PDF417_TIME_STAMP
*Sender	Not Transmitted	ITC_PDF417_SENDER
*Address	Not Transmitted	ITC_PDF417_ADDRESSEE
*File Size	Not Transmitted	ITC_PDF417_FILE_SIZE
*Check Sum	Not Transmitted	ITC_PDF417_CHECKSUM
These are Macro PDF Optional Fields.		

PDF 417 Enumerations

```
typedef enum tagPdf417Decoding
{
ITC_PDF417_NOTACTIVE = 0,
ITC_PDF417_ACTIVE = 1,               // Default
ITC_PDF417_NO_CHANGE = 255
} ITC_PDF417_DECODING;
typedef enum tagPdf417MacroPdf
{
ITC_PDF417_MACRO_UNBUFFERED = 0,
ITC_PDF417_MACRO_BUFFERED = 1,       // Default
ITC_PDF417_MACRO_NO_CHANGE = 255
} ITC_PDF417_MACRO_PDF;
typedef enum tagPdf417ControlHeader
{
ITC_PDF417_CTRL_HEADER_NOTXMIT = 0,      // Default
ITC_PDF417_CTRL_HEADER_XMIT = 1,
ITC_PDF417_CTRL_HEADER_NO_CHANGE = 255
} ITC_PDF417_CTRL_HEADER;
typedef enum tagPdf417FileName
{
ITC_PDF417_FILE_NAME_NOTXMIT = 0,    // Default
ITC_PDF417_FILE_NAME_XMIT = 1,
ITC_PDF417_FILE_NAME_NO_CHANGE = 255
} ITC_PDF417_FILE_NAME;
typedef enum tagPdf417SegmentCount
{
ITC_PDF417_SEGMENT_COUNT_NOTXMIT = 0, // Default
ITC_PDF417_SEGMENT_COUNT_XMIT = 1,
ITC_PDF417_SEGMENT_COUNT_NO_CHANGE = 255
} ITC_PDF417_SEGMENT_COUNT;

typedef enum tagPdf417TimeStamp
{
ITC_PDF417_TIME_STAMP_NOTXMIT = 0,   // Default
ITC_PDF417_TIME_STAMP_XMIT = 1,
ITC_PDF417_TIME_STAMP_NO_CHANGE = 255
} ITC_PDF417_TIME_STAMP;
typedef enum tagPdf417Sender
{
ITC_PDF417_SENDER_NOTXMIT = 0,       // Default
ITC_PDF417_SENDER_XMIT = 1,
ITC_PDF417_SENDER_NO_CHANGE = 255
} ITC_PDF417_SENDER;
typedef enum tagPdf417Addressee
{
ITC_PDF417_ADDRESSEE_NOTXMIT = 0,    // Default
ITC_PDF417_ADDRESSEE_XMIT = 1,
ITC_PDF417_ADDRESSEE_NO_CHANGE = 255
} ITC_PDF417_ADDRESSEE;
typedef enum tagPdf417FileSize
{
ITC_PDF417_FILE_SIZE_NOTXMIT = 0,    // Default
ITC_PDF417_FILE_SIZE_XMIT = 1,
ITC_PDF417_FILE_SIZE_NO_CHANGE = 255
} ITC_PDF417_FILE_SIZE;
typedef enum tagPdf417Checksum
{
ITC_PDF417_CHECKSUM_NOTXMIT = 0,        // Default
ITC_PDF417_CHECKSUM_XMIT = 1,
ITC_PDF417_CHECKSUM_NO_CHANGE = 255
} ITC_PDF417_CHECKSUM;
```

IS9CConfig::GetPlessey

This function retrieves the current Plessey settings.

Syntax

```
HRESULT IS9CConfig::GetPlessey( ITC_PLESSEY_DECODING*
peDecode, ITC_PLESSEY_CHECK_DIGIT* peCheck, DWORD* pdwLength
);
```

Parameters

peDecode	[out]	Pointer to the ITC_PLESSEY_DECODING location to receive the decoding for Plessey symbology.
peCheck	[out]	Pointer to the ITC_PLESSEY_CHECK_DIGIT location to receive the check digit.
pdwLength	[out]	Pointer to the DWORD location to receive the bar code length.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig::SetPlessey

This function updates the Plessey settings with new values.

Syntax

```
HRESULT IS9CConfig::SetPlessey( ITC_PLESSEY_DECODING
eDecode, ITC_PLESSEY_CHECK_DIGIT eCheck, DWORD dwLength );
```

Parameters

eDecode	[in]	Identifies the decoding for Plessey symbology.
eCheck	[in]	Identifies the check digit.
dwLength	[in]	Identifies the bar code length.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Plessey Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_PLESSEY_DECODING
Check Digit	Not Transmitted	ITC_PLESSEY_CHECK_DIGIT
Bar Code Length	Any Bar Code Length	0x00`0xFE ITC_BC_LENGTH_NO_CHANGE

Plessey Enumerations

```
typedef enum tagPlesseyDecoding
{
ITC_PLESSEY_NOTACTIVE = 0,           // Default
ITC_PLESSEY_ACTIVE = 1,
ITC_PLESSEY_NO_CHANGE = 255
} ITC_PLESSEY_DECODING;
typedef enum tagPlesseyCheckDigit
{
ITC_PLESSEY_CHECK_NOTXMIT = 0,    // Default
ITC_PLESSEY_CHECK_XMIT = 1,
ITC_PLESSEY_CHECK_NO_CHANGE = 255
} ITC_PLESSEY_CHECK_DIGIT;
#define ITC_BC_LENGTH_NO_CHANGE 255. This definition can be used when the bar
code length does not require any change.
```

IS9CConfig::GetStandard2of5

This function retrieves the current Standard 2 of 5 settings.

Syntax

```
HRESULT IS9CConfig::GetStandard2of5(
ITC_STANDARD2OF5_DECODING* peDecode,
ITC_STANDARD2OF5_FORMAT* peFormat,
ITC_STANDARD2OF5_CHECK_DIGIT* peCheck,
ITC_BARCODE_LENGTH_ID* peLengthId, BYTE rgbLengthBuff,
DWORD* pdwNumBytes );
```

Parameters

peDecode	[out]	Pointer to the ITC_STANDARD2OF5_DECODING location to receive the decoding for Standard 2 of 5 symbology.
peFormat	[out]	Pointer to the ITC_STANDARD2OF5_FORMAT location to receive the format.
peCheck	[out]	Pointer to the ITC_STANDARD2OF5_CHECK_DIGIT location to receive Modulo 10 check digit.
peLengthId	[out]	Pointer to the ITC_BARCODE_LENGTH_ID location to receive an indicator of either ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.
rgbLengthBuff	[out,size_is(3)]	An array of bytes to receives 1 byte of data for ITC_BARCODE_LENGTH, or 3 bytes of data for ITC_BARCODE_FIXED_LENGTH.
pdwNumBytes	[out]	Pointer to the DWORD location to receive a number indicating number of bytes in *rbgLengthBuff[]*: 1 byte for ITC_BARCODE_LENGTH or 3 bytes for ITC_BARCODE_FIXED_LENGTH.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig::SetStandard2of5

This function updates the Standard 2 of 5 settings with new values.

Syntax

```
HRESULT IS9CConfig::SetStandard2of5(
ITC_STANDARD2OF5_DECODING eDecode, ITC_STANDARD2OF5_FORMAT
eFormat, ITC_STANDARD2OF5_CHECK_DIGIT eCheck,
ITC_BARCODE_LENGTH_ID eLengthId, BYTE rgbLengthBuff[], DWORD
dwNumBytes );
```

Parameters

eDecode	[in]	Identifies the decoding for Standard 2 of 5 symbology.
eFormat	[in]	Identifies the format.
eCheck	[in]	Identifies the Modulo 10 check digit.
eLengthId	[in]	Use ITC_BARCODE_LENGTH_NO_CHANGE to indicate no change for bar code length. Use ITC_BARCODE_LENGTH for any length and minimum length, and set rgbLengthBuff[0] to a valid length value. Use ITC_BARCODE_FIXED_LENGTH to compose 1 or 2 or 3 fixed lengths, and set 3 bytes: *rgbLengthBuff[0]*, *rgbLengthBuff[1]*, *rgbLengthBuff[2]* with valid values.
rgbLengthBuff	[in,size_is(dwNumBytes)]	An array of bytes containing bar code lengths when *eLengthId* = ITC_BARCODE_LENGTH or ITC_BARCODE_FIXED_LENGTH.
dwNumBytes	[in]	Number of bytes in *rbgLengthBuff[]*. For S9C, this value is 1 when *eLengthId* = ITC_BARCODE_LENGTH or 3 when *eLengthId* = ITC_BARCODE_FIXED_LENGTH.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Standard 2 of 5 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_STANDARD2OF5_DECODING
Format	Identicon (6 Start/Stop bars)	ITC_STANDARD2OF5_FORMAT
Check Digit	Not Used	ITC_STANDARD2OF5_CHECK_DIGIT
Bar Code Length	Minimum Length = 6	0x00-0xFE ITC_BC_LENGTH_NO_CHANGE

Standard 2 of 5 Enumerations

```
typedef enum tagStandard2of5Decoding
{
ITC_STANDARD2OF5_NOTACTIVE = 0, // Default
ITC_STANDARD2OF5_ACTIVE = 1,
ITC_STANDARD2OF5_NO_CHANGE = 255
} ITC_STANDARD2OF5_DECODING;
typedef enum tagStandard2of5Format
{
ITC_STANDARD2OF5_FORMAT_IDENTICON, // Default
ITC_STANDARD2OF5_FORMAT_COMPUTER_IDENTICS,
ITC_STANDARD2OF5_FORMAT_NO_CHANGE = 255
} ITC_STANDARD2OF5_FORMAT;
typedef enum tagStandard2of5CheckDigit
{
ITC_STANDARD2OF5_CHECK_NOTUSED, // Default
ITC_STANDARD2OF5_CHECK_XMIT,
ITC_STANDARD2OF5_CHECK_NOTXMIT,
ITC_STANDARD2OF5_CHECK_NO_CHANGE = 255
} ITC_STANDARD2OF5_CHECK_DIGIT;
typedef enum tagBarcodeLengthId
{
ITC_BARCODE_LENGTH = 0,
ITC_BARCODE_FIXED_LENGTH,
ITC_BARCODE_LENGTH_NO_CHANGE = 255
} ITC_BARCODE_LENGTH_ID;
```

IS9CConfig::GetTelepen

This function retrieves the current Telepen settings.

Syntax

```
HRESULT IS9CConfig::GetTelepen( ITC_TELEPEN_DECODING*
peDecode, ITC_TELEPEN_FORMAT* peFormat );
```

Parameters

peDecode	[out]	Pointer to the ITC_TELEPEN_DECODING location to receive the decoding for TELEPEN symbology.
peFormat	[out]	Pointer to the ITC_TELEPEN_FORMAT location to receive the format.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig::SetTelepen

This function updates the Telepen settings with new values.

Syntax

```
HRESULT IS9CConfig::SetTelepen( ITC_TELEPEN_DECODING*
eDecode, ITC_TELEPEN_FORMAT* eFormat );
```

Parameters

eDecode	[in]	Identifies the decoding for Telepen symbology.
eFormat	[in]	Identifies the format.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Telepen Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_TELEPEN_DECODING
Format	ASCII	ITC_TELEPEN_FORMAT

Telepen Enumerations

```
typedef enum tagTelepenDecoding
{
ITC_TELEPEN_NOTACTIVE = 0, // Default
ITC_TELEPEN_ACTIVE = 1,
ITC_TELEPEN_NO_CHANGE = 255
} ITC_TELEPEN_DECODING;
typedef enum tagTelepenDecoding
{
ITC_TELEPEN_FORMAT_ASCII, // Default
ITC_TELEPEN_FORMAT_NUMERIC,
ITC_TELEPEN_FORMAT_NO_CHANGE = 255
} ITC_TELEPEN_FORMAT;
```

IS9CConfig::GetUpcEan

This function retrieves the current UPC/EAN settings.

Syntax

```
HRESULT IS9CConfig::GetUpcEan( ITC_UPCEAN_DECODING*
upceanDecode, ITC_UPCA_SELECT* upcASelect, ITC_UPCE_SELECT*
upcESelect, ITC_EAN8_SELECT* ean8Select, ITC_EAN13_SELECT*
ean13Select, ITC_UPCEAN_ADDON_DIGITS* upcAddOnDigits,
ITC_UPCEAN_ADDON_TWO* upcAddOn2, ITC_UPCEAN_ADDON_FIVE*
upcAddOn5, ITC_UPCA_CHECK_DIGIT* upcACheck,
ITC_UPCE_CHECK_DIGIT* upcECheck, ITC_EAN8_CHECK_DIGIT*
ean8Check, ITC_EAN13_CHECK_DIGIT* ean13Check,
ITC_UPCA_NUMBER_SYSTEM* upcANumSystem,
ITC_UPCE_NUMBER_SYSTEM* upcENumSystem, ITC_UPCA_REENCODE*
upcAReencode, ITC_UPCE_REENCODE* upcEReencode,
ITC_EAN8_REENCODE* ean8Reencode );
```

Parameters

upceanDecode	[out]	Pointer to the ITC_UPCEAN_DECODING location to receive the decoding for UPC/EAN symbology.
upcASelect	[out]	Pointer to the ITC_UPCA_SELECT location to receive the UPC-A selection state.
upcESelect	[out]	Pointer to the ITC_UPCE_SELECT location to receive the UPC-E selection state.
ean8Select	[out]	Pointer to the ITC_EAN8_SELECT location to receive the EAN-8 selection state.
ean13Select	[out]	Pointer to the ITC_EAN13_SELECT location to receive the EAN-13 selection state.
upcAddOnDigits	[out]	Pointer to the ITC_UPCEAN_ADDON_DIGITS location to receive the add-on digits.
upcAddOn2	[out]	Pointer to the ITC_UPCEAN_ADDON_TWO location to receive the add-on 2 digits.
upcAddOn5	[out]	Pointer to the ITC_UPCEAN_ADDON_FIVE location to receive the add-on 5 digits.

upcACheck	[out]	Pointer to the ITC_UPCA_CHECK_DIGIT location to receive the UPC-A check digit.
upcECheck	[out]	Pointer to the ITC_UPCE_CHECK_DIGIT location to receive the UPC-E check digit.
ean8Check	[out]	Pointer to the ITC_EAN8_CHECK_DIGIT location to receive the EAN-8 check digit.
ean13Check	[out]	Pointer to the ITC_EAN13_CHECK_DIGIT location to receive the EAN-13 check digit.
upcANumSystem	[out]	Pointer to the ITC_UPCA_NUMBER_SYSTEM location to receive the UPC-A number system.
upcENumSystem	[out]	Pointer to the ITC_UPCE_NUMBER_SYSTEM location to receive the UPC-E number system.
upcAReencode	[out]	Pointer to the ITC_UPCA_REENCODE location to receive the UPC-A reencoding.
upcEReencode	[out]	Pointer to the ITC_UPCE_REENCODE location to receive the UPC-E reencoding.
ean8Reencode	[out]	Pointer to the ITC_EAN8_REENCODE location to receive the EAN-8 reencoding.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig::SetUpcEan

This function updates the UPC/EAN settings with new values.

Syntax

```
HRESULT IS9CConfig::SetUpcEan( ITC_UPCEAN_DECODING
upceanDecode, ITC_UPCA_SELECT upcASelect, ITC_UPCE_SELECT
upcESelect, ITC_EAN8_SELECT ean8Select, ITC_EAN13_SELECT
ean13Select, ITC_UPCEAN_ADDON_DIGITS upcAddOnDigits,
ITC_UPCEAN_ADDON_TWO upcAddOn2, ITC_UPCEAN_ADDON_FIVE
upcAddOn5, ITC_UPCA_CHECK_DIGIT upcACheck,
ITC_UPCE_CHECK_DIGIT upcECheck, ITC_EAN8_CHECK_DIGIT
ean8Check, ITC_EAN13_CHECK_DIGIT ean13Check,
ITC_UPCA_NUMBER_SYSTEM upcANumSystem, ITC_UPCE_NUMBER_SYSTEM
upcENumSystem, ITC_UPCA_REENCODE upcAReencode,
ITC_UPCE_REENCODE upcEReencode, ITC_EAN8_REENCODE
ean8Reencode );
```

Parameters

upceanDecode	[in]	Identifies the decoding for UPC/EAN symbology.
upcASelect	[in]	Identifies the UPC-A selection state.
upcESelect	[in]	Identifies the UPC-E selection state.
ean8Select	[in]	Identifies the EAN-8 selection state.
ean13Select	[in]	Identifies the EAN-13 selection state.
upcAddOnDigits	[in]	Identifies the Add-on digits.
upcAddOn2	[in]	Identifies the Add-on 2 digits.
upcAddOn5	[in]	Identifies the Add-on 5 digits.
upcACheck	[in]	Identifies the UPC-A check digit.
upcECheck	[in]	Identifies the UPC-E check digit.
ean8Check	[in]	Identifies the EAN-8 check digit.
ean13Check	[in]	Identifies the EAN-13 check digit.
upcANumSystem	[in]	Identifies the UPC-A number system.
upcENumSystem	[in]	Identifies the UPC-E number system.
upcAReencode	[in]	Identifies the UPC-A reencoding.
upcEReencode	[in]	Identifies the UPC-E reencoding.
ean8Reencode	[in]	Identifies the EAN-8 reencoding.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

UPC/EAN Default Settings

Parameter	Default	Valid Range
Decoding	ITC_UPCEAN_NO_CHANGE	This parameter is no longer used, set it to this value.
UPC-A	Active	ITC_UPCA_SELECT
UPC-E	Active	ITC_UPCE_SELECT
EAN-8	Active	ITC_EAN8_SELECT
EAN-13	Active	ITC_EAN13_SELECT
Add On Digits	Not Required	ITC_UPCEAN_ADDON_DIGITS
Add On 2 Digits	Not Active	ITC_UPCEAN_ADDON_TWO
Add On 5 Digits	Not Active	ITC_UPCEAN_ADDON_FIVE
UPC-A Check Digit	Transmitted	ITC_UPCA_CHECK_DIGIT
UPC-E Check Digit	Transmitted	ITC_UPCE_CHECK_DIGIT
EAN-8 Check Digit	Transmitted	ITC_EAN8_CHECK_DIGIT
EAN-13 Check Digit	Transmitted	ITC_EAN13_CHECK_DIGIT
UPC-A Number System	Transmitted	ITC_UPCA_NUMBER_SYSTEM
UPC-E Number System	Transmitted	ITC_UPCE_NUMBER_SYSTEM
Reencode UPC-A	UPC-A transmitted as EAN-13	ITC_UPCA_REENCODE
Reencode UPC-E	UPC-E transmitted as UPC-E	ITC_UPCE_REENCODE
Reencode EAN-8	EAN-8 transmitted as EAN-8	ITC_EAN8_REENCODE

UPC/EAN Enumerations

```
typedef enum tagUpcEanDecoding
{
ITC_UPCEAN_NOTACTIVE = 0,
ITC_UPCEAN_ACTIVE = 1,              // Default
ITC_UPCEAN_NO_CHANGE = 255
} ITC_UPCEAN_DECODING;
typedef enum tagUpcASelect
{
ITC_UPCA_DEACTIVATE,
ITC_UPCA_ACTIVATE,                  // Default
ITC_UPCA_NO_CHANGE = 255
} ITC_UPCA_SELECT;
typedef enum tagUpcESelect
{
ITC_UPCE_DEACTIVATE,
ITC_UPCE_ACTIVATE,                  // Default
ITC_UPCE_NO_CHANGE = 255
} ITC_UPCE_SELECT;
typedef enum tagEan8Select
{
ITC_EAN8_DEACTIVATE,
ITC_EAN8_ACTIVATE,                  // Default
ITC_EAN8_NO_CHANGE = 255
} ITC_EAN8_SELECT;
typedef enum tagEan13Select
{
ITC_EAN13_DEACTIVATE,
ITC_EAN13_ACTIVATE,                      // Default
ITC_EAN13_NO_CHANGE = 255
} ITC_EAN13_SELECT;
typedef enum tagUpcEanAddonDigits
{
ITC_UPCEAN_ADDON_NOT_REQUIRED,           // Default
ITC_UPCEAN_ADDON_REQUIRED,
ITC_UPCEAN_ADDON_NO_CHANGE = 255
} ITC_UPCEAN_ADDON_DIGITS;
typedef enum tagUpcEanAddonTwo
{
ITC_UPCEAN_ADDON_TWO_NOTACTIVE = 0,      // Default
ITC_UPCEAN_ADDON_TWO_ACTIVE = 1,
ITC_UPCEAN_ADDON_TWO_NO_CHANGE = 255
} ITC_UPCEAN_ADDON_TWO;
typedef enum tagUpcEanAddonFive
{
ITC_UPCEAN_ADDON_FIVE_NOTACTIVE = 0,     // Default
ITC_UPCEAN_ADDON_FIVE_ACTIVE = 1,
ITC_UPCEAN_ADDON_FIVE_NO_CHANGE = 255
} ITC_UPCEAN_ADDON_FIVE;
typedef enum tagUpcACheckDigit
{
ITC_UPCA_CHECK_NOTXMIT = 0,
ITC_UPCA_CHECK_XMIT = 1,                 // Default
ITC_UPCA_CHECK_NO_CHANGE = 255
} ITC_UPCA_CHECK_DIGIT;
typedef enum tagUpcECheckDigit
{
ITC_UPCE_CHECK_NOTXMIT = 0,
ITC_UPCE_CHECK_XMIT = 1,                 // Default
ITC_UPCE_CHECK_NO_CHANGE = 255
} ITC_UPCE_CHECK_DIGIT;
typedef enum tagEan8CheckDigit
{
ITC_EAN8_CHECK_NOTXMIT = 0,
ITC_EAN8_CHECK_XMIT = 1,                 // Default
ITC_EAN8_CHECK_NO_CHANGE = 255
} ITC_EAN8_CHECK_DIGIT;
typedef enum tagEan13CheckDigit
{
ITC_EAN13_CHECK_NOTXMIT = 0,
ITC_EAN13_CHECK_XMIT = 1,                // Default
ITC_EAN13_CHECK_NO_CHANGE = 255
} ITC_EAN13_CHECK_DIGIT;
typedef enum tagUpcANumberSystem
{
ITC_UPCA_NUM_SYS_NOTXMIT = 0,
ITC_UPCA_NUM_SYS_XMIT = 1,               // Default
ITC_UPCA_NUM_SYS_NO_CHANGE = 255
} ITC_UPCA_NUMBER_SYSTEM;
typedef enum tagUpcENumberSystem
{
ITC_UPCE_NUM_SYS_NOTXMIT = 0,
ITC_UPCE_NUM_SYS_XMIT = 1,               // Default
ITC_UPCE_NUM_SYS_NO_CHANGE = 255
} ITC_UPCE_NUMBER_SYSTEM;
typedef enum tagUpcAReencode
{
```

```
ITC_UPCA_XMIT_AS_EAN13,                 // Default
ITC_UPCA_XMIT_AS_UPCA,
ITC_UPCA_XMIT_NO_CHANGE = 255
} ITC_UPCA_REENCODE;
typedef enum tagUpcEReencode
{
ITC_UPCE_XMIT_AS_UPCE,                  // Default
ITC_UPCE_XMIT_AS_UPCA,
ITC_UPCE_XMIT_NO_CHANGE = 255
} ITC_UPCE_REENCODE;
typedef enum tagEan8Reencode
{
ITC_EAN8_XMIT_AS_EAN8,                  //Default
ITC_EAN8_XMIT_AS_EAN13,
ITC_EAN8_XMIT_NO_CHANGE = 255
} ITC_EAN8_REENCODE;
```

IS9CConfig2 Functions

This interface is derived from the IS9CConfig interface and provides additional methods that can be used to set and retrieve the 700 Series Computer' s bar code configuration. All supported symbologies are initialized to their defaults when the S9C firmware is loaded.

GET/SET functions use enumerations as their parameters. In most enumerations, there is an enumerator xx_NO_CHANGE (such as ITC_CODE39_NO_CHANGE), where xx refers to a particular enumeration. This enumerator can be used during a call to a SET to indicate that no change is to be made to that particular parameter. This prevents the called function from having to format the same S9C command and send it down to the scanner.

To specify a bar code length of "any length," use a value of "0" for the bar code length argument.

IS9CConfig2 functions are the following. IS9CCONFIG.H is the header file and ITCUUID.LIB contains the IID_IADC Interface GUID value used to obtain the interface.

- IS9CConfig2::GetCode11 *(page 205)*
- IS9CConfig2::SetCode11 *(page 205)*
- IS9CConfig2::GetCustomSymIds *(page 207)*
- IS9CConfig2::SetCustomSymIds *(page 208)*
- IS9CConfig2::GetGlobalAmble *(page 211)*
- IS9CConfig2::SetGlobalAmble *(page 212)*
- IS9CConfig2::GetPDF417Ext *(page 213)*
- IS9CConfig2::SetPDF417Ext *(page 213)*
- IS9CConfig2::GetSymIdXmit *(page 214)*
- IS9CConfig2::SetSymIdXmit *(page 214)*

IS9CConfig2::GetCode11

This function retrieves the current settings for Code 11.

Syntax

```
HRESULT GetCode11( ITC_CODE11_DECODING* peDecode,
ITC_CODE11_CHECK_DIGIT* peCheck,
ITC_CODE11_CHECK_VERIFICATION* peVer );
```

Parameters

peDecode	[out]	Pointer to ITC_CODE11_DECODING location to receive Code 11 decoding.
peCheck	[out]	Pointer to ITC_CODE11_CHECK_DIGIT location to receive the check digit option.
peVer	[out]	Pointer to ITC_CODE11_CHECK_VERIFICATION location to receive the check verification option.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig2::SetCode11

This function updates the current setting of Code 11 symbology.

Syntax

```
HRESULT SetCode11( ITC_CODE11_DECODING eDecode,
ITC_CODE11_CHECK_DIGIT eCheck, ITC_CODE11_CHECK_VERIFICATION
eVer );
```

Parameters

eDecode	[in]	An enumeration that identifies decoding option for Code 11.
eCheck	[in]	An enumeration that identifies the check digit option.
eVer	[in]	An enumeration that identifies check verification option.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Code 11 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_CODE11_DECODING
Check Verification	1 Digit	ITC_CODE11_CHECK_VERIFICATION
Check Digit	Enable	ITC_CODE11_CHECK_DIGIT

Code 11 Enumerations

```
typedef enum tagCode11Decoding
{
ITC_CODE11_NOTACTIVE = 0,
ITC_CODE11_ACTIVE = 1, // Default
ITC_CODE11_NO_CHANGE = 255
} ITC_CODE11_DECODING;
typedef enum tagCode11CheckVerification
{
ITC_CODE11_CHK_VERIFY_ONEDIGIT = 1,
ITC_CODE11_CHK_VERIFY_TWODIGIT = 2, // Default
ITC_CODE11_CHK_VERIFY_NO_CHANGE = 255
} ITC_CODE11_CHECK_VERIFICATION;
typedef enum tagCode11CheckDigit
{
ITC_CODE11_CHECK_NOTXMIT = 0, // Default
ITC_CODE11_CHECK_XMIT = 1,
ITC_CODE11_CHECK_NO_CHANGE = 255
} ITC_CODE11_CHECK_DIGIT;
```

IS9CConfig2::GetCustomSymIds

This function retrieves all the custom symbology identifiers defined for the currently supported symbologies. *This is not supported when using an imager on the 700 Series Computer.*

Syntax

```
HRESULT GetCustomSymIds( ITC_CUST_SYM_ID_PAIR*
pStructSymIdPair,DWORD dwMaxNumElement, DWORD* pdwNumElement
);
```

Parameters

pStructSymIdPair	[out]	Pointer to ITC_CUST_SYM_ID_PAIR location to receive the current defined symbology identifiers for the supported symbologies. The caller must preallocate this buffer with *dwMaxNumElement* elements.
dwMaxNumElement	[in]	Maximum number of elements allocated for the *pStructSymIdPair* buffer which should always be equal to the last defined enumeration constant + 1 of the enumeration ITC_CUSTOM_ID. In this case, it is ITC_CUSTOMID_LAST_ELEMENT.
pdwNumElement	[out]	Pointer to DWORD location to receive the actual number of elements returned in the *pStructSymIdPair* buffer, which should be the same as *dwMaxNumElement*.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

- Custom Identifier Assignments *(page 209)*
- Custom Identifier Example *(page 210)*
- Custom Identifier Default Settings *(page 210)*

IS9CConfig2::SetCustomSymIds

This function updates the symbology identifiers (any ASCII values) for the currently supported symbologies. *This is not supported when using an imager on the 700 Series Computer.*

Syntax

```
HRESULT SetCustomSymIds( ITC_CUST_SYM_ID_PAIR*
pStructSymIdPair, DWORD dwNumElement );
```

Parameters

pStructSymIdPair	[in]	Pointer to ITC_CUST_SYM_ID_PAIR location, containing the new symbology identifiers for any supported symbologies to update.
dwNumElement	[in]	Identifies the number of symbology identifiers to update in the *pStructSymIdPair* buffer.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Custom Identifier Assignments

Each custom identifier is a one byte ASCII value within the range from 0x00 to 0xff. The enumerations in the ITC_CUSTOM_ID enumerator can be used as symbology identifications in the GetCustomSymIds() and SetCustomSymIds() functions.

```
typedef enum tagCustomId
{
ITC_CUSTOMID_CODABAR = 0       Identifies the Codabar symbology
ITC_CUSTOMID_CODE39            Identifies the Code 39 symbology
ITC_CUSTOMID_CODE93            Identifies the Code 93 symbology
ITC_CUSTOMID_CODE128_EAN_128   Identifies the Code 128 symbology
ITC_CUSTOMID_EAN8              Identifies the EAN-8 symbology
ITC_CUSTOMID_EAN13             Identifies the EAN-13 symbology
ITC_CUSTOMID_I2OF5             Identifies the Interleaved 2 of 5 symbology
ITC_CUSTOMID_MATRIX2OF5        Identifies the Matrix 2 of 5 symbology
ITC_CUSTOMID_MSI               Identifies the MSI symbology
ITC_CUSTOMID_PDF417            Identifies the PDF 417 symbology
ITC_CUSTOMID_PLESSEY           Identifies the Plessey symbology
ITC_CUSTOMID_CODE2OF5          Identifies the Standard 2 of 5 symbology
ITC_CUSTOMID_TELEPEN           Identifies the Telepen symbology
ITC_CUSTOMID_UPCA              Identifies the UPC-A symbology
ITC_CUSTOMID_UPCE              Identifies the UPC-E symbology
ITC_CUSTOMID_CODE11            Identifies the Code 11 symbology
ITC_CUSTOMID_LAST_ELEMENT      Identifies the last element. Use to preallocate
the buffer on GetCustomSymIds
}ITC_CUSTOM_ID;
typedef struct tagCustSymbIdPair
{

ITC_CUSTOM_ID eSymbology;      Identifies the symbology of interest

BYTE byteId;
 ASCII value (1 byte within the range0x00 – 0xf)

}ITC_CUST_SYM_ID_PAIR;
```

Custom Identifier Default Settings

Symbology	Default	Valid Range
Codabar	D	0x00-0xFF
Code 11	*	0x00-0xFF
Code 39	*	0x00-0xFF
Code 93	D	0x00-0xFF
Code128/EAN 128	D	0x00-0xFF
EAN-8	0xFF	0x00-0xFF
EAN-13	F	0x00-0xFF
Interleaved 2 of 5	I	0x00-0xFF
Matrix 2 of 5	D	0x00-0xFF
MSI	D	0x00-0xFF
PDF 417	*	0x00-0xFF
Plessey	D	0x00-0xFF
Standard 2 of 5	D	0x00-0xFF
Telepen	*	0x00-0xFF
UPC-A	A	0x00-0xFF
UPC-E	E	0x00-0xFF

Custom Identifier Example

The following code segment is an example of updating the UPC-E and UPC-A symbology identifiers with new values, and then retrieving the currently defined symbology identifiers for all the supported symbologies:

```
ITC_CUST_SYM_ID_PAIR oStructSymIdPair [ITC_CUSTOMID_LAST_ELEMENT];
oStructSymIdPair[0].eSymbology = ITC_CUSTOMID_UPCE;
oStructSymIdPair[0].byteId = 0x41;           // ASCII char A
oStructSymIdPair[1].eSymbology = ITC_CUSTOMID_UPCA;
oStructSymIdPair[1].byteId = 0x42;           // ASCII char B
HRESULT hr = pIS9CConfig2->SetCustomSymIds(&oStructSymIdPair[0], 2];
DWORD dwNum = 0;
HRESULT hr = pIS9CConfig2->GetCustomSymIds(&oStructSymIdPair[0],
ITC_CUSTOMID_LAST_ELEMENT, &dwNum);
```

IS9CConfig2::GetGlobalAmble

This retrieves the scanner' s current preamble or postamble setting.

Syntax

```
HRESULT GetGlobalAmble( ITC_GLOBAL_AMBLE_ID eAmbleId, BYTE
rgbBuffer[], DWORD dwBufferSize, DWORD* pdwBufferSize );
```

Parameters

eAmbleId	[in]	An enumeration of type ITC_GLOBAL_AMBLE_ID identifies whether the preamble or postamble setting is to be retrieved. Only one setting can be queried at a time.
rgbBuffer	[in]	Contains the buffer for the postamble or preamble setting to be queried.
dwBufferSize	[in]	The maximum number of bytes that rgbBuffer can store. Must be at least ITC_GLOBAL_AMBLE_MAX_CHARS bytes.
pdwBufferSize	[out]	A pointer to DWORD location to store the actual number of returned bytes in *rgbBuffer*.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig2::SetGlobalAmble

This function updates the scanner' s current preamble or postamble setting depending on the input parameters.

Syntax

```
HRESULT SetGlobalAmble( ITC_GLOBAL_AMBLE_ID eAmbleId, BYTE
rgbBuffer[], DWORD dwBufferSize );
```

Parameters

eAmbleId	[in]	An enumeration of type ITC_GLOBAL_AMBLE_ID identifies whether the preamble or postamble setting is to be updated. Only one setting can be updated at a time.
rgbBuffer	[in]	Contains the buffer for the postamble or preamble setting to be updated.
dwBufferSize	[in]	Identifies number of bytes in *rgbBuffer*.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Postamble and Preamble Defaults

Parameter	Default	Valid Range
Preamble	Null	0 to 20 ASCII characters
Postamble	Null	0 to 20 ASCII characters

IS9CConfig2::GetPDF417Ext

This function is an extended function for retrieving the PDF 417 settings not included in the IS9CConfig::GetPDF417.

Syntax

```
HRESULT GetPDF417Ext( ITC_MICRO_PDF417_DECODING* peDecode,
ITC_MICRO_PDF417_CODE128_EMULATION* peCode128 );
```

Parameters

peDecode	[out]	Pointer to ITC_MICRO_PDF417_DECODING location to receive the Micro PDF 417 decoding.
peCode128	[out]	Pointer to ITC_MICRO_PDF417_CODE128_EMULATION* location to receive the Micro PDF 417 Code 128 emulation option.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig2::SetPDF417Ext

This function is an extended function for updating the additional PDF 417 settings not included in IS9CConfig::SetPDF417.

Syntax

```
HRESULT SetPDF417Ext( ITC_MICRO_PDF417_DECODING eDecode,
ITC_MICRO_PDF417_CODE128_EMULATION eCode128 );
```

Parameters

eDecode	[in]	An enumeration that identifies decoding option for the Micro PDF 417.
eCode128	[in]	An enumeration that identifies the Code 128 emulation option for the Micro PDF 417.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

PDF 417 Extended: Micro PDF 417 Default Settings

Parameter	Default	Valid Range
Decoding	Not Active	ITC_MICRO_PDF417_DECODING
Code 128 Emulation	Not Active	ITC_MICRO_PDF417_CODE128_EMULATION
** These are Micro PDF 417 parameters.*		

IS9CConfig2::GetSymIdXmit

This function retrieves the current symbology ID transmission option as described on the next page.

Syntax

```
HRESULT GetSymIdXmit( ITC_SYMBOLOGY_ID_XMIT* peSymIdXmit );
```

Parameters

peSymIdXmit	[out]	Pointer to ITC_SYMBOLOGY_ID_XMIT location to receive the current symbology identifier transmission option.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

IS9CConfig2::SetSymIdXmit

This updates the symbology ID transmission option shown next page.

Syntax

```
HRESULT SetSymIdXmit( ITC_SYMBOLOGY_ID_XMIT eSymIdXmit );
```

Parameters

eSymIdXmit	[in]	Identifies the symbology identifier transmission option to update.

Return Values

HRESULT that indicates success or failure.

Remarks

None.

See Also

None.

Symbology ID Transmission Option

The symbology identifier (or code mark) concept provides a standardized way for a device receiving data from a bar code reader to differentiate between the symbologies.

The following symbology ID transmission option specifies whether or not the symbology ID should be transmitted as part of the scanned bar code label to all the connected data collection applications. Options for transmission are: do not transmit, transmit the standard AIM identifiers, or transmit the one byte custom defined identifiers. AIM and custom identifiers cannot be selected to be transmitted at the same time; only the last selected option will be active.

```
typedef enum tagSymbologyIdXmit

{
```

`ITC_ID_XMIT_DISABLE = 0`	Symbology identifier will not be transmitted as part of the label. This is the default setting.
`ITC_ID_XMIT_CUSTOM = 1`	Activate custom symbology identifier transmission for all symbologies. Example of the transmitted label: [preamble] [Custom ID] <data> [postamble]
`ITC_ID_XMIT_AIM = 2`	Activate AIM symbology identifier transmission for all symbologies. Example of the transmitted label: [preamble] [AIM symbology ID] <data> [postamble]

```
}ITC_SYMBOLOGY_ID_XMIT;
```

IS9CConfig3 Functions

The IS9CConfig3 interface provides generic methods for retrieving and setting configuration using ISCP commands.

ISCP Commands

An ISCP Command is composed of three or more bytes formatted as <SG><FID><parameters> where:

• *SG*	Setup group.
• *FID*	Function ID.
• *parameters*	One or more configuration value bytes depending on the configuration.

ISCP commands include the following:

Imager Settings

This dictates the start and end column positions for the image dimension.

SG	FID	Parameter	Description
0x7B	80	Value [0..639]	Start column position.
0x7B	81	Value [0..639]	End column position.

Trigger Settings

This sets the duration of the aiming beam before acquiring images to be decoded.

SG	FID	Parameter	Description
0x70	81	Value [0..65535]	Number of milliseconds.

QRCode Symbology

This enables or disables the QRCode symbology.

SG	FID	Parameter	Description
0x55	40	0	Disable this symbology.
0x55	40	1	Enable this symbology.

Data Matrix Symbology

This enables or disables the Data Matrix symbology.

SG	FID	Parameter	Description
0x54	40	0	Disable this symbology.
0x54	40	1	Enable this symbology.

ISCP::GetConfig

This retrieves configurations using the ISCP commands format.

Syntax

```
HRESULT ISCPGetConfig( BYTE rgbCommandBuff[], DWORD
dwCommandBuffSize, BYTE rgbReplyBuff[], DWORD
dwReplyBuffMaxSize, DWORD *pdwReplyBuffSize );
```

Parameters

rgbCommandBuff	[in, size_is]	Contains ISCP commands in array of bytes.
dwCommandBuffSize	[in]	Number of bytes in *rgbCommandBuff*.
rgbReplyBuff	[in, out, size_is]	Results of query in array of bytes.
dwReplyBuffMaxSize	[in]	Maximum size of *rgdReplyBuff*.
pdwReplyBuffSize	[in, out]	Number of bytes placed in *rbfReplyBuff*.

Return Values

None.

Remarks

None.

See Also

None.

ISCP::SetConfig

This updates configurations using the ISCP commands format.

Syntax

```
HRESULT ISCPSetConfig( BYTE rgbCommandBuff[], DWORD
dwCommandBuffSize, BYTE rgbReplyBuff[], DWORD
dwReplyBuffMaxSize, DWORD *pdwReplyBuffSize );
```

Parameters

rgbCommandBuff	[in, size_is]	Contains ISCP commands in array of bytes.
dwCommandBuffSize	[in]	Number of bytes in *rgbCommandBuff*.
rgbReplyBuff	[in, out, size_is]	Results of request in array of bytes.
dwReplyBuffMaxSize	[in]	Maximum size of *rgbReplyBuff*.
pdwReplyBuffSize	[in, out]	Number of bytes placed in *rgbReplyBuff*.

Return Values

None.

Remarks

None.

See Also

None.

AIM Symbology ID Defaults

Refer to the official AIM documentation on symbology identifiers for full information on the different processing options supported.

Symbology	ID Character	Modifier Characters	
Codabar	F	0	Standard Codabar symbol. No special processing.
		1	ABC Codabar (American Blood commission) concatenate/message append performed.
		2	Reader has validated the check character.
		4	Reader has stripped the check character before transmission.
Code 11	H	0	Single modulo 11 check character validated and transmitted.
		1	Two modulo 11 check characters validated and transmitted.
		3	Check characters validated but not transmitted.
Code 39	A	0	No check character validation nor full ASCII processing. All data transmitted as decoded.
		1	Modulo 43 check character validated and transmitted.
		3	Modulo 43 check character validated but not transmitted.
		4	Full ASCII character conversion performed. No check character validation.
		5	Full ASCII character conversion performed. Modulo 43 check character validated and transmitted.
		7	Full ASCII character conversion performed. Modulo 43 check character validated but not transmitted.
Code 93	G	0	No options specified. Always transmit 0.
Code128	C	0	Standard data packet. No FNC1 in first or second symbol character position after start character.
		1	EAN/UCC-128 data packet. FNC1 in first symbol character position after start character.
		2	FNC1 in second symbol character position after start character.
		4	Concatenation according to International Society for Blood Transfusion specifications was performed. Concatenated data follows.
Interleaved 2 of 5	I	0	No check character validation.
		1	Modulo 10 symbol check character validated and transmitted
		3	Modulo 10 symbol check character validated but not transmitted.
Matrix 2 of 5	X	0`F	For symbologies or symbology options not listed, a code character with the value 0-F may be assigned by the decoder manufacturer to identify those symbologies and options implemented in the reader.
MSI	M	0	Modulo 10 symbol check character validated and transmitted.
		1	Modulo 10 symbol check character validated but not transmitted.

Symbology (continued)	ID Character	Modifier Characters
PDF 417/ Micro PDF 417	L	0 Reader set to conform with protocol defined in 1994 PDF 417 specifications. 1 Reader set to follow protocol of ENV 12925 for Extended Channel Interpretation (all data characters 92 doubled). 2 Reader set to follow protocol of ENV 12925 for Basic Channel Interpretation (data characters 92 are not doubled). 3 Code 128 emulation: implied FNC1 in first position. 4 Code 128 emulation: implied FNC1 after initial letter or pair of digits. 5 Code 128 emulation: no implied FNC1.
Plessey	P	0 No options specified. Always transmit 0.
Standard 2 of 5 (2-bar start/stop)	R	0 No check character validation. 1 Modulo 7 check character validated and transmitted. 3 Modulo 7 check character validated but not transmitted.
Standard 2 of 5 (3-bar start/stop)	S	0 No options specified. Always transmit 0.
Telepen	B	0 Full ASCII mode 1 Double density numeric only mode 2 Double density numeric followed by full ASCII 4 Full ASCII followed by double density numeric
UPC/EAN	E	Consider UPC/EAN symbols with supplements as two separate symbols. The first symbol is the main data packet, and the second symbol is the 2 or 5 digit supplement. Transmit these two symbols separately, each with its own symbology identifier. Provision is made for the option of transmitting both symbols as a single data packet. 0 Standard data packet in full EAN format (13 digits for EAN-13, UPC-A, and UPC-E; does not include add-on data). 1 Two digit add-on data only. 2 Five digit add-on data only. 3 Combined data packet comprising 13 digits from EAN-13, UPC-A, or UPC-E symbol and 2 or 5 digits from add-on symbol. 4 EAN-8 data packet
IMPORTANT: The “symbology_id” character letter ***must*** *be uppercase for the above definitions.*		

IImage Interface

The IImage interface gives the application the capability to acquire images. The image acquired can be either a raw image as captured by the digital camera or it can be normalized. A normalized image is presented the same as if the picture were taken at right angles to the image and at the same distance. The normalized image is commonly used for signature capture applications.

- IImage::ReadSigCapBuffer *(page 221)*
- IImage::ReadSigCapFile *(page 224)*
- IImage::ReadImage *(page 225)*
- IImage::CancelReadImage *(page 226)*
- IImage::Start *(page 226)*
- IImage::Stop *(page 227)*
- IImage::Open *(page 227)*
- IImage::Close *(page 228)*

IImage::ReadSigCapBuffer

Syntax

```
HRESULT IImage::ReadSigCapBuffer( ITC_SIGCAP_SPEC
*pSigCapSpec, ITC_IMAGE_SPEC *pImgBuffer, DWORD nMaxBuffSize
);
```

Parameters

Parameters:

pSigCapSpec	[in]	Pointer to the structure that identifies the signature capture region. This structure is defined as follows:

```
typedef struct tagITCSigCapSpec
{
DWORD dwStructSize;
INT iAspectRatio;
INT iOffsetX;
INT iOffsetY;
UINT uiWidth;
UINT uiHeight;
INT iResolution;
ITCFileFormat eFormat;
DWORD eDepth;
} ITC_SIGCAP_SPEC;
```

where:

• *dwStructSize*	Size, in bytes, of this struct. This is for version control.
• *iAspectRatio*	Ratio of the bar code height (linear bar codes) or row height (2D bar codes) to the narrow element width.
• *iOffsetX*	Offset in X direction, relative to barcode center. Positive values are right of the bar code, negative values to the left.

• *iOffsetY*	Offset in Y direction, relative to barcode center. Positive values are higher than the bar code, negative values lower.
• *uiWidth*	Width of signature capture image region in intelligent bar code units.
• *uiHeight*	Height of the signature capture image region in intelligent bar code units.
• *iResolution*	Number of pixels per intelligent bar code unit.
• *eFormat*	Format of the image buffer returned as follows. Currently, only ITC_FILE_RAW is supported.

```
  ITC_FILE_KIM =              0,    // Returns data a KIM file
  ITC_FILE_TIFF_BIN =         1,    // TIFF Binary file
  ITC_FILE_TIFF_BIN_GROUP4 =  2,    // TIFF Binary Group 4 compressed
  ITC_FILE_TIFF_GRAY_SCALE =  3,    // TIFF Gray Scale
  ITC_FILE_RAW =              4,    // Raw image
  ITC_FILE_JPEG =             5,    // JPEG image
```

• *eDepth*	Number of bits per pixel. Currently, only one (monochrome) or eight (gray-scale) are supported.

pImgBuffer	[out]	Pointer to the buffer in which the signature capture image will be put.

```
typedef struct tagITCImageSpec
{
  DWORD dwStructSize;
  LONG  biWidth;
  LONG  biHeight;
  WORD  biBitCount;
  ITC_FILE_FORMAT eFormat;
  DWORD biActualImageSize;
  DWORD biMaxImageBytes;
  BYTE  rgbImageData[1];
} ITC_IMAGE_SPEC;
```

where:

• *dwStructSize*	Size, in bytes, of this struct. This is for version control.
• *biWidth*	The width of each row in pixels.
• *biHeight*	The number of rows in the image data.
• *biBitCount*	The number of bits per pixel.
• *eFormat*	Identifies the image format.
• *biActualImageSize*	Total bytes of image data returned.
• *biMaxImageBytes*	Maximum bytes that can be stored in *rgbImageData[]*.
• *rgbImageData*	Buffer containing the actual data, for example a 640x480 uses a 307200-byte buffer. The array size of this buffer is arbitrary so do *not* use this structure directly to reserve memory. The actual dimension of the buffer is identified by *biMaxImageBytes*.

Return Values

HRESULT identifying success or error. On error, the following codes will be returned:

- **S_OK**
  Image successfully returned.
- **ITC_RESULT_ERR_BADREGION_E**
  The specified region is not in the image.
- **ITC_RESULT_NO_BC_DECODED_E**
  A bar code has not yet been decoded or the last bar code decoded was not a signature capture symbology.
- **ITC_IMGBUFF_TOO_SMALL_E**
  pImgBuffer is too small to contain the signature captured image.
- **ITC_INV_PARAMETER_E**
  One of the parameters is invalid.
- **S_DEVICE_NOT_OPENED_E**
  The device had not been opened.

Remarks

ReadSigCapBuffer() will return the image from the last decoded label with dimensions identified by the calling parameter. This signature capture region must include the signature capture bar code. The supported bar codes for signature capture are: PDF 417, Code 128, and Code 39. The caller specifies the width, height, and center of the image to be retrieved. This image is independent of any rotation of the bar code relative to the imager. Thus, if the bar code is decoded with the code itself upside down to the imager, the retrieved image will still be right side up. However, if the specified image is outside the field of view a result code of ITC_RESULT_ERR_BADREGION_E will be returned.

This function uses the dimensions of the last decoded bar code as its coordinate system. Thus, all the parameters describing the image size and position are in units called “Intelligent Bar Code Units.” An Intelligent Bar Code Unit is equivalent to the narrow element width of the bar code.

The dimensions of the resulting image can be calculated with this formula:

```
Resulting Width = Specified Width * Specified Resolution
Resulting Height = Specified Height * Specified Resolution
```

See Also

None.

IImage::ReadSigCapFile

Note: This has not been implemented as of this publication.

Syntax

```
HRESULT IImage::ReadSigCapFile( ITC_SIGCAP_SPEC
*pSigCapSpec, LPCTSTR pszFileName );
```

Parameters

pSigCapSpec	[in]	Pointer to the structure that identifies the signature capture region. See ReadSigCapFile (page 221) for a description of this structure.
pszFileName	[in]	Name of the file in which to copy the image.

Return Values

HRESULT identifying success or error. On error, the following codes will be returned:

- S_OK
  Image successfully returned.
- ITC_RESULT_ERR_BADREGION_E
  The specified region is not in the image.
- ITC_RESULT_NO_BC_DECODED_E
  A bar code has not yet been decoded or the last bar code decoded was not a signature capture symbology.
- ITC_FILE_OPEN_E
  The file could not be opened.
- ITC_INV_PARAMETER_E
  One of the parameters is invalid.
- S_DEVICE_NOT_OPENED_E
  The device had not been opened.

Remarks

ReadSigCapFile() will write the image from the last decoded label with dimensions identified by the calling parameter. If the file already exists, its contents will be overwritten.

This signature capture region must include the signature capture bar code. The supported bar codes for signature capture are: PDF 417, Code 128, and Code 39. The caller specifies the width, height, and center of the image to be retrieved. This image is independent of any rotation of the bar code relative to the imager. Thus, if the bar code is decoded with the code itself upside down to the imager, the retrieved image will still be right side up. However, if the specified image is outside the field of view a result code of ITC_RESULT_ERR_BADREGION_E will be returned.

This function uses the dimensions of the last decoded bar code as its coordinate system. Thus, all the parameters describing the image size and position are in units called “Intelligent Bar Code Units”. An Intelligent Bar Code Unit is equivalent to the narrow element width of the bar code.

The dimensions of the resulting image can be calculated with this formula:

```
Resulting Width = Specified Width * Specified Resolution
Resulting Height = Specified Height * Specified Resolution
```

See Also

None.

IImage::ReadImage

Syntax

```
HRESULT IImage::Read( ITCFileFormat eFormat, DWORD nDepth,
ITC_IMAGE_SPEC *pImgBuffer, DWORD dwTimeout );
```

Parameters

eFormat	[in]	Format of the image buffer returned as follows. Currently, only ITC_FILE_RAW is supported.

```
ITC_FILE_KIM =              0,    // Returns data a KIM file
ITC_FILE_TIFF_BIN =         1,    // TIFF Binary file
ITC_FILE_TIFF_BIN_GROUP4 =  2,    // TIFF Binary Group 4 compressed
ITC_FILE_TIFF_GRAY_SCALE =  3,    // TIFF Gray Scale
ITC_FILE_RAW =              4,    // Raw image
ITC_FILE_JPEG =             5,    // JPEG image
```

nDepth	[in]	Number of bits per pixel. Currently, only eight (gray-scale) are supported.
pImgBuffer	[in/out]	Pointer to the buffer containing the image.
dwTimeout	[in]	Milliseconds to wait for the image to be returned.

Return Values

HRESULT identifying success or error. On error, these will be returned:

• S_OK	Image successfully returned.
• ITC_IMGBUFF_TOO_SMALL_E	*pImgBuffer* is too small to contain the signature captured image.
• ITC_TIMEOUT_E	Timeout.
• ITC_INV_PARAMETER_E	One of the parameters is invalid.
• S_DEVICE_NOT_OPENED_E	The device had not been opened.

Remarks

The image is returned in *pImgBuffer* in the caller specified format.

See Also

None.

IImage::CancelReadImage

Syntax

```
HRESULT IImage::CancelReadImage( );
```

Parameters

None.

Return Values

Status code indicating success or failure as follows:

• S_OK	Imager closed.
• S_DEVICE_NOT_OPENED_E	The device had not been opened.

Remarks

This function causes a pending image read of IImage::ReadImage() to return immediately with an error status. The purpose of this function is to allow the application to release a thread blocked on the ReadImage() call.

See Also

None.

IImage::Start

Syntax

```
HRESULT IImage::Start( );
```

Parameters

None.

Return Values

Status code indicating success or failure as follows:

• S_OK	Imager started.
• S_DEVICE_NOT_OPENED_E	The device had not been opened.

Remarks

This function starts the image continuously capturing images.

See Also

None.

IImage::Stop

Syntax

```
HRESULT IImage::Stop( );
```

Parameters

None.

Return Values

Status code indicating success or failure as follows:

• S_OK	Imager started.
• S_IMG_NOT_PRESENT_E	Unit does not contain an imager.
• S_DEVICE_NOT_OPENED_E	Device had not been opened.

Remarks

This function stops the image continuously capturing images.

See Also

None.

IImage::Open

Syntax

```
HRESULT IImage::Open( BOOL fSigCapEnable );
```

Parameters

fSigCapEnable	[in]	When TRUE, signature capture is enabled. When FALSE, it is disabled. Bar code labels are decoded and images (via IImage::ReadImage) the same.

Return Values

Status code indicating success or failure as follows:

• S_OK	Imager opened.
• S_IMG_NOT_PRESENT_E	Unit does not contain an imager.
• S_DEVICE_CONTENTION_E	Device has already been opened.

Remarks

This function exclusively allocates the imager device so that the other IImage methods can be safely called.

See Also

None.

IImage::Close

Syntax

```
HRESULT IImage::Close( );
```

Parameters

None.

Return Values

Status code indicating success or failure as follows:

• S_OK	Imager closed.
• S_DEVICE_NOT_OPENED_E	The device had not been opened.

Remarks

This function releases the imager device so that other applications can open it. An IImage::Release() will also close the imager device.

See Also

None.

Data Collection Configuration

Data Collection

Scanner settings for the 700 Series Computer can be configured via the **Data Collection** control panel applet. From the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Data Collection** icon. See *Appendix A*, "*Control Panel Applets*" for more information about the following parameters. *Note that these are in alphabetical order.*

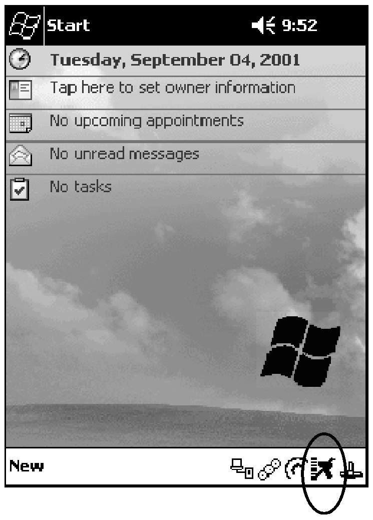
- Codabar *(page 292)*
- Code 11 *(page 306)*
- Code 128 *(page 295)*
  - Code 128 Options *(page 296)*
  - Code 128 FNC1 Character *(page 297)*
- Code 39 *(page 290)*
- Code 93 *(page 294)*
  - Code 93 Length *(page 294)*
- Data Matrix *(page 308)*
- Interleaved 2 of 5 *(page 303)*
- Matrix 2 of 5 *(page 304)*
- MSI *(page 299)*
- PDF 417 *(page 300)*
  - Macro PDF *(page 300)*
  - Micro PDF 417 *(page 302)*
- Plessey *(page 298)*
- QR Code *(page 307)*
- Standard 2 of 5 *(page 291)*
- Telepen *(page 305)*
- UPC/EAN *(page 293)*

Tethered Scanner

The Intermec Tethered Scanner feature accepts data from the COM1 port wedges it to the keyboard interface, and allows some ADC. This feature can be enabled or disabled from the Today Screen on the 700 Series Computer.

Enabling and Disabling

On the 700 Series Computer, tap **Start** → **Today**. Tap the bar code scanner icon in the System Tray *(circled in the following illustration)*. Initially, the bar code scanner icon indicates that this feature is disabled *(shown to the left)*.



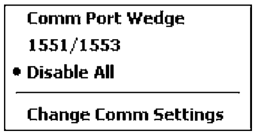
Tuesday, September 04, 2001
Tap here to set owner information
No upcoming appointments
No unread messages
No tasks

- Select **Comm Port Wedge** to send any data, coming into the 700 Series Computer through the COM1 port from an external input device, as keyboard data to an application on the desktop.

  For example, if you have Pocket Word running on your 700 Series Computer desktop, information scanned with a scanner connected to the COM1 port will appear in the Word document. If another data collection application is running and is active on the 700 Series Computer, the scanned information will appear in that application.

Note: When **Comm Port Wedge** is selected, regardless of the data sent by the external input device, you cannot control the device or the data format using any of the Intermec scanner control or data transfer APIs from the SDK or the internal Data Collection software. The external input device is governed by what software it has onboard to tell it how to scan, take pictures, or send the data elsewhere.

- Select **1551/1553** to enable the Sabre 1551E or 1553 Tethered Scanner to scan, then send data as keyboard data. The 1551/1553 Tethered Scanner has software onboard that translates scanned data into characters, so the running/active application does not need to know how to do that. All the scanner control and data transfer APIs will work with the 1551/1553 Tethered Scanner, so you can control the device.
- Select **Disable All** to disable this feature and use the COM1 port for another application, such as ActiveSync. An error message will result if this option were not selected, but this action was attempted. Similarly, if ActiveSync is using the COM1 port, and you select **Comm Port Wedge** or **1551/1553**, an error message will result. See "*Error Message*" on page 232 for more information.



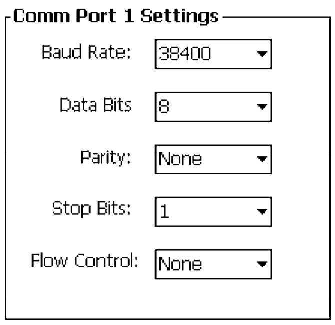
Changing Comm Settings

Tap **Change Comm Settings** to configure the settings for the COM1 port. Current settings are restored after a warm-boot, but are lost after a cold-boot. When these settings have not been changed, the **OK** button is disabled (grayed out). When changes are made, tap **OK** after it is enabled to accept these changes.

• **Baud Rate**:	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
• **Data Bits**:	7 or 8
• **Parity**:	None, Odd, Even, Mark, Space
• **Stop Bits**:	1 or 2
• **Flow Control**:	None or Hardware

Tethered Scanner

The default settings for the Tethered Scanner are shown in the following illustration:



Comm Port 1 Settings

Baud Rate: 38400

Data Bits 8

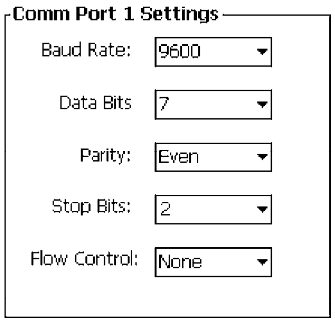
Parity: None

Stop Bits: 1

Flow Control: None

Sabre 1551E or 1553 Tethered Scanner

The default communication configuration for the Sabre 1551E or 1553 Tethered Scanner is shown in the following illustration. Scan the EasySet Reset Factory Defaults label to set the Sabre 1551E or 1553 tethered scanner communications settings to this configuration. The COM1 port configuration settings must also match those of the scanner to scan labels.



Comm Port 1 Settings

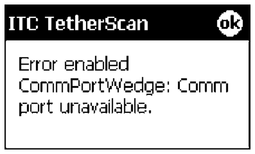
Baud Rate:	9600
Data Bits	7
Parity:	Even
Stop Bits:	2
Flow Control:	None

Welch Allyn 1470 Imager Settings

The Welch Allyn 1470 Imager can be set to this configuration by scanning the Factory Default Settings label.

Error Message

If the COM1 port is used by another application, such as ActiveSync, neither the Comm Port Wedge nor the 1551/1553 Tethered Scanner can be enabled. As a result, the following message may appear. *Note that this message is for the Comm Port Wedge.* You must disable that application to free up the COM1 port before you can enable either the wedge or the scanner.



ITC TetherScan

Error enabled CommPortWedge: Comm port unavailable.

Scanner Cabling

A null modem cable is required for the Welch Allyn 1470 Imager to communicate with the 700 Series Computer when using the 700 Series Serial Cable (P/N: 226-999-001).

The Sabre 1551E / 1553 Cable connects directly to the Model 700 Comm Port.

Limitations and Capabilities

The Tethered Scanner has the following limitations:

- No auto detection of a scanner' s physical connection to COM1 port. User needs to ensure the communication settings of COM1 port matched the settings of the device.
- The Pocket PC Pocket Office applications misbehave when control characters such as carriage return are wedged. This is a known Pocket PC problem, which is being worked with Microsoft and for which a work around is being developed.
- Communications port is COM1 and cannot be changed.
- A complete bar code label is detected when the time between bytes (the inter-byte gap) exceeds 100 ms. This allows that data could be concatenated if two labels were received while the Comm Port Wedge or the 1551/1553 Tethered Scanner was not performing a read. That is, it could be wedging data just read or the read thread could be preempted. Also, the labels could appear concatenated if the scanner itself were to buffer the labels before transmitting them.

When enabled, the Comm Port Wedge menu option has the following limitation:

- There is no bar code API to get bar code data from the bar code scanner. The Comm Port Wedge transmits the data through the keyboard interface only.

When enabled, the 1551/1553 menu option has the following capabilities:

- Grid Data Editing is available.
- The source of the symbology configurations is only available via the Easy Set command labels. Only the Virtual Wedge configurations can be configured via the Data Collection Control Panel Applet Virtual Wedge page. See Appendix A, "*Control Panel Applets*," for more information.
- May transmit the data through the keyboard interface (via the Virtual Wedge).

- The bar code APIs, defined in the IADC interface, are available to get bar code data from the bar code scanner. The following example shows how to programmatically collects bar code data:

```
#include "IADC.h"                  // Linked with ITCUUID.LIB
#include "ITCAdcMgmt.h"            // Linked with ITCAdcDevMgmt.lib

  IADC* pIADC;
  HRESULT hrStatus = S_OK;

// Create a ADC COM interface to collect bar code data from the 1551E/1553
// when the 1551/1553 menu option is enabled.
  hrStatus =
  ITCDeviceOpen(TEXT("ExtScanner"),  // Name of the ADC device.
    IID_IADC,                        // COM interface to return
    ITC_DHDEVFLAG_READAHEAD,         // Device's Flags
    (LPVOID *) &pIADC);              // the returned interface

if( SUCCEEDED(hrStatus) )
  {
    BYTE byteBuffer[MAX_LABEL_SIZE];
    DWORD dwLength = 0;
  HRESULT hr = pIDC->Read(
    byteBuffer,                      // Buffer to put the ADC data.
    MAX_LABEL_SIZE,                  // Size of pDataBuffer in bytes.
    &dwLength,                       // Number bytes returned.
    NULL,                            // Time stamp of the received data. NULL.
    INFINITE                         // Number of milliseconds to wait.
  );

}
    when done using this COM interface, delete it:
ITCDeviceClose( (IUnknown **) pIADC);
```

7 Programming

The following programming information pertains to the 700 Series Color Mobile Computer:

Creating CAB Files

The Windows CE operating system uses a .CAB file to install an application on a Windows CE-based device. A .CAB file is composed of multiple files that are compressed into one file. Compressing multiple files into one file provides the following benefits:

- All application files are present.
- A partial installation is prevented.
- The application can be installed from several sources, such as a desktop computer or a Web site.

Use the CAB Wizard application (CABWIZ.EXE) to generate a .CAB file for your application.

Creating Device-Specific CAB Files

Do the following to create a device-specific .CAB file for an application, *in the order provided*:

1. Create an .INF file with Windows CE-specific modifications *(page 236)*.
2. *Optional* Create a SETUP.DLL file to provide custom control of the installation process *(page 248)*.
3. Use the CAB Wizard to create the .CAB file, using the .INF file, the optional SETUP.DLL file, and the device-specific application files as parameters *(page 249)*.

Creating an .INF File

An .INF file specifies information about an application for the CAB Wizard. Below are the sections of an .INF file:

[Version]

This specifies the creator of the file, version, and other relevant information.

Required? Yes

- **Signature**: "*signature_name*"
  Must be "$Windows NT$" as Windows CE is not available on Windows 95.
- **Provider**: "*INF_creator*"
  The company name of the application, such as "Microsoft."
- **CESignature**: "$Windows CE$"

EXAMPLE:

```
[Version]
Signature = "$Windows NT$"
Provider = "Microsoft"
CESignature = "$Windows CE$"
```

[CEStrings]

This specifies string substitutions for the application name and the default installation directory.

Required? Yes

- **AppName**: *app_name*
  Name of the application. Other instances of %AppName% in the .INF file will be replaced with this string value, such as RP32.
- **InstallDir**: *default_install_dir*
  Default installation directory on the device. Other instances of %InstallDir% in the .INF file will be replaced with this string value. Example: \storage_card\%AppName%

EXAMPLE:

```
[CEStrings]
AppName="Game Pack"
InstallDir=%CE1%\%AppName%
```

[Strings]

This section is optional and defines one or more string keys. A string key represents a string of printable characters.

Required? No

- **string_key**: *value*
  String consisting of letters, digits, or other printable characters. Enclose *value* in double quotation marks """" if the corresponding string key is used in an item that requires double quotation marks. No string_keys is okay.

EXAMPLE:

```
[Strings]
reg_path = Software\Microsoft\My Test App
```

[CEDevice]

Describes the platform for the targeted application. All keys in this section are optional. If a key is nonexistent or has no data, Windows CE does not perform any checking with the exception being *UnsupportedPlatforms*. If the *UnsupportedPlatforms* key exists but no data, the previous value is not overridden.

Required? Yes

- **ProcessorType** : *processor_type*
  The value that is returned by **SYSTEMINFO**.dwProcessorType. For example, the value for the SH3 CPU is 10003 and the MIPS CPU is 4000.
- **UnsupportedPlatforms**: *platform_family_name*
  This lists known unsupported platform family names. If the name specified in the [**CEDevice.xxx**] section is different from that in the [**CEDevice**] section, both *platform_family_name* values are unsupported for the microprocessor specified by xxx. That is, the list of unsupported platform family names is appended to the previous list of unsupported names. Application Manager will not display the application for an unsupported platform. Also, a user will be warned during the setup process if the .CAB file is copied to an unsupported device.

EXAMPLE:

```
[CEDevice]
UnsupportedPlatforms = pltfrm1 ; pltfrm1 is unsupported
[CEDevice.SH3]
UnsupportedPlatforms = ; pltfrm1 is still unsupported
```

- **VersionMin**: *minor_version*
  Numeric value returned by **OSVERSIONINFO**.dwVersionMinor. The .CAB file is valid for the currently connected device if the version of this device is greater than or equal to **VersionMin**. For Windows CE Japanese language devices, set this to 2.01
- **VersionMax**: *major_version*
  Numeric value returned by **OSVERSIONINFO**.dwVersionMajor. The .CAB file is valid for the currently connected device if the version of this device is less than or equal to **VersionMax**. For Windows CE Japanese language devices, set this to 2.01

Note: Supported Windows CE operating system versions include 1.0, 1.01, 2.0, 2.01, and 2.10. When using these numbers, be sure to include all significant digits.

- **BuildMin**: *build_number*
  Numeric value returned by **OSVERSIONINFO**.dwBuildNumber. The .CAB file is valid for the currently connected device if the version of this device is greater than or equal to **BuildMin**.
- **BuildMax**: *build_number*
  Numeric value returned by **OSVERSIONINFO**.dwBuildNumber. The .CAB file is valid for the currently connected device if the version of this device is less than or equal to **BuildMax**.

EXAMPLE:

The following code example shows three [**CEDevice**] sections: one that gives basic information for any CPU and two that are specific to the SH3 and the MIPS microprocessors.

```
[CEDevice]                   ; A "template" for all platforms
UnsupportedPlatforms = pltfrm1 ; Does not support pltfrm1

; The following specifies version 1.0 devices only.
VersionMin = 1.0
VersionMax = 1.0

[CEDevice.SH3]               ; Inherits all [CEDevice] settings
; This will create a .CAB file specific to SH3 devices.
ProcessorType = 10003        ; SH3 .cab file is valid for SH3 microprocessors.
UnsupportedPlatforms =       ; pltfrm1 is still unsupported

; The following overrides the version settings so that no version checking is
performed.
VersionMin =
VersionMax =

[CEDevice.MIPS]              ; Inherits all [CEDevice] settings
; This will create a .CAB file specific to "MIPS" devices.
ProcessorType = 4000         ; MIPS .CAB file is valid for MIPS microprocessor.
UnsupportedPlatforms =pltfrm2  ; pltfrm1, pltfrm2 unsupported for MIPs .CAB file.
```

Note: To create the two CPU-specific .CAB files for the SETUP.INF file in the previous example, run the CAB Wizard with the "/cpu sh3 mips" parameter.

[DefaultInstall]

This describes the default installation of your application. Note that under this section, you will list items expanded upon later in this description.

Required? Yes

- **Copyfiles**: *copyfile_list_section*
  Maps to files defined later in the .INF file, such as Files.App, Files.Font, and Files.Bitmaps.
- **AddReg**: *add_registry_section*
  Example: RegSettings.All
- **CEShortcuts**: *shortcut_list_section*
  String that identifies one more section that defines shortcuts to a file, as defined in the [**CEShortcuts**] section.
- **CESetupDLL**: *setup_DLL*
  Optimal string that specifies a SETUP.DLL file. It is written by the Independent Software Vendor (ISV) and contains customized functions for operations during installation and removal of the application. The file must be specified in the [**SourceDisksFiles**] section.
- **CESelfRegister**: *self_reg_DLL_filename*
  String that identifies files that self-register by exporting the **DllRegisterServer** and **DllUnregisterServer** Component Object Model (COM) functions. Specify these files in the [**SourceDiskFiles**] section. During installation, if installation on the device fails to call the file' s exported **DllRegisterServer** function, the file' s exported **DllUnregisterServer** function will not be called during removal.

EXAMPLE:

```
[DefaultInstall]
AddReg = RegSettings.All
CEShortcuts = Shortcuts.All
```

[SourceDiskNames]

This section describes the name and path of the disk on which your application resides.

Required? Yes

- **disk_ordinal**: *disk_label,,path*
  1=,"App files" , C:\Appsoft\RP32\...
  2=,"Font files",,C:\RpTools\...
  3=,"CE Tools" ,,C:\windows ce tools...
- **CESignature**: "$Windows CE$"

Example

```
[SourceDisksNames]                        ; Required section
1 = ,"Common files",,C:\app\common        ; Using an absolute path
[SourceDisksNames.SH3]
2 = ,"SH3 files",,sh3                     ; Using a relative path
[SourceDisksNames.MIPS]
2 = ,"MIPS files",,mips                   ; Using a relative path
```

[SourceDiskFiles]

This describes the name and path of the files in which your application resides.

Required? Yes

- **filename**: *disk_number[,subdir]*
  RPM.EXE = 1,c:\appsoft\...
  WCESTART.INI = 1
  RPMCE212.INI = 1
  TAHOMA.TTF = 2

Note: [**,subdir**] is relative to the location of the INF file.

Example

```
[SourceDisksFiles]       ; Required section
begin.wav = 1
end.wav = 1
sample.hlp = 1
[SourceDisksFiles.SH3]
sample.exe = 2           ; Uses the SourceDisksNames.SH3 identification of 2.
[SourceDisksFiles.MIPS]
sample.exe = 2           ; Uses the SourceDisksNames.MIPS identification of 2.
```

[DestinationDirs]

This describes the names and paths of the destination directories for the application on the target device. *Note Windows CE does not support directory identifiers.*

Required? Yes

- **file_list_section**: *0,subdir*
  String that identifies the destination directory. The following list shows the string substitutions supported by Windows CE. These can be used only for the beginning of the path. \

%CE1%	\Program Files
%CE2%	\Windows
%CE3%	\My Documents
%CE4%	\Windows\Startup
%CE5%	\My Documents
%CE6%	\Program Files\Accessories
%CE7%	\Program Files\Communication
%CE8%	\Program Files\Games
%CE9%	\Program Files\Pocket Outlook
%CE10%	\Program Files\Office
%CE11%	\Windows\Start Menu\Programs
%CE12%	\Windows\Start Menu\Programs\Accessories
%CE13%	\Windows\Start Menu\Programs\Communications
%CE14%	\Windows\Start Menu\Programs\Games
%CE15%	\Windows\Fonts
%CE16%	\Windows\Recent
%CE17%	\Windows\Start Menu

%InstallDir%
Contains the path to the target directory selected during installation. It is declared in the [**CEStrings**] section
%AppName%
Contains the application name defined in the [**CEStrings**] section.

Example

```
[DestinationDirs]
Files.Common  = 0,%CE1%\My Subdir    ; \Program Files\My Subdir
Files.Shared  = 0,%CE2%              ; \Windows
```

[CopyFiles]

This section, under the [**DefaultInstall**] section, describes the default files to copy to the target device. Within the [**DefaultInstall**] section, files were listed that must be defined elsewhere in the INF file. This section identifies that mapping and may contain flags.

Required? Yes

- **copyfile_list_section**: *destination_filename,[source_filename]*
  The *source_filename* parameter is optional if it is the same as *destination_filename*.
- **copyfile_list_section**: *flags*
  The numeric value that specifies an action to be done while copying files. The following table shows values supported by Windows CE.

Flag	Value	Description
COPYFLG_WARN_IF_SKIP	0x00000001	Warn user if skipping a file is attempted after error.
COPYFLG_NOSKIP	0x00000002	Do not allow a user to skip copying a file.
COPYFLG_NO_OVERWRITE	0x00000010	Do not overwrite files in destination directory.
COPYFLG_REPLACEONLY	0x00000400	Copy the source file to the destination directory only if the file is already in the destination directory.
CE_COPYFLG_NO_DATE_DIALOG	0x20000000	Do not copy files if the target file is newer.
CE_COPYFLG_NODATECHECK	0x40000000	Ignore date while overwriting the target file.
CE_COPYFLG_SHARED	0x80000000	Create a reference when a shared DLL is counted.

Example

```
[DefaultInstall.SH3]
CopyFiles = Files.Common, Files.SH3
[DefaultInstall.MIPS]
CopyFiles = Files.Common, Files.MIPS
```

[AddReg]

This section, under the [**DefaultInstall**] section, is optional and describes the keys and values that the .CAB file adds to the device registry. Within the [**DefaultInstall**] section, a reference may have been made to this section, such as “AddReg=RegSettings.All”. This section defines the options for that setting.

Required? No

- **add_registry_section**: *registry_root_string*
  String that specifies the registry root location. The following list shows the values supported by Windows CE.
  - HKCR Same as HKEY_CLASSES_ROOT
  - HKCU Same as HKEY_CURRENT_USER
  - HKLM Same as HKEY_LOCAL_MACHINE
- **add_registry_section**: *value_name*
  Registry value name. If empty, the “default” registry value name is used.
- **add_registry_section**: *flags*
  Numeric value that specifies information about the registry key. The following table shows the values that are supported by Window CE.

Flag	Value	Description
FLG_ADDREG_NOCLOBBER	0x00000002	If the registry key exists, do not overwrite it. Can be used with any of the other flags in this table.
FLG_ADDREG_TYPE_SZ	0x00000000	REG_SZ registry data type.
FLG_ADDREG_TYPE_MULTI_SZ	0x00010000	REG_MULTI_SZ registry data type. Value field that follows can be a list of strings separated by commas.
FLG_ADDREG_TYPE_BINARY	0x00000001	REG_BINARY registry data type. Value field that follows must be a list of numeric values separated by commas, one byte per field, and must not use the 0x hexadecimal prefix.
FLG_ADDREG_TYPE_DWORD	0x00010001	REG_DWORD data type. The noncompatible format in the Win32 Setup .INF documentation is supported.

Example

```
AddReg = RegSettings.All

[RegSettings.All]
HKLM,%reg_path%,,0x00000000,alpha          ; <default> = “alpha”
HKLM,%reg_path%,test,0x00010001,3          ; Test = 3
HKLM,%reg_path%\new,another,0x00010001,6    ; New\another = 6
```

[CEShortCuts]

This section, a Windows CE-specific section under the [**DefaultInstall**] section, is optional and describes the shortcuts that the installation application creates on the device. Within the [**DefaultInstall**] section, a reference may have been made to this section, such as “ShortCuts.All”. This section defines the options for that setting.

Required? No

- **shortcut_list_section**: *shortcut_filename*
  String that identifies the shortcut name. It does not require the .LNK extension.
- **shortcut_list_section**: *shortcut_type_flag*
  Numeric value. Zero or empty represents a shortcut to a file; any non-zero numeric value represents a shortcut to a folder.
- **shortcut_list_section**: *target_file_path*
  String value that specifies the destination location. Use the target file name for a file, such as MyApp.exe, that must be defined in a file copy list. For a path, use a *file_list_section* name defined in the [**DestinationDirs**] section, such as *DefaultDestDir*, or the *%InstallDir%* string.
- **shortcut_list_section**: *standard_destination_path*
  Optional string value. A standard *%CEx%* path or *%InstallDir%*. If no value is specified, the *shortcut_list_section* name of the current section or the *DefaultDestDir* value from the [**DestinationDirs**] section is used.

Example

```
CEShortcuts = Shortcuts.All
[Shortcuts.All]
Sample App,0,sample.exe        ; Uses the path in DestinationDirs. Sample
App,0,sample.exe,%InstallDir%  ; The path is explicitly specified.
```

Sample .INF File

```
[Version]         ; Required section
Signature = “$Windows NT$”
Provider = “Intermec Technologies Corporation”
CESignature = “$Windows CE$”

;[CEDevice]
;ProcessorType =

[DefaultInstall]  ; Required section
CopyFiles = Files.App, Files.Fonts, Files.BitMaps, Files.Intl,
Files.TelecomNcsCE, Files.Windows, Files.Import, Files.Export, Files.Work,
Files.Database, Files.WinCE AddReg = RegSettings.All ;CEShortcuts =
Shortcuts.All

[SourceDisksNames] ; Required section
1 = ,“App files” ,,c:\appsoft\...
2 = ,”Font files” ,,c:\WinNT\Fonts
3 = ,”CE Tools”  ,,c:\windows ce tools\wce212\6110ie\mfc\lib\x86

[SourceDisksFiles] ; Required section
rpm.exe   = 1,C:\Appsoft\program\wce212\WCEX86Rel6110
wcestart.ini = 1
```

```
rpmce212.ini = 1
intermec.bmp = 1
rpmlogo.bmp = 1
rpmname.bmp = 1
import.bmp  = 1
export.bmp  = 1
clock.bmp  = 1
printer.bmp = 1
filecopy.bmp = 1
readme.txt  = 1
lang_eng.bin = 1
rpmdata.dbd = 1,database\wce1
tahoma.ttf  = 2
mfcce212.dll = 3
olece212.dll = 3
olece211.dll = 1,c:\windows ce tools\wce211\NMSD61102.11\mfc\lib\x86
rdm45wce.dll = 1,c:\rptools\rdm45wce\4_50\lib\wce212\wcex86rel
picfmt.dll  = 1,c:\rptools\picfmt\1_00\wce212\wcex86rel6110
fmtctrl.dll = 1,c:\rptools\fmtctrl\1_00\wce212\wcex86rel6110
ugrid.dll  = 1,c:\rptools\ugrid\1_00\wce212\wcex86rel6110
simple.dll  = 1,c:\rptools\pspbm0c\1_00\wce211\wcex86rel
psink.dll = 1,c:\rptools\psink\1_00\wce211\WCEX86RelMinDependency
pslpwce.dll =1,c:\rptools\pslpm0c\1_00\wce211\WCEX86RelMinDependency
npcpport.dll = 1,c:\rptools\cedk\212_03\installable drivers\printer\npcp
;dexcom.dll  = 1,c:\rptools\psdxm0c\1_00\x86
ncsce.exe  = 1,c:\rptools\ncsce\1_04
nrinet.dll  = 1,c:\rptools\ncsce\1_04

[DestinationDirs]  ; Required section
;Shortcuts.All = 0,%CE3%  ; \Windows\Desktop
Files.App       = 0,%InstallDir%
Files.DataBase     = 0,%InstallDir%\DataBase
Files.BitMaps      = 0,%InstallDir%\Bitmaps
Files.Fonts        = 0,%InstallDir%\Fonts
Files.Intl         = 0,%InstallDir%\Intl
Files.TelecomNcsCE = 0,%InstallDir%\Telecom\NcsCE
Files.Windows      = 0,%InstallDir%\Windows
Files.Import       = 0,%InstallDir%\Import
Files.Export       = 0,%InstallDir%\Export
Files.Work         = 0,%InstallDir%\Work
Files.WinCE        = 0,\storage_card\wince

[CEStrings]        ; Required section
AppName = Rp32
InstallDir = \storage_card\%AppName%

[Strings]       ; Optional section
;[Shortcuts.All]
;Sample App,0,sample.exe                ; Uses the path in DestinationDirs.
;Sample App,0,sample.exe,%InstallDir%    ; The path is explicitly specified.

[Files.App]
rpm.exe,,,0
rpm.ini,rpmce212.ini,,0
mfcce212.dll,,,0
olece212.dll,,,0
olece211.dll,,,0
rdm45wce.dll,,,0
picfmt.dll,,,0
```

```
fmtctrl.dll,,,0
ugrid.dll,,,0
simple.dll,,,0
psink.dll,,,0
pslpwce.dll,,,0
npcpport.dll,,,0
;dexcom.dll,,,0

[Files.DataBase]
rpmdata.dbd,,,0

[Files.Fonts]
tahoma.ttf,,,0

[Files.BitMaps]
intermec.bmp,,,0
rpmlogo.bmp,,,0
rpmname.bmp,,,0
import.bmp,,,0
export.bmp,,,0
clock.bmp,,,0
printer.bmp,,,0
filecopy.bmp,,,0

[Files.Intl]
lang_eng.bin,,,0

[Files.TelecomNcsCE]
ncsce.exe,,,0
nrinet.dll,,,0

[Files.Windows]
readme.txt,,,0

[Files.Import]
readme.txt,,,0

[Files.Export]
readme.txt,,,0

[Files.Work]
readme.txt,,,0

[Files.WinCE]
wcestart.ini,,,0

[RegSettings.All]
HKLM,"SOFTWARE\Microsoft\Shell\AutoHide",,0x00010001,1
; Autohide the taskbar HKLM,"SOFTWARE\Microsoft\Shell\OnTop",,0x00010001,0
  ; Shell is not on top HKLM,"SOFTWARE\Microsoft\Clock",SHOW_CLOCK,0x00010001,0
; Clock is not on taskbar
```

Using Installation Functions in SETUP.DLL

SETUP.DLL is an optional file that enables you to perform custom operations during installation and removal of your application. The following list shows the functions that are exported by SETUP.DLL.

- **Install_Init**
  Called before installation begins. Use this function to check the application version when reinstalling an application and to determine if a dependent application is present.
- **Install_Exit**
  Called after installation is complete. Use this function to handle errors that occur during application installation.
- **Uninstall_Init**
  Called before the removal process begins. Use this function to close the application, if the application is running.
- **Uninstall_Exit**
  Called after the removal process is complete. Use this function to save database information to a file and delete the database and to tell the user where the user data files are stored and how to reinstall the application.

Note; Use [**DefaultInstall**] → **CESelfRegister** (page 240) in the .INF file to point to SETUP.DLL.

After the CAB File Extraction

Cab files that need to cause a warm reset after cab extraction will need to create the __RESETMEPLEASE__.TXT file in the “\Windows” directory. The preferred method to create this file is within the DllMain portion of the SETUP.DLL file. It looks like this:

```
BOOL APIENTRY DllMain( HANDLE hModule, DWORD  ul_reason_for_call, LPVOID
lpReserved  )
{
  switch (ul_reason_for_call)
    {
      case DLL_PROCESS_ATTACH:
        break;
      case DLL_THREAD_ATTACH:
        break;
      case DLL_THREAD_DETACH:
        break;
      case DLL_PROCESS_DETACH:
        if (bInstallSuccessful) {
          HANDLE h;
          h = CreateFile(L”\\Windows\\__resetmeplease__.txt”,
            GENERIC_READ|GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
            FILE_ATTRIBUTE_HIDDEN, NULL);
          if (h != INVALID_HANDLE_VALUE)
            CloseHandle(h);
        }
        break;
  }
  return TRUE;
}
```

The system software looks for the following directory structure and files on the installed media card whether it be an SD card or CF card or embedded flash file system. No other folders need exist.

```
\2577\autorun.exe
\2577\autorun.dat
\2577\autocab.exe
\2577\autocab.dat
\cabfiles\*.cab
```

Creating CAB Files with CAB Wizard

After you create the .INF file and the optional SETUP.DLL file, use the CAB Wizard to create the .CAB file. The command-line syntax for the CAB Wizard is as follows:

```
cabwiz.exe "inf_file" [/dest dest_directory] [/err error_file] [/cpu cpu_type
[cpu_type]]
```

A batch file, located in <program> directory, with the following commands, works well:

```
cd\"Windows CE Tools"\WCE211\"MS HPC Pro"\support\appinst\bin
cabwiz.exe c:\appsoft\<program>\<inf_file_name>
cd \appsoft\<program>
```

- "inf_file"
  The SETUP.INF file path.
- *dest_directory*
  The destination directory for the .CAB files. If no directory is specified, the .CAB files are created in the "inf_file" directory.
- *error_file*
  The file name for a log file that contains all warnings and errors that are encountered when the .CAB files are compiled. If no file name is specified, errors are displayed in message boxes. If a file name is used, the CAB Wizard runs without the user interface (UI); this is useful for automated builds.
- *cpu_type*
  Creates a .CAB file for each specified microprocessor tag. A microprocessor tag is a label used in the Win32 SETUP.INF file to differentiate between different microprocessor types. The */cpu* parameter, followed by multiple *cpu_type* values, must be the last qualifier in the command line.

Example

This example creates .CAB files for the SH3 and MIPS microprocessors, assuming that the Win32 SETUP.INF file contains the SH3 and MIPS tags:

```
cabwiz.exe "c:\myfile.inf" /err myfile.err /cpu sh3 mips
```

Note: CABWIZ.EXE, MAKECAB.EXE, and CABWIZ.DDF (Windows CE files available on the Windows CE Toolkit) must be installed in the same directory on the desktop computer. Call CABWIZ.EXE using its full path for the CAB Wizard application to run correctly.

Troubleshooting the CAB Wizard

To identify and avoid problems that might occur when using the CAB Wizard, follow these guidelines:

- Use %% for a percent sign (%) character when using this character in an .INF file string, as specified in Win32 documentation. This will not work under the [**Strings**] section.
- Do not use .INF or .CAB files created for Windows CE to install applications on Windows-based desktop platforms.
- Ensure the MAKECAB.EXE and CABWIZ.DDF files, included with Windows CE, are in the same directory as CABWIZ.EXE.
- Use the full path to call CABWIZ.EXE.
- Do not create a .CAB file with the MAKECAB.EXE file included with Windows CE. You must use CABWIZ.EXE, which uses MAKECAB.EXE to generate the .CAB files for Windows CE.
- Do *not* set the read-only attribute for .CAB files.

FTP Server

FTP support is provided through the FTP Server application FTPDCE.EXE (MS Windows CE Versions) which is provided as part the base system.

FTPDCE is the Internet File Transfer Protocol (FTP) server process. The server can be invoked from an application or command line. Besides servicing FTP client requests the FTP Server also send a "network announcement" to notify prospective clients of server availability.

Synopsis

ftpdce [*options*]

Options

- *-Aaddr*
  Sets the single target address to which to send the network announcement. *Default is broadcast.*
- *-Bbyte*
  Sets the FTP data block size. Smaller sizes may be useful over slower links. *Default is 65536.*
- *-Cname*
  Sets the device name. Used by Intermec management software.
- *-Fvalue*
  Disables the default Intermec account. A value of "0" disables the account. *Default is "1".*

Note: Disabling the default account without providing a working access control list on the server will result in a device that will not accept any FTP connections.

- *-Hsec*
  Sets the interval between network announcements in seconds.A value of "0" turns the network announcement off. *Default is 30 seconds.*
- *-Iip*
  Sets the preferred 6920 Communications Server *(optional).*
- *-Llog*
  Sets the state of logging. *Default is 0 (disabled).*
- *-Nsec*
  Specifies the number of seconds to wait before starting FTP server services.
- *-Pport*
  Sets the UDP port on which the network announcement will be sent. *Default port is 52401.*
- *-Qport*
  Sets the port on which the FTP Server will listen for connections. *Default port is 21.*

- *-Rdir*
  Sets the FTP mount point to this directory. Default is the rootdirectory of the drive from which the FTP Server program was executed.
- *-Tscript*
  Sets the script name for the 6920 Communications Server to process.
- *-Uurl*
  Sets the default URL for this device.
- *-Z"parms"*
  Sets extended parameters to be included in the network announcement.

Configurable Parameters Via the Registry Editor

The following parameters receive default values during the installation of the Intermec FTP Server components. A few of the parameters are visible in the registry by default, but most must be created in order to modify the default behavior of the FTP server.

BlockSize

Setting this parameter forces the Intermec FTP Server to transmit and receive Ethernet packets using the specified data block size. By default, the FTP server transmits and receives data using a 64K data block size. Adjusting this value may be useful in certain wireless TCP/IP installations.

Key

`HKLM\Software\Intermec\IFTP`

Value Type

REG_DWORD - data block size, in bytes.

Valid Range

0x100-0x10000 (256-65536 decimal).

Default

65536

DeviceName

This parameter forces the Intermec FTP Server to include the specified device name in the Intermec Device Network Announcement (IDNA). Adjusting this value may be useful in assigning a symbolic name to this device for asset tracking.

Key

`HKLM\Software\Intermec\IFTP`

Value Type

REG_SZ

Valid Range

None.

Default

None.

DeviceURL

This parameter forces the Intermec FTP Server to transmit the specified URL in the IDNA. This can be used by Intermec management software for asset management.

Key

`HKLM\Software\Intermec\IFTP`

Value Type

REG_SZ

Valid Range

None.

Default

None.

IDNATarget

This parameter forces the Intermec FTP Server to transmit the IDNA to a specific destination instead of a general UDP broadcast. This parameter is useful on networks that do not allow UDP broadcasts to be routed between subnets. The use of this parameter will restrict the reception of the IDNA to the target destination only.

Key

`HKLM\Software\Intermec\IFTP`

Value Type

REG_SZ

Valid Range

None.

Default

None.

ManifestName

This parameter forces the Intermec FTP Server to transmit the specified manifest name in the IDNA. This parameter is used by the Intermec 6920 Communications Server for communication transactions. See the 6920 Communications Server documentation for proper use of this parameter.

Key

`HKLM\Software\Intermec\IFTP`

Value Type

REG_SZ

Valid Range

None.

Default

iftp.ini

PauseAtStartup

This parameter forces the Intermec FTP Server to sleep for the specified number of seconds before making the FTP service available on the device.

Key

`HKLM\Software\Intermec\IFTP`

Value Type

REG_DWORD - stored in seconds.

Valid Range

None.

Default

0

Root

This parameter forces the Intermec FTP Server to set the root of the FTP mount point to the specified value. *Note that this must map to an existing directory or you will not be able to log into the FTP Server.*

Key

`HKLM\Software\Intermec\IFTP`

Value Type

REG_SZ

Valid Range

None.

Default

\

Transferring Files Over TCP/IP Networks

The File Transfer Protocol (FTP) server transfers files over TCP/IP networks. The FTPDCE.EXE program is a version that does not display a window, but can run in the background.

FTPDCE is the Internet File Transfer Protocol (FTP) server process. The server can be invoked from an application or command line. Besides servicing FTP client requests, the FTP Server also sends a "network announcement" to notify prospective clients of server availability.

Remarks

The FTP Server currently supports the following FTP requests:

- **CDUP**
  Changes to the parent directory of the current working directory.
- **CWD**
  Changes working directory.
- **DELE**
  Deletes a file.
- **HELP**
  Gives help information.
- **LIST** *(This FTP request is the same as the ls -lgA command).*
  Gives list files in a directory.
- **MKD**
  Makes a directory.
- **MODE** *(Always Uses Binary).*
  Specifies data transfer mode.
- **NLST**
  Gives a name list of files in directory (this FTP request is the same as the *ls* command).
- **NOOP**
  Does nothing.
- **PASS**
  Specifies a password.
- **PWD**
  Prints the current working directory.
- **QUIT**
  Terminates session.
- **RETR**
  Retrieves a file.
- **RMD**
  Removes a directory.
- **RNFR**
  Specifies rename-from file name.

- **RNTO**
  Specifies rename-to file name.
- **STOR**
  Stores a file.
- **SYST**
  Shows the operating system type of server system.
- **TYPE** *(Binary transfers only.)*
  Specifies the data transfer type with the Type parameter.
- **USER**
  Specifies user name.
- **XCUP** *(Not Normally Used)*
  Changes the parent directory of the current working directory.
- **XCWD** *(Not Normally Used)*
  Changes the current directory.
- **XMKD** *(Not Normally Used)*
  Creates a directory.
- **XPWD** *(Not Normally Used)*
  Prints the current working directory.
- **XRMD** *(Not Normally Used)*
  Removes a directory.
- **SITE**
  The following nonstandard or operating system (OS)-specific commands are supported by the SITE request. For Microsoft FTP clients, you can send site commands by preceding the command with "quote" such as "quote site status."
  - **ATTRIB**
    Gets or sets the attributes of a given file. (SITE ATTRIB)

    Usage: **QUOTE SITE ATTRIB** [+*R* | -*R*] [+*A* | -*A*] [+*S* | -*S*] [+*H* | -*H*] [[*path*] *filename*]

+	Sets an attribute.
`	Clears an attribute.
R	Read-only file attribute.
A	Archive file attribute.
S	System file attribute.
H	Hidden file attribute.

    To retrieve the attributes of a file, only specify the file. The server response will be: *200-AD SHRCEIX filename*

    If the flag exists in its position shown above, it is set. Also, in addition to the values defined above, there is also defined:

C	Compressed file attribute.
E	Encrypted file attribute.
I	INROM file attribute.
X	XIP file attribute (execute in ROM, not shadowed in RAM).

- BOOT
  Reboots the server OS. This will cause the system on which the server is executing to reboot. The FTP Server will shut down cleanly before reboot. All client connections will be terminated. Cold boot is default except for the PocketPC build in which the default is warm boot.
  (SITE BOOT)

  Usage: **QUOTE SITE BOOT** [*WARM* | *COLD*]

- COPY
  Copies a file from one location to another. (SITE COPY)

  Usage: **QUOTE SITE COPY** [*source*] [*destination*]

Example

```
QUOTE SITE COPY '\Storage Card\one.dat' '\Storage
Card\two.dat'
```

- EXIT
  Exits the FTP Server. This command will shut down the FTP Server thus terminating all client connections. (SITE EXIT)

  Usage: **QUOTE SITE EXIT**

- HELP
  Gives site command help information. (SITE HELP)

  Usage: **QUOTE SITE HELP** [*command*]

- KILL
  Terminates a running program. (SITE KILL)

  Usage: **QUOTE SITE KILL** [*program* | *pid*]

- LOG
  Opens or closes the program log. (SITE LOG)

  Usage: **QUOTE SITE LOG** [*open* [*filename*]| *close*]

- PLIST
  Lists the running processes *(not supported on all platforms)*.
  (SITE PLIST)

  Usage: **QUOTE SITE PLIST**

- RUN
  Starts a program running. If the program to run has spaces in path or filename, wrapping the name with single quotes is required.

  Usage: **QUOTE SITE RUN** [*program*]

Example

```
QUOTE SITE RUN '\Storage Card\app.exe'
```

- **STATUS**
  Returns the current settings of the FTP Server. MAC, serial number, model, IP address, network announcement information as well as OS memory usage are returned. (SITE STATUS)

  Usage: **QUOTE SITE STATUS**

- **TIMEOUT**
  Toggles idle timeout between 120 to 1200 seconds (2 to 20 minutes). If this timer expires with no activity between the client and the server, the client connection will be disconnected. If the optional seconds argument is supplied, the server will set the connection timeout to the number of seconds specified. *Default is 120 seconds or 2 minutes.* (SITE TIMEOUT)

  Usage: **QUOTE SITE TIMEOUT** [*seconds*]

The remaining FTP requests specified in RFC 959 are recognized, but not implemented.

The banner returned in the parenthetical portion of its greeting shows the version number of the FTP Server as well as the MAC address, serial number and OS of the machine hosting the server.

The FTP Server supports browsing from the latest Netscape and Microsoft web browsers. Drag-and-drop capability is available using this environment.

The FTPDCMDS subdirectory contains commands that can be used from the web browser.

- Click EXITME.BIN to execute a SITE EXIT command.
- Click REBOOTME.BIN to execute SITE BOOT command.
- Use the GET command on these files to have the FTP Server execute these commands.
  - **Security**:
    A customer configurable access control list may be installed on the 700 Series Computer. This list will allow customers to restrict access via the FTP Server to the users they wish. This is in addition to the default Intermec account which can be disabled using the *-F0* option at runtime.

    The access control list is named FTPDCE.TXT and is placed in the same directory on the 700 Series Computer as the FTPDCE.EXE server. The FTP Server will encrypt this file to keep the information safe from unauthorized users. This file is encrypted when the FTP Server is started so a file that is placed onto the 700 Series Computer after the FTP Server starts will require a restart of the FTP Server to take effect.

    The format of the FTPDCE.TXT is as follows:

```
FTPDCE:user1!passwd1<cr><lf>user2!passwd2<cr><lf>user3!passw
d3<cr><lf>...
```

Note: The user accounts and passwords are case sensitive.
Once the access control list is encrypted on the 700 Series Computer, the FTP Server will hide this file from users. Once an access control list has been installed on the 700 Series Computer, a new one will not be accepted by the FTP Server until the previous one is removed.
Encrypted access control lists are not portable between 700 Series Computers.

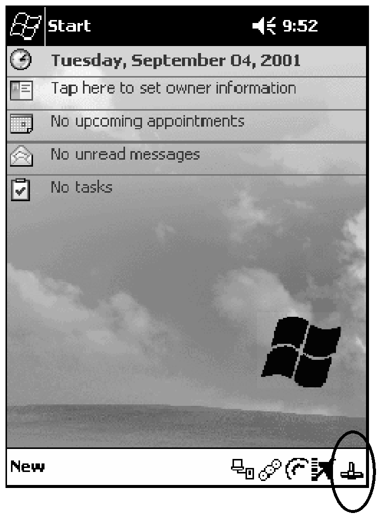
Stopping the FTP Server from Your Application

To allow application programmers the ability to programmatically shut down the FTP Server, the FTP Server periodically tests to see if a named event is signaled. The name for this event is "ITC_IFTP_STOP" (no quotes).

For examples on how to use this event, consult the Microsoft Developer Network Library at *http://www.msdn.com*. The MSDN Library is an essential resource for developers using Microsoft tools, products, and technologies. It contains a bounty of technical programming information, including sample code, documentation, technical articles, and reference guides.

Autostart FTP

This automatically starts the FTP Server (FTPDCE.EXE) when the 700 Series Computer is powered on. This is provided with the NDISTRAY program, which displays the popup menu that currently allows you to load and unload the network drivers. Tap the antenna icon in the System Tray of the Today screen *(a sample antenna icon is circled below)* to get this pop-up menu.



The default is to start the FTP Server at boot time, unless the following registry entry is defined and set to “0” which disables AutoFTP. “1” enables the AutoFTP. The entry can be set from the NDISTRAY pop-up menu by selecting either **AutoFTP On** or **AutoFTP Off**.

```
HKEY_LOCAL_MACHINE\Software\Intermec\Ndistray\StartupIFTP
```

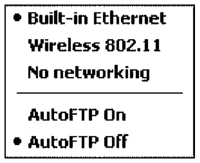
These new entries are located below the selections to load the network drivers. If the StartupIFTP registry key is not defined, the FTP Server is loaded by default, to provide “out-of-the-box” capability for customers who want to begin loading files to the 700 Series Computer without any prior configuration.

Note: If a network driver is unloaded using the NDISTRAY popup menu, and the FTP Server is running, the FTP Server is stopped.

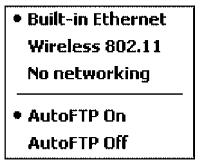
On a resume, if AutoFTP is enabled and the FTP Server is running, it is stopped and restarted. NDISTRAY uses a helper application named RESETIFTP to implement the restart on resume feature. To do an AutoFTP Installation Check:

1 Ensure the FTP Server is running “out-of-the-box” the first time.

2 Tap **Start** → **Today** to access the Today screen, then tap the antenna icon in the System Tray to bring up the NDISTRAY pop-up menu. Select **AutoFTP Off** to disable AutoFTP. Do a warm boot and confirm the FTP Server is not running.



3 Tap **Start** → **Today** to access the Today screen, then tap the antenna icon in the System Tray to bring up the NDISTRAY pop-up menu. Select **AutoFTP On** to enable AutoFTP, reboot and confirm it is running.



4 Unload the network driver when the FTP Server is running and confirm that it is not running any more.

5 Load the FTP Server, establish a connection, then suspend and resume. The server should still be running, but the FTP connection to the client should be dropped.

Full Screen

Pocket PC is a hardware specification created by Microsoft Corporation. Devices that wish to carry the Pocket PC logo must meet the minimum hardware requirements set in the Pocket PC specification. Manufacturers are free to add extra hardware functionality.

Pocket PC 2002 devices also use a specialized version of the CE operating system. This OS is built from Windows CE 3.0 but contains customizations, most notably the lack of a desktop and the addition of the Today Screen.

To carry the Pocket PC logo, all devices must be tested at an Independent Test Laboratory. The ITL testing is done based on Microsoft requirements. The test lab then reports the findings back to Microsoft Corporation and Intermec Technologies. If the 700 Series Computer passed all tests, Intermec is allowed to ship the device with the Pocket PC logo. Each time the operating system is modified, Intermec must resubmit to ITL testing.

This means we cannot change the operating system much and still be a Pocket PC device. For example, if we remove Word from the Start menu, the device would fail ITL testing and we would not be able to ship devices with the Pocket PC logo.

Although many customers want a Pocket PC device, some customers would prefer that their users not have access to all of the Pocket PC features. Intermec cannot customize the operating system in any way but a custom application can:

- Delete items from the Start menu, and Programs folder. These items are just shortcuts in the file system so the application is not really being deleted. Cold booting the device will bring these items back so the application will need to be run on every cold boot.
- Use the RegFlushKey() API to save a copy of the registry to a storage device. See the *Recovery CD Help* for more information on how to do this. Saving a copy of the registry will allow most system settings to be restored in a cold boot situation.
- Use the SHFullScreen() API in conjunction with other APIs to make the application take up the entire display and prevent the start menu from being available.
- Remap keys and disable keys on the keypad.
- Create a custom SIP.
- Make changes to the registry to configure the device.

Should you want your 700 Series Computer to display a full screen, keep in mind that your computer is Pocket-PC certified by Microsoft Corporation. Check out resources on programming for the Pocket PC, using the following links. These instructions give full instructions on how to display full screen.

- Instructions on how to create a full screen application for eVC++ applications using an SHFullScreen() API:
  http://support.microsoft.com/support/kb/articles/Q266/2/44.ASP
- Instructions on how to create a full screen application for eVB applications also using the SHFullScreen() API:
  http://support.microsoft.com/support/kb/articles/Q265/4/51.ASP

Kernel I/O Controls

This describes the KernelIoControl() functions available to application programmers. Most C++ applications will need to prototype the function as the following to avoid link and compile errors.

```
extern "C" BOOL KernelIoControl(DWORD dwIoControlCode, LPVOID lpInBuf, DWORD
nInBufSize, LPVOID lpOutBuf, DWORD nOutBufSize, LPDWORD lpBytesReturned);
```

IOCTL_HAL_GET_DEVICE_INFO

This IOCTL returns either the platform type or the OEMPLATFORM name based on an input value.

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_DEVICE_INFO, LPVOID
lpInBuf, DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Points to a DWORD containing either the SPI_GETPLATFORMTYPE or SPI_GETOEMINFO value.
lpInBufSize	Must be set to sizeof(DWORD).
lpOutBuf	Must point to a buffer large enough to hold the return data of the function. If SPI_GETPLATFORMTYPE is specified in *lpInBuf*, then the "PocketPC\0" Unicode string is returned. If SPI_GETOEMINFO is specified in *lpInBuf*, then the "Intermec 700\0" Unicode string is returned.
nOutBufSize	The size of *lpOutBuf* in bytes. Must be large enough to hold the string returned.
lpBytesReturned	The actual number of bytes returned by the function for the data requested.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_ITC_READ_PARM

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_ITC_READ_PARM,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf Points to this structure. See “*ID Field Values*” below.

```
struct PARMS {
  BYTE  id;
  BYTE  ClassId;
};
```

nInBufSize	Must be set to the size of the PARMS structure.
lpOutBuf	Must point to a buffer large enough to hold the return data of the function. If this field is set to NULL and *nOutBufSize* is set to zero when the function is called the function will return the number bytes required by the buffer.
nOutBufSize	The size of *lpOutBuf* in bytes.
lpBytesReturned	The number of bytes returned by the function for the data requested.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the error value. Either ERROR_INVALID_PARAMETER or ERROR_INSUFFICIENT_BUFFER may be returned when this function is used to get the error.

ID Field Values

The *id* field of the PARMS structure may be one of the following values:

- **ITC_NVPARM_ETHERNET_ID**
  This IOCTL returns the Ethernet 802.11 MAC Address. Six bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_SERIAL_NUM**
  This IOCTL returns the serial number of the device in BCD format. Six bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_MANF_DATE**
  This IOCTL returns the device date of manufacture in the BCD YYYY/MM/DD format. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.

- **ITC_NVPARM_SERVICE_DATE**
  This IOCTL returns the device' s date of last service in BCD YYYY/MM/DD format. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_DISPLAY_TYPE**
  This IOCTL returns the device' s display type. One byte is returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_EDG_IP**
  This IOCTL returns the device Ethernet debug IP address. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_EDBG_SUBNET**
  This IOCTL returns the device Ethernet debug subnet mask. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_ECN**
  This IOCTL returns ECNs applied to the device in a bit array format. Four bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_CONTRAST**
  This IOCTL returns the device default contrast setting. Two bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_MCODE**
  This IOCTL returns the manufacturing configuration code for the device. Sixteen bytes are returned in the buffer pointed to by the *lpOutBuffer* parameter.
- **ITC_NVPARM_VERSION_NUMBER**
  This IOCTL returns the firmware version for various system components. These values for the *ClassId* field of the PARMS structure are allowed when ITC_NVPARM_VERSION_NUMBER is used in the *id* field:
  - **VN_CLASS_KBD**
    Returns a five-byte string, including null terminator, that contains an ASCII value which represents the keyboard microprocessor version in the system. The format of the string is *x.xx* with a terminating null character.
  - **VN_CLASS_ASIC**
    Returns a five-byte string, including null terminator, that contains an ASCII value which represents the version of the FPGA firmware in the system. The format of the string is *x.xx* with a terminating null character.
  - **VN_CLASS_BOOTSTRAP**
    Returns a five-byte string, including null terminator, that contains an ASCII value which represents the version of the Bootstrap Loader firmware in the system. The format of the string is *x.xx* with a terminating null character.

- **ITC_NVPARM_INTERMEC_SOFTWARE_CONTENT**
  This IOCTL reads the manufacturing flag bits from the non-volatile data store that dictates certain software parameters. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates if Intermec Content is enabled in the XIP regions. TRUE indicates that it is enabled. FALSE indicates that it is not enabled.
- **ITC_NVPARM_ANTENNA_DIVERSITY**
  This IOCTL reads the state of the antenna diversity flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates if there is a diversity antenna installed. TRUE indicates that it is installed. FALSE indicates that it is not installed.
- **ITC_NVPARM_WAN_RI**
  This IOCTL reads the state of the WAN ring indicator flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer* that indicates the polarity of the WAN RI signal. TRUE indicates active high. FALSE indicates active low.
- **ITC_NVPARM_RTC_RESTORE**
  This IOCTL reads the state of the real-time clock restore flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the RTC will be restored upon a cold boot. FALSE indicates that the RTC will not be restored.
- **ITC_NVPARM_INTERMEC_DATACOLLECTION_SW**
  This IOCTL reads the state of the data collection software enabled flag. A BOOLEAN DWORD is returned in the buffer pointer to by *lpOutBuffer* that indicates the data collection software is to be installed at boot time. FALSE indicates the data collection software should not be installed.
- **ITC_NVPARM_INTERMEC_DATACOLLECTION_HW**
  This IOCTL reads the data collection hardware flags. A BYTE is returned in the buffer pointer to by *lpOutBuffer* that indicates the type of data collection hardware installed. The maximum possible value returned is ITC_DEVID_SCANHW_MAX.
  - **ITC_DEVID_SCANHW_NONE**
    No scanner hardware is installed.
  - **ITC_DEVID_OEM2D_IMAGER**
    OEM 2D imager is installed.
  - **ITC_DEVID_INTERMEC2D_IMAGER**
    Intermec 2D imager is installed.
  - **ITC_DEVID_SE900_LASER**
    SE900 laser is installed.
  - **ITC_DEVID_SE900HS_LASER**
    SE900HS laser is installed.

  The high bit indicates whether the S6 scanning engine is installed. The bit mask for this is ITC_DEVID_S6ENGINE_MASK. A non-zero value indicates that the S6 scanning engine is installed.

- **ITC_NVPARM_WAN_INSTALLED**
  This IOCTL reads the state of the WAN radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the WAN radio is installed. FALSE indicates that no WAN radio is installed.
- **ITC_NVPARM_WAN_FREQUENCY**
  This IOCTL reads the state of the WAN radio frequency flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the WAN radio frequency is United States. FALSE indicates that the WAN radio frequency is European.
- **ITC_NVPARM_WAN_RADIOTYPE**
  This IOCTL reads the WAN radio ID installed by manufacturing. A BYTE is returned in the buffer pointer to by *lpOutBuffer* which indicates the type of WAN radio hardware installed. The maximum possible value returned is ITC_DEVID_WANRADIO_MAX. The current definitions are:
  - **ITC_DEVID_WANRADIO_NONE**
    No WAN radio installed.
  - **ITC_DEVID_WANRADIO_SIERRA_SB555**
    CDMA Sierra Wireless radio.
  - **ITC_DEVID_WANRADIO_XIRCOM_GEM3503**
    GSM/GPRS Intel (Xircom) radio.
  - **ITC_DEVID_WANRADIO_SIEMENS_MC45**
    GSM/GPRS Siemens radio.
- **ITC_NVPARM_80211_INSTALLED**
  This IOCTL reads the state of the 802.11b radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the 802.11b radio is installed. FALSE indicates that no 802.11b radio is installed.
- **ITC_NVPARM_80211_RADIOTYPE**
  This IOCTL reads the 802.11b radio ID installed by manufacturing. A BYTE is returned in the buffer pointer to by *lpOutBuffer* that indicates the type of 802.11b radio hardware installed. The maximum possible value returned is ITC_DEVID_80211RADIO_MAX. The current definitions are:
  - **ITC_DEVID_80211RADIO_NONE**
    No 802.11b radio installed.
  - **ITC_DEVID_80211RADIO_INTEL_2011B**
    Intel 2011B radio installed.
- **ITC_NVPARM_BLUETOOTH_INSTALLED**
  This IOCTL reads the state of the Bluetooth radio installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the Bluetooth radio is installed. FALSE indicates that no Bluetooth radio is installed.

- **ITC_NVPARM_SERIAL2_INSTALLED**
  This IOCTL reads the state of the serial 2 (COM2) device installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the serial 2 device is installed. FALSE indicates that no serial 2 device is installed.
- **ITC_NVPARM_VIBRATE_INSTALLED**
  This IOCTL reads the state of the vibrate device installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the vibrate device is installed. FALSE indicates that no vibrate device is installed.
- **ITC_NVPARM_LAN9000_INSTALLED**
  This IOCTL reads the state of the Ethernet device installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the Ethernet device is installed. FALSE indicates that no Ethernet device is installed.
- **ITC_NVPARM_SIM_PROTECT_HW_INSTALLED**
  This IOCTL reads the state of the SIM card protection hardware installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection hardware is installed. FALSE indicates that no SIM card protection hardware is installed.
- **ITC_NVPARM_SIM_PROTECT_SW_INSTALLED**
  This IOCTL reads the state of the SIM card protection software installed flag. A BOOLEAN DWORD is returned in the buffer pointed to by *lpOutBuffer*. TRUE indicates that the SIM card protection software is installed. FALSE indicates that no SIM card protection software is installed.

IOCTL_HAL_ITC_WRITE_SYSPARM

Describes and enables the registry save location.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_ITC_WRITE_SYSPARM,LPVOID
lpInBuf,DWORD nInBufSize, LPVOID lpOutBuf, DWORD
nOutBufSize, LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	A single byte that may be one of the *id* values. See “*ID Field Values*” below.
nInBufSize	Must be set to the size of the *lpInBuf* in bytes.
lpOutBuf	Must point to a buffer large enough to hold the data to be written to the non-volatile data store.
nOutBufSize	The size of *lpOutBuf* in bytes.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the error value. Either ERROR_INVALID_PARAMETER or ERROR_INSUFFICIENT_BUFFER may be returned when this function is used to get the error.

ID Field Values

The *id* field of *lpInBuf* may be one of the following values:

- ITC_REGISTRY_LOCATION
  This IOCTL sets the default location for where to write the registry when RegFlushKey() is called by an application. The registry may be saved to Flash, a CompactFlash storage card or a SecureDigital storage card. *lpOutBuf* must point to a buffer that contains a byte value of “1” for the CompactFlash card or “2” for the SecureDigital card to specify the location.
- ITC_REGISTRY_SAVE_ENABLE
  This function enables or disables the save registry to non-volatile media feature of the RegFlushKey() function. *lpOutBuf* must be set to zero (FALSE) if the feature is to be disabled or one (TRUE) if the feature is to be enabled.
- ITC_ DOCK_SWITCH
  This IOCTL sets a position of the dock switch. The dock switch may be set to either “modem” or “serial” positions. *lpOutBuf* must point to a buffer that contains a byte value of either DOCK_MODEM or DOCK_SERIAL as defined in OEMIOCTL.H; the value specifies the position the switch is to be set.

- **ITC_ WAKEUP_MASK**
  This IOCTL sets a bit mask that represents the mask for the five programmable wakeup keys. The I/O key is not a programmable wakeup key. By default it is always the system resume key and all other keys are set to disable key wakeup. A zero in a bit position masks the wakeup for that key. A one in a bit position enables wakeup for that key. *lpOutBuf* must point to a buffer that contains a byte value of a wakeup mask consisting of the OR'ed constants as defined in OEMIOCTL.H. Only the following keys are programmable as wakeup events.

```
#define   SCANNER_TRIGGER 1
#define   SCANNER_LEFT    2
#define   SCANNER_RIGHT   4
#define   GOLD_A1        8
#define   GOLD_A2        0x10
```

- **ITC_AMBIENT_KEYBOARD**
  This IOCTL sets the threshold for the keyboard ambient sensor. This can be a value from 0 (always off) to 255 (always on). *lpOutBuf* must point to a buffer that contains a byte value of the desired setting.
- **ITC_AMBIENT_FRONTLIGHT**
  This IOCTL sets the threshold for the frontlight ambient sensor. This can be a value from 0 (always off) to 255. *lpOutBuf* must point to a buffer that contains a byte value of the desired setting.

IOCTL_HAL_GET_DEVICEID

This IOCTL returns the device ID. There are two types of device IDs supported, which are differentiated based on the size of the *output* buffer. The UUID is returned if the buffer size is set to *sizeof(UNIQUE_DEVICEID)*, otherwise the oldstyle device ID is returned.

Usage

#include “pkfuncs.h”
#include “deviceid.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_DEVICEID,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL. STRICT_ID settings are not supported.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a UNIQUE_DEVICEID structure as defined by DEVICEID.H if the UUID is to be returned.
nOutBufSize	The size of the UNIQUE_DEVICEID in bytes if the UUID is to be returned. A DEVICE_ID as defined by PKFUNCS.H is returned if the size in bytes is greater than or equal to *sizeof(DEVICE_ID)*.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_GET_OAL_VERINFO

Returns the HAL version information of the Pocket PC image.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_OAL_VERINFO,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a VERSIONINFO structure as defined by OEMIOCTL.H. The fields should have these values:
nOutBufSize	The size of VERSIONINFO in bytes.
lpBytesReturned	Returns *sizeof(PVERSIONINFO)*.

Fields of the VERSIONINFO structure for *lpOutBuf*:

• `cboemverinfo`	sizeof (tagOemVerInfo);
• `verinfover`	1
• `sig;`	“ITC\0”
• `id;`	‘N’
• `tgtcustomer`	“”
• `tgtplat`	SeaRay
• `tgtplatversion`	Current build version number
• `tgtcputype[8];`	“Intel\0”
• `tgtcpu`	“PXA250\0”;
• `tgtcoreversion`	“”
• `date`	Build time
• `time`	Build date

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_GET_BOOTLOADER_VERINFO

Returns the HAL version information of the Pocket PC image.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_OAL_VERINFO,LPVOID
lpInBuf, DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a VERSIONINFO structure as defined by OEMIOCTL.H. The fields should have these values:

• `cboemverinfo`	Sizeof (tagOemVerInfo);
• `verinfover`	1
• `sig;`	“ITC\0”
• `id;`	‘ B’
• `tgtcustomer`	“”
• `tgtplat`	SeaRay
• `tgtplatversion`	Current build version number of the bootstrap loader
• `tgtcputype[8];`	“Intel\0”;
• `tgtcpu`	“PXA250\0”
• `tgtcoreversion`	“”
• `date`	Build time
• `time`	Build date

nOutBufSize	The size of VERSIONINFO in bytes.
lpBytesReturned	The number of bytes returned to *lpOutBuf.*

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_WARMBOOT

Causes the system to perform a warm-boot. The object store is retained.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_WARMBOOT,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_HAL_COLDBOOT

Causes the system to perform a cold-boot. The object store is cleared.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_COLDBOOT,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_HAL_GET_RESET_INFO

This IOCTL code allows software to check the type of the most recent reset.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_RESET_INFO,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a HAL_RESET_INFO structure:

```
typedef struct {
  DWORD ResetReason;                     // most recent reset type
  DWORD ObjectStoreState;                // state of object store
} HAL_RESET_INFO, * PHAL_RESET_INFO;

// Reset reason types
#define HAL_RESET_TYPE_UNKNOWN          0
#define HAL_RESET_REASON_HARDWARE       1  // cold
#define HAL_RESET_REASON_SOFTWARE       2  // suspend
#define HAL_RESET_REASON_WATCHDOG       4
#define HAL_RESET_BATT_FAULT            8  // power fail
#define HAL_RESET_VDD_FAULT             16 // warm boot

// Object store state flags
#define HAL_OBJECT_STORE_STATE_UNKNOWN  0
#define HAL_OBJECT_STORE_STATE_CLEAR    1
```

nOutBufSize	The size of HAL_RESET_INFO in bytes.
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_GET_BOOT_DEVICE

This IOCTL code allows software to check which device CE booted from.

Usage

#include "oemioctl.h"

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_GET_BOOT_DEVICE,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Must point to a buffer large enough to hold a DWORD (4 bytes) that contains the boot device. The following boot devices are supported:

```
#define HAL_BOOT_DEVICE_UNKNOWN        0
#define HAL_BOOT_DEVICE_ROM_XIP        1
#define HAL_BOOT_DEVICE_ROM            2
#define HAL_BOOT_DEVICE_PCMCIA_ATA     3
#define HAL_BOOT_DEVICE_PCMCIA_LINEAR  4
#define HAL_BOOT_DEVICE_IDE_ATA        5
#define HAL_BOOT_DEVICE_IDE_ATAPI      6
```

nOutBufSize	The size of lpOutBuf in bytes (4).
lpBytesReturned	The number of bytes returned by the function.

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_HAL_REBOOT

Causes the system to perform a warm-boot. The object store is retained.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_HAL_REBOOT,LPVOID lpInBuf,DWORD
nInBufSize,LPVOID lpOutBuf,DWORD nOutBufSize,LPDWORD
lpBytesReturned );
```

Parameters

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be NULL.
nOutBufSize	Should be zero.

Return Values

None.

IOCTL_PROCESSOR_INFORMATION

Returns processor information.

Usage

#include “pkfuncs.h”

Syntax

```
BOOL KernelIoControl( IOCTL_PROCESSOR_INFORMATION,LPVOID
lpInBuf,DWORD nInBufSize,LPVOID lpOutBuf,DWORD
nOutBufSize,LPDWORD lpBytesReturned );
```

Parameters

Parameters:

lpInBuf	Should be set to NULL.
lpInBufSize	Should be set to zero.
lpOutBuf	Should be a pointer to the PROCESSOR_INFO structure. The PROCESSOR_INFO structure stores information that describes the CPU more descriptively.

```
typedef __PROCESSOR_INFO {
WORD    wVersion;              // Set to value 1
WCHAR   szProcessorCore[40];   // "ARM\0"
WORD    wCoreRevision;         // 4
WCHAR   szProcessorName[40];   // "PXA250\0"
WORD    wProcessorRevision;    // 0
WCAHR   szCatalogNumber[100];  // 0
WCHAR   szVendor[100];         // "Intel Corporation\0"
DWORD   dwInstructionSet;      // 0
DWORD   dwClockSpeed;          // 400
}
```

nOutBufSize	Should be set to sizeof(PROCESSOR_INFO) in bytes.
lpBytesReturned	Returns sizeof(PROCESSOR_INFO);

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

IOCTL_GET_CPU_ID

Returns Xscale processor ID.

Usage

#include “oemioctl.h”

Syntax

```
BOOL KernelIoControl( IOCTL_GET_CPU_ID,LPVOID lpInBuf, DWORD
nInBufSize,LPVOID lpOutBuf,DWORD nOutBufSize,LPDWORD
lpBytesReturned );
```

Parameters

lpInBuf	Should point to a CPUIdInfo structure defined in OEMIOCTL.H.
lpInBufSize	Should be sizeof(CPUIdInfo).
lpOutBuf	Should be NULL.
nOutBufSize	Should be set to 0.
lpBytesReturned	Returns sizeof(PROCESSOR_INFO);

Return Values

Returns TRUE if function succeeds. Returns FALSE if the function fails. GetLastError() may be used to get the extended error value.

Reboot Functions

There are several methods, via Kernel I/O Control functions, that an application program can use to force the 700 Series Computer to reboot.

IOCTL_HAL_REBOOT

IOCTL_HAL_REBOOT performs a warm-boot. See page 278.

IOCTL_HAL_COLDBOOT

Invoking the KernelIOControl function with IOCTL_HAL_COLDBOOT forces a cold reboot. This resets the 700 Series Computer and reloads Windows CE as if a power-up had been performed. The contents of the Windows CE RAM-based object store are discarded. See page 275.

IOCTL_HAL_WARMBOOT

This function is supported on the 700 Series Computers. It performs a warm boot of the system, preserving the object store. See page 275.

Remapping the Keypad

Note; Use caution when remapping the keypad. Improper remapping may render the keypad unusable. Data within the 700 Series Computer could also be lost, should any problems occur.

Applications have the ability to remap keys on the 700 Color Keypad. This will allow applications to enable keys that would otherwise not be available, such as the [F1] function key. Also, to disable keys that should not be available, such as the alpha key because no alpha entry is required. Care should be exercised when attempting to remap the keypad because improper remapping may cause the keypad to become unusable. This can be corrected by cold booting the device which will cause the default keymap to be loaded again.

Note that remapping the keys in this way affects the key mapping for the entire system, not just for the application that does the remapping.

There are three “planes” supported for the 740 Keypad. Keys that are to be used in more than one shift plane must be described in each plane.

Unshifted Plane

The unshifted plane contains values from the keypad when not pressed with other keys, such as the following:

• [1]	1
• [5]	5
• [9]	9

Gold Plane

The gold plane contains values from the keypad when a key is simultaneously pressed with the [**Gold**] key, such as the following:

• [**Gold**] + [1]	Send
• [**Gold**] + [5]	A3
• [**Gold**] + [9]	PageDown

Alpha Plane

The alpha plane contains values from the keypad when the keypad has been placed in alpha mode by pressing the blue alpha key, such as the following:

• [**Alpha**] + [1]	Caps
• [**Alpha**] + [5]	JKL
• [**Alpha**] + [9]	WXYZ

Key Values

Key values for each plane are stored in the registry. All units ship with a default key mapping already loaded in the registry. Applications that wish to change the default mapping need to read the appropriate key from the registry into an array of Words, modify the values required and then write the updated values back into the registry. The registry access can be done with standard Microsoft API calls, such as RegOpenKeyEx(), RegQuery-ValueEx(), and RegSetValueEx().

- The unshifted plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\Vkey`
- The gold plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\VkeyGold`
- The alpha plane mapping can be found in the registry at:
  `HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD\VkeyAlpha`

How Key Values Are Stored in Registry

To know which fields to update in the registry, you must know what Scan Codes are assigned to each physical key (see the table below). The Scan Code is used at the lowest level of the system to let the keypad driver know which physical key has been pressed. The keypad driver takes that scan code and looks it up in a table (a copy of the one stored in the registry) to determine which values to pass on to the operating system.

Each registry key is just an array that describes to the keypad driver what value needs to be passed for each physical key. The key values are indexed by the scan code, this is a zero-based index. For example in the unshifted plane, the [4] key has a scan code of 0x06. This means that the seventh word under the "Vkey" registry key will have the value for the [4] key. Taking a sample of the "Vkey" registry key shows the following values:

```
00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,. . .
```

The value is 34,00. The values are in reverse byte order because that is the way the processor handles data. When writing an application, nothing needs to be done to swap the bytes, as this will happen automatically when the data is read into a byte value. This is something you just need to be aware of this when looking at the registry. Knowing this, we can see that the value that the keypad driver will pass to the system is a hex 34. Looking that up on an UNICODE character chart, we see that it maps to a "4". If you wanted the key, labeled "4", to output the letter "A" instead, you would need to change the seventh word to "41" (the hexadecimal representation of "A" from the UNICODE chart), then put the key back into the registry.

Note: Do not remap scan codes 0x01, 0x41, 0x42, 0x43, 0x44. Remapping these scan codes could render your 700 Series Computer unusable until a cold-boot is performed.

If you wish to disable a certain key, remap its scan code to 0x00.

Change Notification

Just changing the registry keys will not immediately change the key mappings. To notify the keypad driver that the registry has been updated, signal the "ITC_KEYBOARD_CHANGE" named event using the CreateEvent() API.

Advanced Keypad Remapping

It is also possible to map multiple key presses to one button and to map named system events to a button. The multiple key press option could be useful to cut down on the number of keys needed to press in a given situation or to remap which key behaves like the action key. Mapping events to a button could be useful to change which buttons will fire the scanner, control volume, and allow for suspending and resuming the device. If you need help performing one of these advanced topics please contact Intermec Technical Support.

Scan Codes

At the lowest driver level, the 740 Keypad identifies keys as scan codes. These scan codes are sent via the keypad microcontroller, and cannot be changed without modifying the keypad firmware.

Key/Meaning	Scancode
Reserved	0x00
I/O Button	0x01
Scanner Trigger	0x02
Scanner Left	0x03
Scanner Right	0x04
.	0x05
4	0x06
None	0x07
Left Arrow	0x08
None	0x09
Backspace	0x0A
Gold Key	0x0B
None	0x0C
ESC	0x0D
Down Arrow	0x0E
1	0x0F
7	0x10
Alpha Key	0x11
None	0x12
Up Arrow	0x13
Right Arrow	0x14
2	0x15
8	0x16
0	0x17
5	0x18
None	0x19

Key/Meaning	Scancode
Action Key	0x1A
3	0x1B
9	0x1C
ENTER	0x1D
6	0x1E
None	0x1F-0x40
Charge Detect	0x41
LCD Frontlight	0x42
Ambient Light	0x42
Threshold Crossed	0x42
Headset Detected	0x43
Keypad Backlight	0x44
Ambient Light	0x44
Threshold Crossed	0x44

Sample View of Registry Keys

The following is a sample view of the current default key mapping. See the registry on your device for the latest key mappings.

```
[HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\KEYBD]
"ResumeMask"=dword:7
"Vkey"=hex: 00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,\
   25,00,00,00,08,00,03,02,00,00,1B,00,28,00,31,00,\
   37,00,01,02,00,00,26,00,27,00,32,00,38,00,30,00,\
   35,00,00,00,01,03,33,00,39,00,0D,00,36,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,07,05,01,05,03,05,02,05

"VkeyGold"=hex:  00,00,0B,05,02,03,C1,07,04,03,BE,00,34,00,00,00,\
   09,01,00,00,BF,00,03,02,00,00,BD,00,75,00,72,00,\
   21,00,01,02,00,00,76,00,09,00,73,00,38,01,5B,00,\
   35,00,00,00,BB,01,09,05,22,00,32,01,36,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,07,05,01,05,03,05,02,05

"VkeyAlpha"=hex: 00,00,0B,05,02,03,C1,07,04,03,BE,00,47,00,00,00,\
   25,00,00,00,08,00,03,02,00,00,1B,00,28,00,02,02,\
   50,00,01,02,00,00,26,00,27,00,41,00,54,00,20,00,\
   4A,00,00,00,01,03,44,00,57,00,0D,00,4D,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
   00,00,07,05,01,05,03,05,02,05
```

A Control Panel Applets

This appendix contains information about the Data Collection, SNMP, and User Information Control Panel applets that may be on your 700 Series Color Mobile Computer.

SNMP and Data Collection settings that can appear under **Settings** are dependent on what hardware configuration is done for each 700 Series Computer at the time of shipment. These settings will currently only appear if a scanner or an imager option is present.

Likewise, other control panel applets that are specifically related to the 802.11b radio module will appear when a 802.11b radio module is installed in a 700 Series Computer. Control panel applets that are specific for Wireless Printing, CDMA/1xRTT, and GSM/GPRS radio modules will only appear when each respective hardware configuration is done on the 700 Series Computer. *See Chapter 4, "Network Support," for more information about the radio modules or the wireless printing.*

Configuration Parameters

A configuration parameter changes the way the 700 Series Color (700C) Mobile Computer operates, such as configuring a parameter to have the 700 Series Computer emit a very loud beep in a noisy environment. Use any of the following methods to execute configuration parameters:

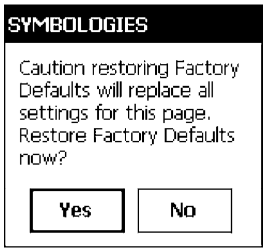
- Change Data Collection and SNMP parameters via control panel applets later in this appendix.
- Access the 700 Series Computer via the Unit Manager through a web browser on your desktop PC via the SRDEVMGMT.CAB file. To use the Unit Manager, install this CAB file from the 700 Color Software Tools CD-ROM. Unit Manager applications are available on the *700 Series Color Unit Manager CD-ROM*. For more information, consult your Intermec sales representative.
- Send parameters from an SNMP management station. See "*SNMP Configuration*" starting on page 123.
- Scan EasySet bar codes. You can use the EasySet bar code creation software from Intermec Technologies Corporation to print configuration labels. Scan the labels to change the scanner configuration and data transfer settings.

Changing a Parameter Setting

Menus of available parameters for each group are listed. Use the scroll bars to go through the list. Expand each menu (+) to view its parameter settings. Tap a parameter to select, or expand a parameter (+) to view its subparameters.

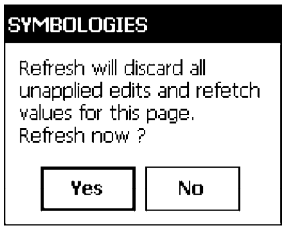
Note that each parameter or subparameter is shown with its default setting or current setting in (< >) brackets. Tap a parameter or subparameter to select that parameter, then do any of the following to change its setting: Tap **Apply** to apply any changes. *Note that these illustrations are from a Symbologies parameter.*

- Typing a new value in an entry field.
- Choosing a new value from the drop-down list.
- Selecting a different option. The selected option contains a bullet.
- Tap **Defaults**, then **Apply** to restore factory-default settings. Tap **Yes** when you are prompted to verify this action.


SYMBOLOGIES

Caution restoring Factory Defaults will replace all settings for this page. Restore Factory Defaults now?

- Tap **Refresh** to discard changes and start again. Tap **Yes** when you are prompted to verify this action.


SYMBOLOGIES

Refresh will discard all unapplied edits and refetch values for this page. Refresh now ?

About Configuration Parameters

You can find the following information about each configuration parameter:

- **Name and Purpose**:
  Describes the parameter and its function.
- **Action**:
  Describes what to do with a parameter once that parameter is selected.
- **SNMP OID**:
  Lists the SNMP OID for the parameter.
- **Syntax** or **Options**:
  Syntax lists the two-character code for the parameter, if the parameter is configurable by scanning a bar code or by sending parameters through a network. Both **Syntax** and **Options** list acceptable values for the parameter. *Default settings are noted in italic.*

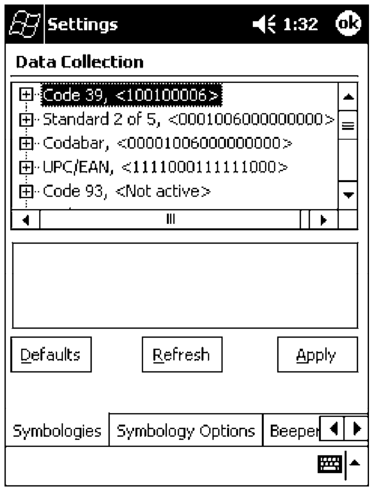
Data Collection Control Panel Applet

See "*Scanner Control and Data Transfer*" in the *Intermec Windows CE/Pocket PC Software Developer's Kit (SDK) User's Manual* shipped with the Software Developer's Kit (SDK) for information about data collection functions.

Note: Icons are shown to the left.

Data Collection

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Data Collection** icon to access its control panel applet.



Code 39, <100100006>
Standard 2 of 5, <0001006000000000>
Codabar, <000010060000000000>
UPC/EAN, <1111000111111000>
Code 93, <Not active>

Use the left and right arrows to scroll through the tabs along the bottom of the control panel applet, then tap a tab to access its menus. These tabs represent the following groups of settings or parameters:

- **Symbologies**
- **Symbology Options** *(starting on page 309)*
- **Beeper/LED** *(starting on page 317)*
- **Imager** *(starting on page 323)*
- **Virtual Wedge** *(starting on page 325)*

Symbologies

You can change bar code symbology parameter settings in your 700 Series Computer via the **Data Collection** control panel applet. The following parameters are for bar code symbologies. Additional information about the more common bar code symbologies are in Appendix C, "*Bar Code Symbologies.*" *Note that these parameters are listed in the order of their appearance within this tab.*

Most of these symbologies apply to both the imager and the laser scanner tools. However, when using an imager, the Macro PDF *(page 300)*, Micro PDF 417 *(page 302)*, Matrix 2 of 5 *(page 304)*, Telepen *(page 305)*, and Code 11 *(page 306)* symbologies are not supported. Likewise, when using a laser scanner, the QR Code *(page 307)* and Data Matrix *(page 308)* symbologies are not supported.

The following table shows which bar code symbologies are supported either by an imager or by a laser scanner.

Bar Code Symbology	Imager	Laser Scanner
Code 39	X	X
Interleaved 2 of 5	X	X
Standard 2 of 5	X	X
Matrix 2 of 5		X
Code 128	X	X
Code 93	X	X
Codabar	X	X
MSI		X
Plessey		X
UPC	X	X
EAN/EAN 128	X	X
Code 11		X
PDF 417	X	X
Micro PDF 417		X
Telepen		X
Data Matrix	X	
QR Code	X	

Code 39

Code 39 is a discrete, self-checking, variable length symbology. The character set is uppercase A-Z, 0-9, dollar sign ($), period (.), slash (/), percent (%), space ( ), plus (+), and minus (-).

Action

Tap (+) to expand the **Code 39** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

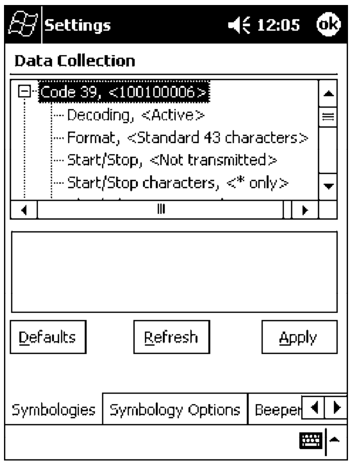
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.3.1

Options

Decoding	0	Not active
	1	Active *(default)*
Format	0	Standard 43 characters *(default)*
	1	Full ASCII
Start/Stop	0	Not transmitted *(default)*
	1	Transmitted
Start/Stop characters *(Not supported when using an imager)*:		
	0	$ (dollar sign) only
	1	* (asterisk) only *(default)*
	2	& and * (dollar sign and asterisk)
Check digit	0	Not used *(default)*
	1	Mod 43 transmitted
	2	Mod 43 not transmitted
	3	French CIP transmitted
	4	French CIP not transmitted
	5	Italian CPI transmitted
	6	Italian CPI not transmitted
Bar code length	0	Any length *(default)*
	1	Minimum length
Minimum length	000-254	Minimum length 1-254 *(6)*

Note: If **Bar code length** = “1” then **Minimum length** is entered.



Data Collection

Code 39, <100100006>
Decoding, <Active>
Format, <Standard 43 characters>
Start/Stop, <Not transmitted>
Start/Stop characters, <* only>

Standard 2 of 5

Standard 2 of 5 is a discrete and self-checking symbology that uses the bars to encode information and the spaces to separate the individual bars.

Action

Tap (+) to expand the **Standard 2 of 5** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

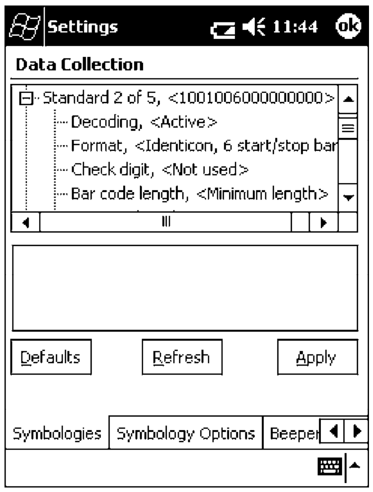
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.4.1

Options

Decoding	0	Not active *(default)*
	1	Active
Format	0	Identicon, 6 start/stop bars *(default)*
	1	Computer Identics, 4 start/stop
Check digit	0	Not used *(default)*
	1	Mod 10 transmitted
	2	Mod 10 not transmitted
Bar code length	0	Any length
	1	Minimum length *(default)*
	2	Fixed lengths
Minimum length	001-254	Minimum length 1-254 *(6)*
Fixed length 1	000-254	Fixed bar code length 0-254 *(0)*
Fixed length 2	000-254	Fixed bar code length 0-254 *(0)*
Fixed length 3	000-254	Fixed bar code length 0-254 *(0)*

Note: If **Bar code length** = "1" then **Minimum length** is entered. If **Bar code length** = "2" then **Fixed length 1**, **Fixed length 2**, or **Fixed length 3** is entered.



Standard 2 of 5, <1001006000000000>
Decoding, <Active>
Format, <Identicon, 6 start/stop bar
Check digit, <Not used>
Bar code length, <Minimum length>

Codabar

Codabar is a self-checking, discrete symbology.

Action

Tap (+) to expand the **Codabar** parameter, select a setting to be changed, then select an option from the drop-down list to change this setting.

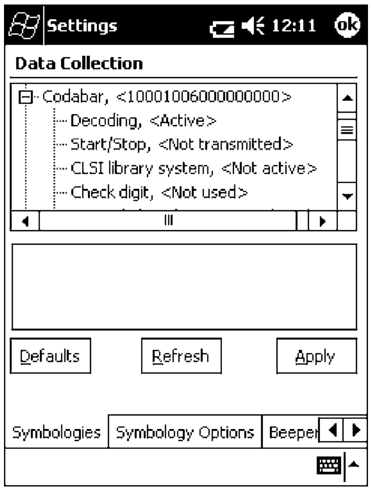
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.5.1

Options

Decoding	0	Not active *(default)*
	1	Active
Start/Stop	0	Not transmitted *(default)*
	1	abcd transmitted
	2	ABCD transmitted
	3	abcd/tn*e transmitted
	4	DC1`DC4 transmitted
CLSI library system *(Not supported when using an imager)*:		
	0	Not active *(default)*
	1	Active
Check digit	0	Not used *(default)*
	1	Transmitted
	2	Not transmitted
Bar code length	0	Any length
	1	Minimum length *(default)*
	2	Fixed lengths
Minimum length	003-254	Minimum length 3-254 *(6)*
Fixed length 1	000-254	Fixed length 0-254 *(0)*
Fixed length 2	000-254	Fixed length 0-254 *(0)*
Fixed length 3	000-254	Fixed length 0-254 *(0)*

Note: If **Bar code length** = "1" then **Minimum length** is entered. If **Bar code length** = "2" then **Fixed length 1**, **Fixed length 2**, or **Fixed length 3** is entered.



Data Collection

Codabar, <1000100600000000>
Decoding, <Active>
Start/Stop, <Not transmitted>
CLSI library system, <Not active>
Check digit, <Not used>

UPC/EAN

UPC/EAN are fixed-length, numeric, continuous symbologies that use four element widths.

Action

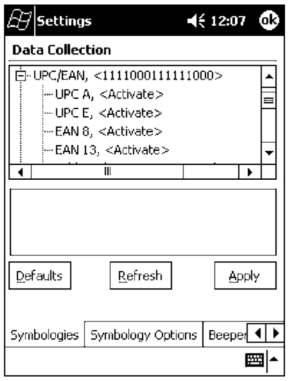
Tap (+) to expand the **UPC/EAN** parameter, select the setting to be changed, then select an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.6.1

Options

UPC A	0	Not active
	1	Active *(default)*
UPC E	0	Not active
	1	Active *(default)*
EAN 8	0	Not active
	1	Active *(default)*
EAN 13	0	Not active
	1	Active *(default)*
Add-on digits	0	Not required *(default)*
	1	Required
Add-on 2 digits	0	Not active *(default)*
	1	Active
Add-on 5 digits *(Not supported when using an imager)*:		
	0	Not active *(default)*
	1	Active
UPC A check digit	0	Not transmitted
	1	Transmitted *(default)*
UPC E check digit	0	Not transmitted
	1	Transmitted *(default)*
EAN 8 check digit	0	Not transmitted
	1	Transmitted *(default)*
EAN 13 check digit	0	Not transmitted
	1	Transmitted *(default)*
UPC A number system	0	Not transmitted
	1	Transmitted *(default)*
UPC E number system	0	Not transmitted
	1	Transmitted *(default)*
UPC A re-encoding	0	UPC A transmitted as EAN 13 *(default)*
	1	UPC A transmitted as UPC A
UPC E re-encoding	0	UPC E transmitted as UPC E *(default)*
	1	UPC E transmitted as UPC A
EAN 8 re-encoding	0	EAN 8 transmitted as EAN 8 *(default)*
	1	EAN 8 transmitted as EAN 13



Data Collection

UPC/EAN, <1111000111111000>
UPC A, <Activate>
UPC E, <Activate>
EAN 8, <Activate>
EAN 13, <Activate>

Code 93

Code 93 is a variable length, continuous symbology that uses four element widths.

Action

Tap the **Code 93** parameter, then select an option to change this parameter setting. Tap (+) to access the **Code 93 Lengths** parameter.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.7.1

Options

0	Not active *(default)*
1	Active

Code 93 Length

Sets the Code 93 bar code length.

Action

Tap (+) to expand the **Code 93** parameter, then tap (+) to expand the **Code 93 Lengths** parameter. Tap the setting to be changed, then tap an option to change this setting.

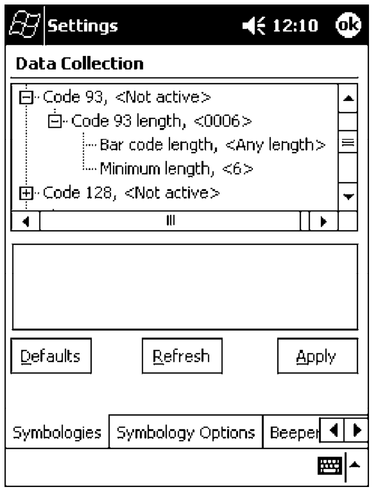
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.19.1

Options

Bar code length	0	Any length *(default)*
	1	Minimum length
Minimum length	001-254	Minimum length 1-254 *(6)*

Note: If **Bar code length** = "1" then **Minimum length** is entered.



Code 93, <Not active>
Code 93 length, <0006>
Bar code length, <Any length>
Minimum length, <6>
Code 128, <Not active>

Code 128

Code 128 is a variable-length, continuous, high-density, alphanumeric symbology that uses multiple element widths and supports the extended ASCII character set.

Action

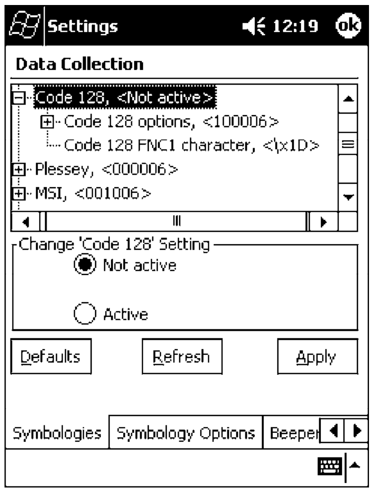
Tap the **Code 128** parameter, then select an option to change this parameter setting. *The following illustration is for a 700 Series Computer using a laser scanner.*

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.9.1

Options

0	Not active *(default)*
1	Active



Settings 12:19

Data Collection

Code 128, <Not active>
Code 128 options, <100006>
Code 128 FNC1 character, <\x1D>
Plessey, <000006>
MSI, <001006>

Change 'Code 128' Setting
[✓ Not active] Active

Code 128 Options

Set the following for the Code 128 parameter. *Note that the **EAN 128]C1** and **CIP 128 French Pharmaceutical** options are not available when you use an imager with your 700 Series Computer.*

Action

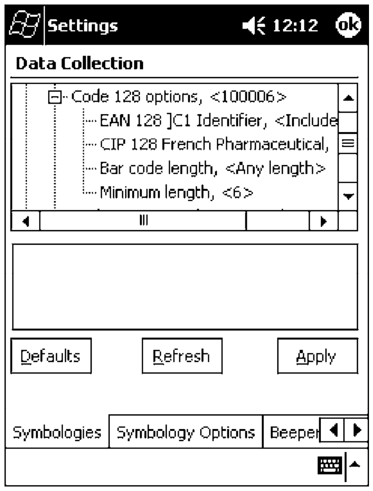
Tap (+) to expand the **Code 128 Options** parameter, select a setting, then select an option to change this setting.

SNMP OID

None.

Options

EAN 128]C1 Identifier *(disabled when using an imager)*		
	0	Remove
	1	Include *(default)*
CIP 128 French Pharmaceutical *(disabled when using an imager)*		
	0	Not active *(default)*
	1	Active
Bar code length	0	Any length *(default)*
	1	Minimum length
Minimum length	001-254	Minimum length 1-254 *(6)*


Settings 12:12
Data Collection
Code 128 options, <100006>
EAN 128]C1 Identifier, <Include
CIP 128 French Pharmaceutical,
Bar code length, <Any length>
Minimum length, <6>

Code 128 FNC1 Character

The Code 128 FNC1 character (EAN 128 norms) can be any ASCII character and is used as a separator when multiple identifiers and their fields are concatenated. *Note that this is not available when you use an imager with your 700 Series Computer.*

Non-printable ASCII characters can be entered using the following syntax where *HH* is the hexadecimal value of the character.

```
\xHH
```

For example, the GS character, whose hexadecimal value is 1D, would be entered as `\x1D`. In addition,the following characters have their own identifiers:

- BEL \a
- BS \b
- FF \f
- LF \n
- CR \r
- HT \t
- VT \v

Action

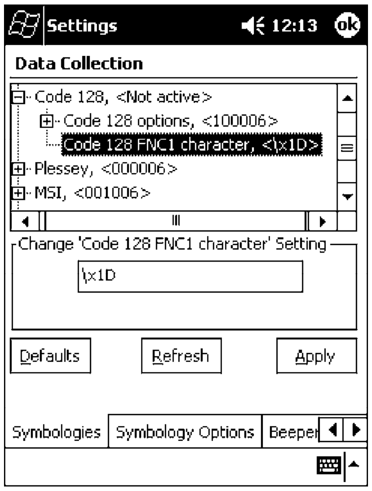
Tap (+) to expand the **Code 128** parameter, then type the ASCII characters to be set for the **Code 128 FNC1 character** parameter.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.21.1

Options

Any ASCII character *(default is the GS function character - ID hex)*



Code 128, <Not active>
Code 128 options, <100006>
Code 128 FNC1 character, <\x1D>
Plessey, <000006>
MSI, <001006>
Change 'Code 128 FNC1 character' Setting
\x1D

Plessey

Plessey is a pulse-width modulated symbology like most other bar codes. It includes a start character, data characters, an eight-bit cyclic check digit, and a termination bar. The code is continuous and not self-checking. You need to configure two parameters for Plessey code: Start Code and Check Digit. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

Tap (+) to expand the **Plessey** parameter, select the setting to be changed, then select an option to change this setting or select an option from the drop-down list.

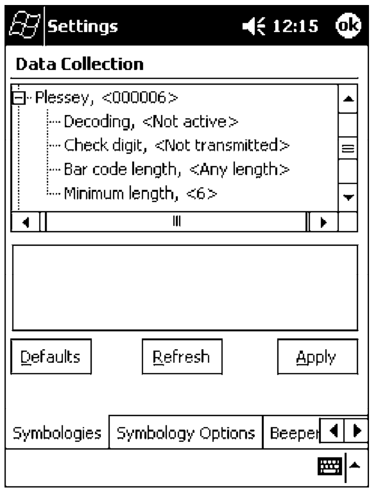
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.10.1

Options

Decoding	0	Not active *(default)*
	1	Active
Check digit	0	Not transmitted *(default)*
	1	Transmitted
Bar code length	0	Any length *(default)*
	1	Minimum length
Minimum length	001-254	Minimum bar code length 1-254 *(6)*

Note: If **Bar code length** = "1" then **Minimum length** is entered.



Plessey, <000006>
Decoding, <Not active>
Check digit, <Not transmitted>
Bar code length, <Any length>
Minimum length, <6>

MSI

MSI is a symbology similar to Plessey code (page 298) that includes a start pattern, data characters, one or two check digits, and a stop pattern. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

Tap (+) to expand the **MSI** parameter, select the setting to be changed, then select an option to change this setting or select an option from the drop-down list.

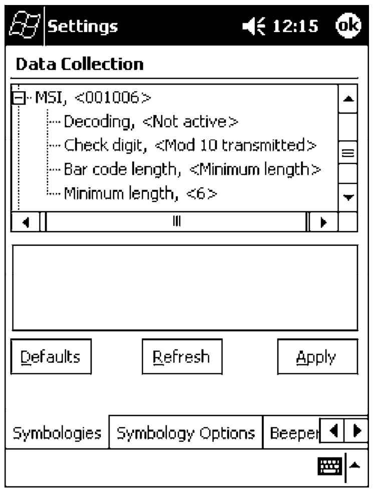
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.15.1

Options

Decoding	0	Not active *(default)*
	1	Active
Check digit	0	Mod 10 transmitted *(default)*
	1	Mod 10 Not transmitted
	2	Double Mod 10 transmitted
	3	Double Mod 10 not transmitted
Bar code length	0	Any length
	1	Minimum length *(default)*
Minimum length	001-254	Minimum length 1-254 *(6)*

Note: If **Bar code length** = “1” then **Minimum length** is entered.



MSI, <001006>
Decoding, <Not active>
Check digit, <Mod 10 transmitted>
Bar code length, <Minimum length>
Minimum length, <6>

PDF 417

PDF 417 is a stacked two-dimensional symbology that provides the ability to scan across rows of code. Each row consists of start/stop characters, row identifiers, and symbol characters, which consist of four bars and four spaces each and contain the actual data. This symbology uses error correction symbol characters appended at the end to recover loss of data.

Because the virtual wedge translates incoming data into keypad input, the size of the keypad buffer limits the effective length of the label to 128 characters. Longer labels may be truncated. For PDF 417 labels of more than 128 characters, you can develop an application that bypasses the keypad buffer.

Action

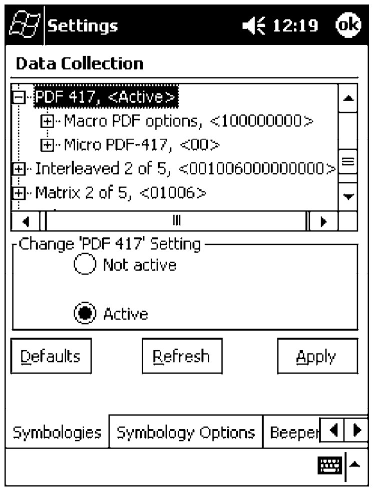
Tap the **PDF 417** parameter, then select an option to change this parameter setting. Tap (+) to access either the **Macro PDF options** parameter or the **Micro PDF 417** parameter.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.17.1

Options

0	Not active
1	Active *(default)*



PDF 417, <Active>
Macro PDF options, <100000000>
Micro PDF-417, <00>
Interleaved 2 of 5, <001006000000000>
Matrix 2 of 5, <01006>

Change 'PDF 417' Setting
Not active
[✓ Active]

Macro PDF options

Macro PDF is used when a long message requires more than one PDF 417 label. *Note that this is not available when you use an imager with your 700 Series Computer.*

- Select **Buffered** to store a multi-label PDF 417 message in the Sabre buffer, thus transmitting the entire message when all labels have been read.

- Select **Unbuffered** for multi-label PDF 417 messages that are too long for the Sabre buffer (memory overflow). Each part of the PDF 417 label is transmitted separately, and the host application must then assemble the message using the macro PDF control header transmitted with each label. *Control Header is only present in macro PDF codes and is always transmitted with unbuffered option.*

Action

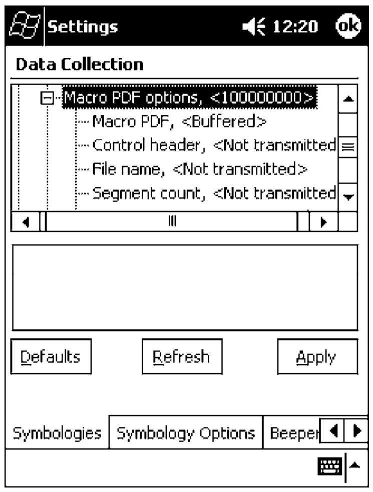
Tap (+) to expand the **PDF 417** parameter, tap (+) to expand the **Macro PDF** parameter, select a setting to be changed, then select an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.22.1

Options

Macro PDF	0	Unbuffered
	1	Buffered *(default)*
Control header	0	Not transmitted *(default)*
	1	Transmitted
File name	0	Not transmitted *(default)*
	1	Transmitted
Segment count	0	Not transmitted *(default)*
	1	Transmitted
Time stamp	0	Not transmitted *(default)*
	1	Transmitted
Sender	0	Not transmitted *(default)*
	1	Transmitted
Addressee	0	Not transmitted *(default)*
	1	Transmitted
File size	0	Not transmitted *(default)*
	1	Transmitted
Checksum	0	Not transmitted *(default)*
	1	Transmitted



Macro PDF options, <100000000>
Macro PDF, <Buffered>
Control header, <Not transmitted
File name, <Not transmitted>
Segment count, <Not transmitted

Micro PDF 417

Micro PDF 417 is a multi-row symbology derived from and closely based on PDF 417 *(page 300)*. A limited set of symbology sizes is available, together with a fixed level of error correction for each symbology size. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

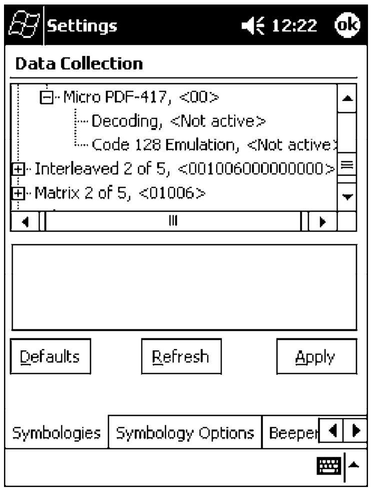
Tap (+) to expand the **PDF 417** parameter, tap (+) to expand the **Micro PDF 417** parameter, select a setting to be changed, then select an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.27.1

Options

Decoding	0	Not active *(default)*
	1	Active
Code 128 Emulation	0	Not active *(default)*
	1	Active



Micro PDF-417, <00>
Decoding, <Not active>
Code 128 Emulation, <Not active
Interleaved 2 of 5, <001006000000000>
Matrix 2 of 5, <01006>

Interleaved 2 of 5

Interleaved 2 of 5 (I 2 of 5) is a high-density, self-checking, continuous, numeric symbology used mainly in inventory distribution and the automobile industry.

Note: An Interleaved 2 of 5 bar code label must be at least three characters long for the 700 Series Computer to scan and decode correctly.

Action

Tap (+) to expand the **Interleaved 2 of 5** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

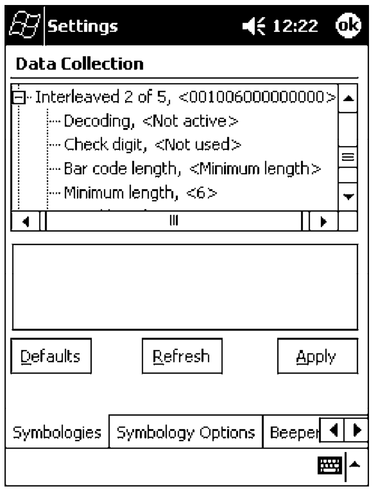
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.23.1

Options

Decoding	0	Not active *(default)*
	1	Active
Check digit	0	Not used *(default)*
	1	Mod 10 transmitted
	2	Mod 10 not transmitted
	3	French CIP transmitted
	4	French CIP not transmitted
Bar code length	0	Any length
	1	Minimum length *(default)*
	2	Fixed lengths
Minimum length	003-254	Minimum length 3-254 *(6)*
Fixed length 1	003-254	Fixed length 3-254 *(0)*
Fixed length 2	003-254	Fixed length 3-254 *(0)*
Fixed length 3	003-254	Fixed length 3-254 *(0)*

Note: If **Bar code length** = “1” then **Minimum length** is entered. If **Bar code length** =“2” then **Fixed length 1**, **Fixed length 2**, or **Fixed length 3** is entered.



Interleaved 2 of 5, <001006000000000>
Decoding, <Not active>
Check digit, <Not used>
Bar code length, <Minimum length>
Minimum length, <6>

Matrix 2 of 5

Matrix 2 of 5 is a numerical symbology. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

Tap (+) to expand the **Matrix 2 of 5** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

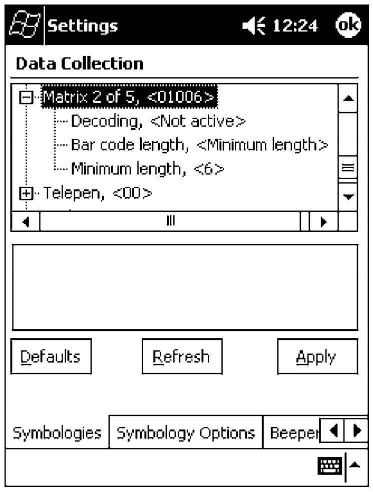
SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.24.1

Options

Decoding	0	Not active *(default)*
	1	Active
Bar code length	0	Any length
	1	Minimum length *(default)*
Minimum length	001-254	Minimum length 1-254 *(6)*

Note: If **Bar code length** = "1" then **Minimum length** is entered.



Matrix 2 of 5, <01006>
Decoding, <Not active>
Bar code length, <Minimum length>
Minimum length, <6>
Telepen, <00>

Telepen

Telepen is an alphanumeric, case-sensitive, full ASCII symbology. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

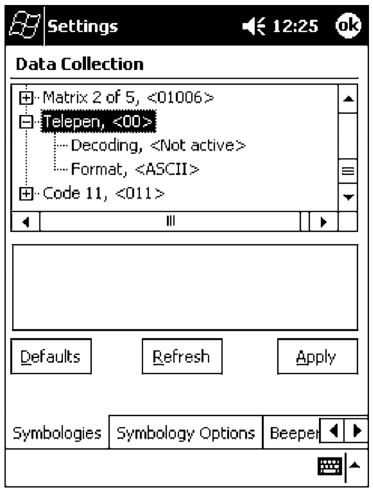
Tap (+) to expand the **Telepen** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.25.1

Options

Decoding	0	Not active *(default)*
	1	Active
Format	0	ASCII *(default)*
	1	Numeric



```
⊞ Matrix 2 of 5, <01006>
⊟ Telepen, <00>
    Decoding, <Not active>
    Format, <ASCII>
⊞ Code 11, <011>
```

Code 11

Code 11 is a high density, discrete numeric symbology that is extensively used in labeling telecommunications components and equipment. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

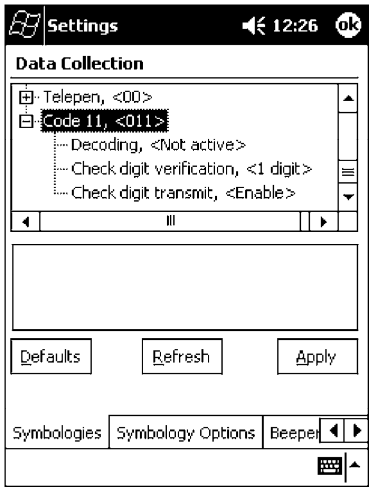
Tap (+) to expand the **Code 11** parameter, select the setting to be changed, then tap an option to change this setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.26.1

Options

Decoding	0	Not active *(default)*
	1	Active
Check digit verification	1	1 digit *(default)*
	2	2 digits
Check digit transmit	0	Disable *(default)*
	1	Enable


Settings 12:26
Data Collection
Telepen, <00>
Code 11, <011>
Decoding, <Not active>
Check digit verification, <1 digit>
Check digit transmit, <Enable>

QR Code

QR Code (Quick Response Code) is a two-dimensional matrix symbology containing dark and light square data modules. It has position detection patterns on three of its four corners and features direct encodation of the Japanese Kana-Kanji character set. It can encode up to 2509 numeric or 1520 alphanumeric characters and offers three levels of error detection. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

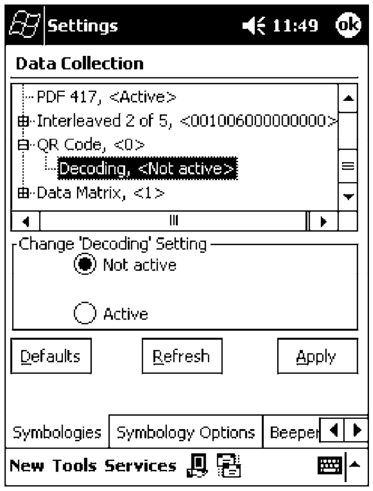
Tap (+) to expand the **QR Code** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.35.1

Options

Decoding	0	Not active
	1	Active *(default)*



Settings 11:49

Data Collection

PDF 417, <Active>
Interleaved 2 of 5, <001006000000000>
QR Code, <0>
Decoding, <Not active>
Data Matrix, <1>

Change 'Decoding' Setting
[✓ Not active]
Active

Data Matrix

A two-dimensional matrix symbology, which is made of square modules arranged within a perimeter finder pattern. The symbology utilizes Error Checking and Correcting (ECC) algorithm with selectable levels for data error recovery and Cyclic Redundancy Check algorithm to validate the data. The character set includes either 128 characters conforming to ISO 646 (ANSI X3.4 - 1986) or 256 extended character set. Maximum capacity of a symbol is 2335 alphanumeric characters, 1556 8-bit byte characters or 3116 numeric digits. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

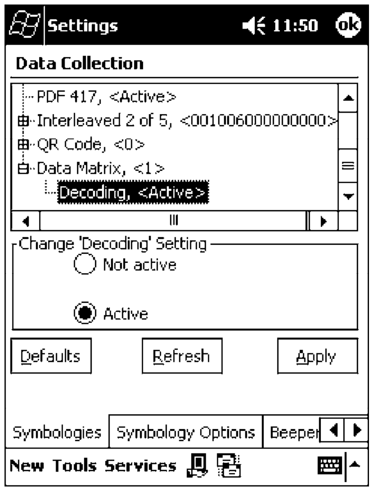
Tap (+) to expand the **Data Matrix** parameter, select the setting to be changed, then tap an option to change this setting or select an option from the drop-down list.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.1.1.34.1

Options

Decoding	0	Not active
	1	Active *(default)*



Settings 11:50
Data Collection
PDF 417, <Active>
Interleaved 2 of 5, <001006000000000>
QR Code, <0>
Data Matrix, <1>
Decoding, <Active>
Change 'Decoding' Setting
Not active
Active [✓]

Symbology Options

Data Collection

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Data Collection** icon to access its control panel applet.

Use the right and left arrows to scroll to the **Symbology Options** tab, then tap this tab to access its parameters. The following are parameters for bar code symbology options. *Note that these are listed in the order of their appearance within the Symbology Options tab.*

Symbology ID

Identifies the bar code symbology in which data has been encoded by prepending a user-specified symbology identifier to the data. You can prepend one of these types of character strings to identify the symbology:

- **User-defined ASCII Character** *(Option 1)*:
  A user-defined symbology identifier is a single ASCII character. You can assign a custom identifier character to each bar code symbology. *Note that this is not available when you use an imager with your 700 Series Computer.*
- **AIM ISO/IEC Standard** *(Option 2 - Required to define symbology IDs)*:
  The AIM Standard has a three-character structure which indicates the symbology and optional features. See the *AIM ISO/IEC Standard* for more information.

Action

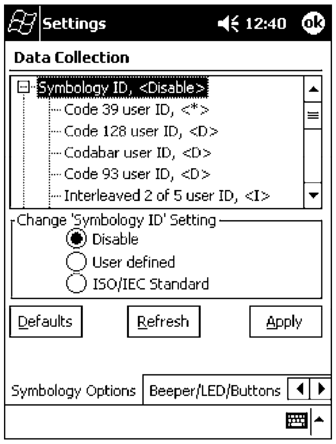
Select **Symbology ID**, then select an option to change this parameter setting. Tap (+) to expand the **Symbology ID** parameter, then select any of the user ID parameters listed. *See the top of the next page for a sample screen of the Code 39 user ID.*

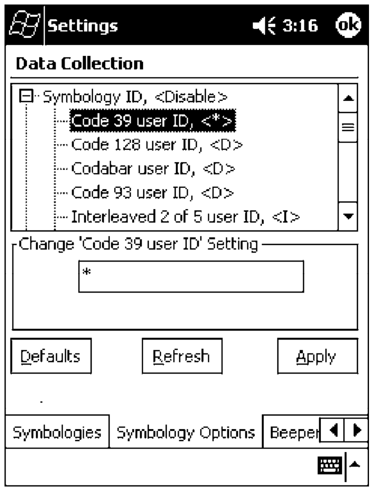
SNMP OID

1.3.6.1.4.1.1963.15.3.3.4.1.22.1

Options

0	Disable *(default)*
1	User defined *(disabled when using an imager)*
2	ISO/IEC Standard


Settings 12:40
Data Collection
Symbology ID, <Disable>
Code 39 user ID, <*>
Code 128 user ID, <D>
Codabar user ID, <D>
Code 93 user ID, <D>
Interleaved 2 of 5 user ID, <I>
Change 'Symbology ID' Setting
● Disable
○ User defined
○ ISO/IEC Standard


Settings 3:16 ok

Data Collection

- Symbology ID, <Disable>
  - Code 39 user ID, <*>
  - Code 128 user ID, <D>
  - Codabar user ID, <D>
  - Code 93 user ID, <D>
  - Interleaved 2 of 5 user ID, <I>

Change 'Code 39 user ID' Setting

*

Code 39 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 39 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 39 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.3.1

Options: *x* where *x* is a single ASCII character. *Default is asterisk (*).*

Code 128 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 128 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 128 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.5.1

Options: *x* where *x* is a single ASCII character. *Default is asterisk (*).*

Codabar User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Codabar bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Codabar user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.2.1

Options: *x* where *x* is a single ASCII character. *Default is D.*

Code 93 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 93 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 93 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.4.1

Options: *x* where *x* is a single ASCII character. *Default is asterisk (*).*

Interleaved 2 of 5 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Interleaved 2 of 5 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Interleaved 2 of 5 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.10.1

Options: *x* where *x* is a single ASCII character. *Default is I (not lowercase L).*

PDF-417 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify PDF 417 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **PDF 417 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.12.1

Options: *x* where *x* is a single ASCII character. *Default is an asterisk (*).*

MSI User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify MSI bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **MSI user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.11.1

Options: *x* where *x* is a single ASCII character. *Default is D.*

Plessey User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Plessey bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Plessey user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.13.1

Options: *x* where *x* is a single ASCII character. *Default is D.*

Standard 2 of 5 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Standard 2 of 5 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Standard 2 of 5 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.23.1

Options: *x* where *x* is a single ASCII character. *Default is D.*

UPC A User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify UPC-A (Universal Product Code) bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **UPC A user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.6.1

Options: *x* where *x* is a single ASCII character. *Default is A.*

UPC E User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify UPC-E bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **UPC E user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.7.1

Options: *x* where *x* is a single ASCII character. *Default is E.*

EAN 8 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify EAN-8 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **EAN 8 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.8.1

Options: *x* where *x* is a single ASCII character. *Default is \xFF.*

EAN 13 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify EAN-13 (European Article Numbering) bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **EAN 13 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.9.1

Options: *x* where *x* is a single ASCII character. *Default is F.*

Matrix 2 of 5 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Matrix 2 of 5 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Matrix 2 of 5 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.24.1

Options: *x* where *x* is a single ASCII character. *Default is D.*

Telepen User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Telepen bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Telepen user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.25.1

Options: *x* where *x* is a single ASCII character. *Default is an asterisk (*).*

Code 11 User ID

If “1” was selected in the **Symbology ID** parameter, you can set your own ASCII character to identify Code 11 bar code data. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action: Tap (+) to expand the **Symbology ID** parameter, select the **Code 11 user ID** parameter, then enter a user ID value to change this parameter setting.

SNMP OID: 1.3.6.1.4.1.1963.15.3.3.4.1.16.1

Options: *x* where *x* is a single ASCII character. *Default is asterisk (*).*

Prefix

Prepends a string of up to 20 ASCII characters to all scanned data.

Action

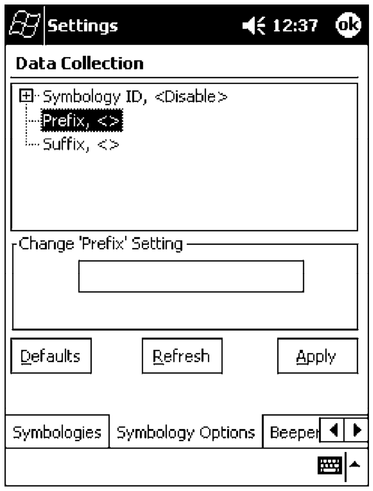
Tap the **Prefix** parameter, then enter a prefix value to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.4.1.29.1

Options

Acceptable values are up to 20 ASCII characters.
Embedded null (<NUL >) characters are not allowed.
Default is no characters (disabled).


Settings 12:37
Data Collection
Symbology ID, <Disable>
Prefix, <>
Suffix, <>
Change 'Prefix' Setting

Suffix

Appends a string of up to 20 ASCII characters to all scanned data.

Action

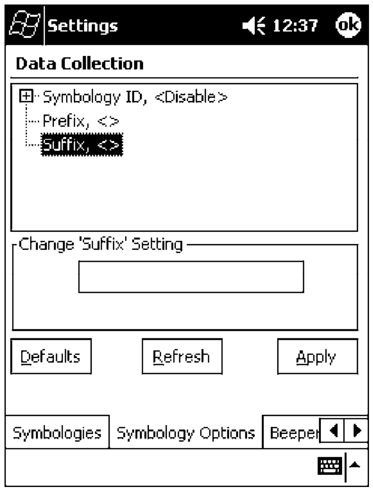
Tap the **Suffix** parameter, then enter a suffix value to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.4.1.30.1

Options

Acceptable values are up to 20 ASCII characters. Embedded null (<NUL >) characters are not allowed. *Default is no characters (disabled).*


Data Collection
Symbology ID, <Disable>
Prefix, <>
Suffix, <>
Change 'Suffix' Setting

Beeper/LED

Data Collection

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Data Collection** icon to access its control panel applet.

Use the right and left arrows to scroll to the **Beeper/LED** tab, then tap this tab to access its parameters.

Most of these functions are not available when using an imager. The following table shows which functions are supported either by an imager or by a laser scanner.

Beeper Function	Imager	Laser Scanner
Beeper Volume	X	X
Beeper Frequency		X
Good Read Beeps		X
Good Read Beep Duration		X

The following are parameters for features on the 700 Series Computer. *Note that these are listed in the order of their appearance.*

Beeper Volume

Sets the volume for the good read beep.

Action

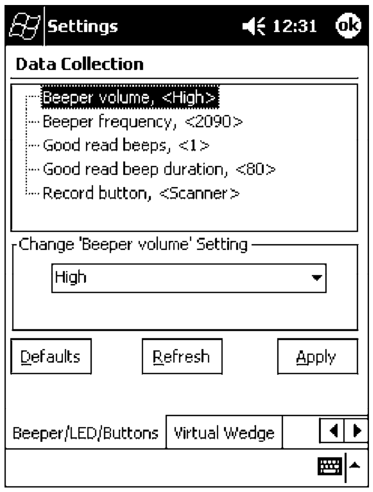
Tap the **Beeper volume** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.6.1

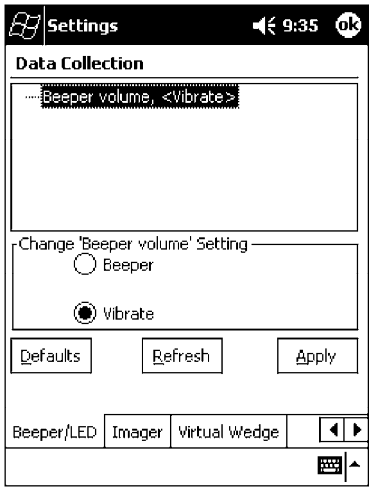
Laser Scanner Options

0	Low
1	High *(default)*
2	Medium
3	Off
4	Vibrate


Settings 12:31
Data Collection
Beeper volume, <High>
Beeper frequency, <2090>
Good read beeps, <1>
Good read beep duration, <80>
Record button, <Scanner>
Change 'Beeper volume' Setting
High

Imager Options

1	Beeper *(default)*
4	Vibrate


Settings 9:35 ok

Data Collection

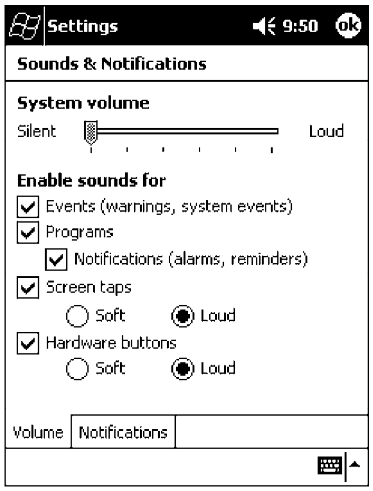
Beeper volume, <Vibrate>

Change 'Beeper volume' Setting
○ Beeper
● Vibrate

Silencing the Beeper Volume

Sounds & Notifications

To turn the beeper off, tap **Start** → **Settings** → the **Personal** tab → **Sounds and Notifications** → the **Volume** tab, drag the **System volume** slider bar to the left "Silent" position, then tap **ok** to exit this applet.


Settings 9:50 ok

Sounds & Notifications

System volume
Silent — Loud

Enable sounds for
☑ Events (warnings, system events)
☑ Programs
☑ Notifications (alarms, reminders)
☑ Screen taps
○ Soft ● Loud
☑ Hardware buttons
○ Soft ● Loud

Beeper Frequency

Sets the frequency for the good read beep. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

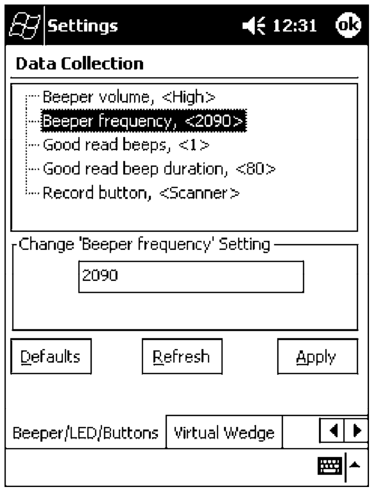
Tap the **Beeper frequency** parameter, then enter a frequency value to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.7.1

Options

1000-4095 *(default is 2090)*



Settings 12:31

Data Collection

- Beeper volume, <High>
- Beeper frequency, <2090>
- Good read beeps, <1>
- Good read beep duration, <80>
- Record button, <Scanner>

Change 'Beeper frequency' Setting

2090

Good Read Beeps

Sets the number of good read beeps. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

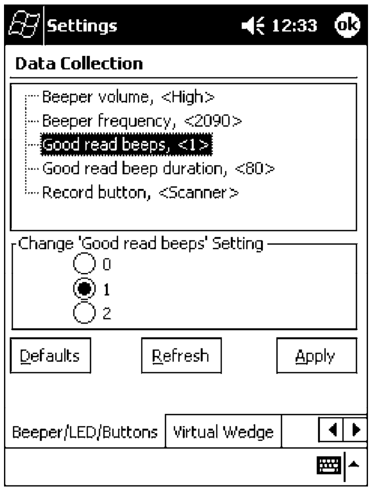
Tap the **Good read beeps** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.8.1

Options

0	No beeps
1	One beep *(default)*
2	Two beeps


Settings 12:33 ok

Data Collection

Beeper volume, <High>
Beeper frequency, <2090>
Good read beeps, <1>
Good read beep duration, <80>
Record button, <Scanner>

Change 'Good read beeps' Setting
○ 0
● 1
○ 2

Good Read Beep Duration

Sets the duration of the good read beep. *Note that this is not available when you use an imager with your 700 Series Computer.*

Action

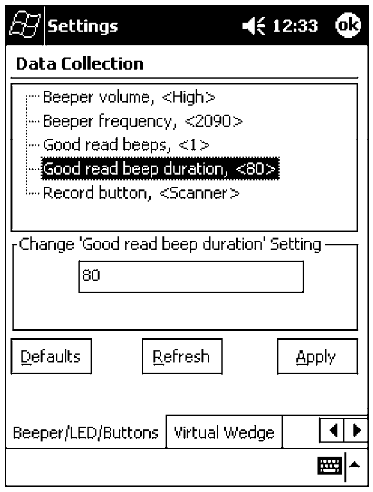
Tap the **Good read beep duration** parameter, then enter a duration value to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.1.4.1.9.1

Options

0`2550	Beep duration in milliseconds. *(default is 80)*



Settings 12:33

Data Collection

Beeper volume, <High>
Beeper frequency, <2090>
Good read beeps, <1>
Good read beep duration, <80>
Record button, <Scanner>

Change 'Good read beep duration' Setting

80

Imager

Data Collection

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Data Collection** icon to access its control panel applet.

Use the right and left arrows to scroll to the **Imager** tab, then tap this tab to access its parameters.

The following are parameters for the imager. Note that these are listed in the order of their appearance within the Imager tab.

Aimer LED duration

The Aimer LED duration controls the time the Aimer LED is turned on when the scan button is pressed. After this time, images are captured for decoding. The purpose is to position the Aimer LED on the bar code symbol before attempting to decode the bar code. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

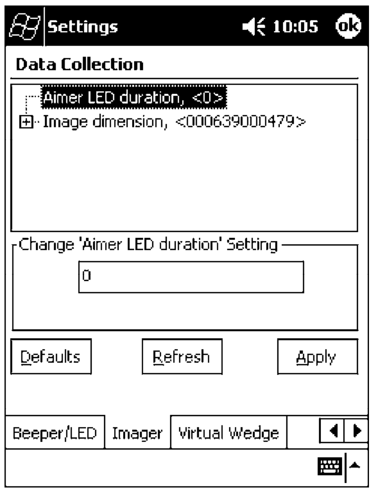
Tap the **Aimer LED duration** parameter, then enter a value to change this setting. Note that values must be in 50 ms increments, such as 500, 650, or 32500. Values not entered in 50 ms increments will be rounded down. For example, 2489 ms would be rounded down to 2450 ms, 149 ms would be rounded down to 100 ms, etc..

SNMP OID

1.3.6.1.4.1.1963.15.3.3.3.1.1.21.1

Options

0-65500 ms *(Default is 0)*



Data Collection

Aimer LED duration, <0>

Image dimension, <000639000479>

Change 'Aimer LED duration' Setting

0

Image Dimension

The image dimensions control the horizontal size of the image for decoding. This can restrict the image to one bar code when otherwise, there might be more than one bar code in the image to be decoded. *Note that this is not available when you use a laser scanner with your 700 Series Computer.*

Action

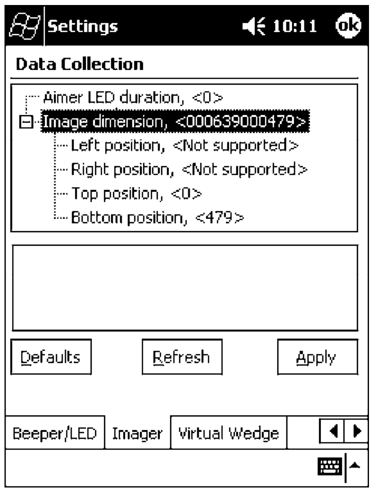
Tap the **Image dimension** parameter, select the position to be changed, then tap an option or enter a value to change this position.

SNMP OID

1.3.6.1.4.1.1963.15.3.3.3.1.1.22.1

Options

Left position	0	Not supported
Right position	0	Not supported
Top position	0-478	Position in pixels *(0)*
Bottom position	0-479	Position in pixels *(479)*



Settings 10:11 ok

Data Collection

Aimer LED duration, <0>
Image dimension, <000639000479>
Left position, <Not supported>
Right position, <Not supported>
Top position, <0>
Bottom position, <479>

Virtual Wedge

Data Collection

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Data Collection** icon to access its control panel applet.

Use the right and left arrows to scroll to the **Virtual Wedge** tab, then tap this tab to access its parameters.

The following are parameters for the virtual wedge scanner. *Note that these are listed in the order of their appearance within the Virtual Wedge tab.*

Virtual Wedge

Enables or disables the virtual wedge for the internal scanner. The virtual wedge retrieves scanned Automatic Data Collection (ADC) data and sends it to the keypad driver so that the 700 Series Computer can receive and interpret the data as keypad input.

Because the virtual wedge translates incoming data into keypad input, the size of the keypad buffer limits the effective length of the label to 128 characters. Longer labels may be truncated. For labels of more than 128 characters, you need to develop an application that bypasses the keypad buffer.

Action

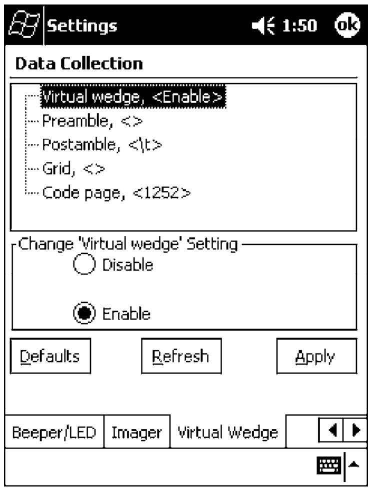
Tap the **Virtual Wedge** parameter, then tap an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.2.1.1.2.1

Options

0	Disable
1	Enable *(default)*


Settings 1:50 ok
Data Collection
Virtual wedge, <Enable>
Preamble, <>
Postamble, <\t>
Grid, <>
Code page, <1252>

Change 'Virtual wedge' Setting
Disable [✓ Enable]

Defaults Refresh Apply

Beeper/LED | Imager | Virtual Wedge

Preamble

Sets the preamble that precedes any data you scan with the 700 Series Computer. Common preambles include a data location number or an operator number.

Action

Tap the **Preamble** parameter, then enter a preamble value to change this parameter setting.

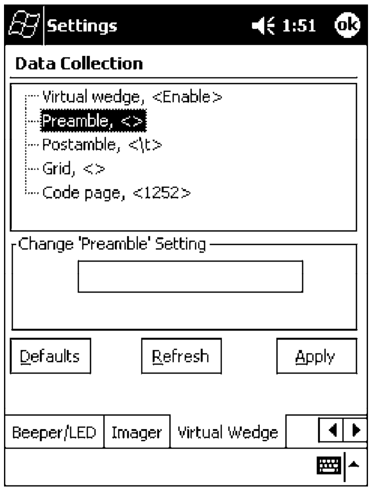
SNMP OID

1.3.6.1.4.1.1963.15.3.2.1.1.3.1

Syntax

`ADdata`

where *data* is acceptable values up to 31 ASCII characters. Embedded null (<NUL >) characters are not allowed. *Default is no characters (disabled).*



```
Virtual wedge, <Enable>
Preamble, <>
Postamble, <\t>
Grid, <>
Code page, <1252>
```

Note: When you enter the AD command without data, the preamble is disabled. If you want to use quotation marks or the following combinations of characters as part of the appended data, separate those characters from the AD command with quotes. If you do not use quotes as described here, the 700 Series Computer will interpret the characters as another configuration command:

AD
AE
AF
KC
BV
EX
DF

EXAMPLE:

To use the two-character string BV as a preamble, scan this command (as a Code 39 label) or send this command through the network: `$+AD"BV"`

Postamble

Sets the postamble that is appended to any data you scan with the 700 Series Computer. Common postambles include cursor controls, such as tabs or carriage return line feeds.

Action

Tap the **Postamble** parameter, then enter a postamble value to change this parameter setting.

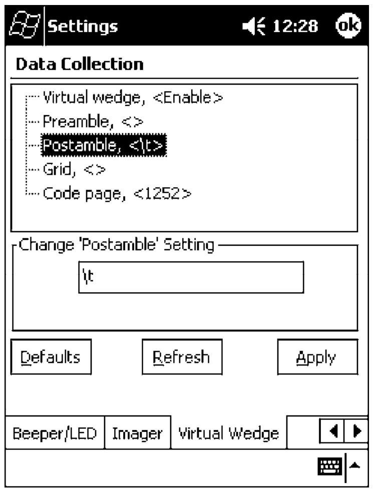
SNMP OID

1.3.6.1.4.1.1963.15.3.2.1.1.4.1

Syntax

`AEdata`

where *data* is any acceptable values up to 31 ASCII characters. Embedded null (<NUL >) characters are not allowed. *Default is the tab character (\t).*


Settings 12:28
Data Collection
Virtual wedge, <Enable>
Preamble, <>
Postamble, <\t>
Grid, <>
Code page, <1252>
Change 'Postamble' Setting
\t

Note: When you enter the AE command without data, the postamble is disabled. If you want to use quotation marks or the following combinations of characters as part of the appended data, separate those characters from the AE command with quotes. If you do not use quotes as described here, the 700 Series Computer will interpret the characters as another configuration command.
AD
AE
AF
KC
BV
EX
DF

EXAMPLE:

To use the two-character string BV as a postamble, scan this command (as a Code 39 label) or send this command through the network: `$+AE"BV"`

Grid

Sets the virtual wedge grid, which filters the data coming from this 700 Series Computer. The data server supports data filtering, which allows you to selectively send scanned data. The virtual wedge grid is similar to the "format" argument of the C Runtime Library scan function.

Action

Tap the **Grid** parameter, then enter a grid value to change this parameter setting.

SNMP OID

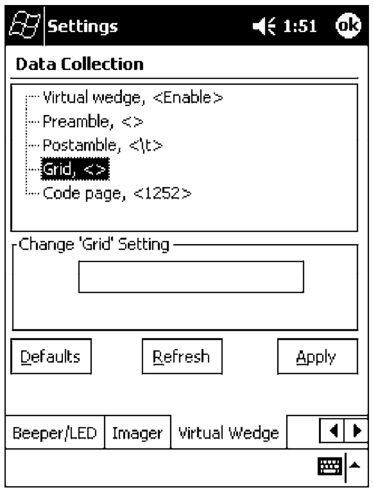
1.3.6.1.4.1.1963.15.3.2.1.1.5.1

Syntax

```
AF<symID> filter-expression= > editing-expression
```

where:

- *<symID>*
  The AIM symbology ID.
- *filter-expression*
  Any character string that includes valid filter expression values, and editing-expression is any character string that includes valid editing expression values.
- *<width>*
  Any positive integer or NULL. A NULL width means that the field type (defined next) applies all the way to the end of the data string. A non-NULL width means that the field applies to that many characters of data. The grid can be up to 240 characters in length. *Default is NULL.*



Virtual wedge, <Enable>
Preamble, <>
Postamble, <\t>
Grid, <>
Code page, <1252>

Code Page

Sets the virtual wedge code page. The code page controls the translation from the character set of the raw collected data to Unicode, which is the character set expected by Windows CE applications. The default code page is 1252, which is the Windows Latin 1 (ANSI) character set.

Action

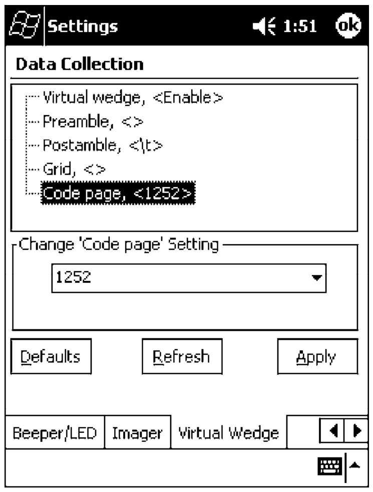
Tap the **Code Page** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.15.3.2.1.1.6.1

Options

The only acceptable value for the code page parameter is "*1252*," which is the default.



Virtual wedge, <Enable>
Preamble, <>
Postamble, <\t>
Grid, <>
Code page, <1252>

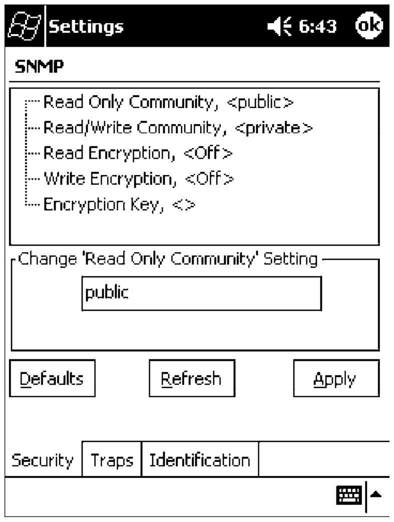
Change 'Code page' Setting
1252

SNMP Control Panel Applet

Simple Network Management Protocol (SNMP) parameters include identification information, security encryption, security community strings, and traps.

SNMP

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **SNMP** icon to access its control panel applet.


Settings 6:43 ok

SNMP

- Read Only Community, <public>
- Read/Write Community, <private>
- Read Encryption, <Off>
- Write Encryption, <Off>
- Encryption Key, <>

Change 'Read Only Community' Setting

public

Tap a tab to access its menus. These tabs represent three groups of settings or parameters:

- **Security** *(starting on the next page)*
- **Traps** *(starting on page 336)*
- **Identification** *(starting on page 338)*

Security

SNMP

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **SNMP** icon → the **Security** tab to access its parameters.

The following are parameters that affect encryption and community strings. *Note that these are listed in the order of their appearance within the Security tab.*

Read Only Community

Sets the read-only community string for this 700 Series Computer, which is required for processing of SNMP get and get next requests.

Action

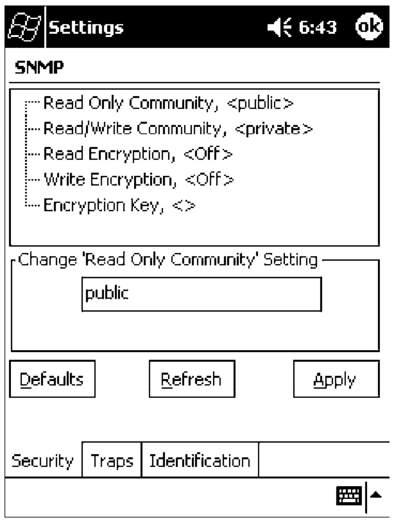
Tap the **Read Only Community** parameter, then enter a community string to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.2.0

Options

The read-only community string can be up to 128 ASCII characters. *Default is Public.*



Settings 6:43 ok

SNMP

Read Only Community, <public>
Read/Write Community, <private>
Read Encryption, <Off>
Write Encryption, <Off>
Encryption Key, <>

Change 'Read Only Community' Setting

public

Read/Write Community

Sets the read/write community string, which is required for processing of SNMP set requests by this 700 Series Computer. An SNMP packet with this name as the community string will also process SNMP get and next requests.

Action

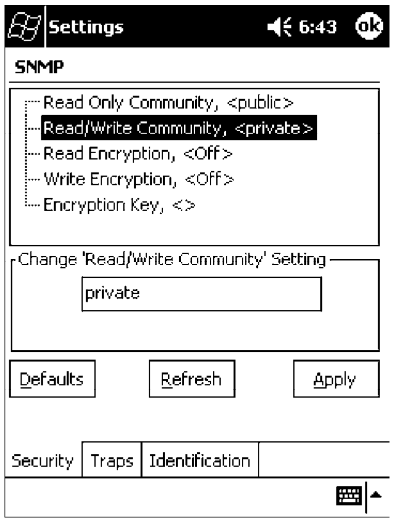
Tap the **Read/Write Community** parameter, then enter a community string to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.3.0

Options

The read/write community string can be up to 128 ASCII characters. *Default is Private.*



Settings 6:43 ok

SNMP

- Read Only Community, <public>
- Read/Write Community, <private>
- Read Encryption, <Off>
- Write Encryption, <Off>
- Encryption Key, <>

Change 'Read/Write Community' Setting

private

Read Encryption

Sets the packet-level mode of security for SNMP read-only requests. If you enable read encryption, all received SNMP get and get next packets have to be encrypted or the packet will not be authorized. If encryption is enabled, you can only use software provided by Intermec Technologies.

Note: To enable security encryption, you also need to set the Security Encryption Key (page 335).

Action

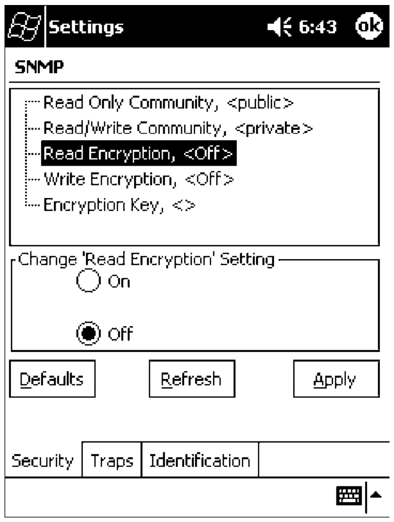
Tap the **Read Encryption** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.4.0

Options

1	On	SNMP get and get next packets must be encrypted
2	Off	SNMP packets do not have to be encrypted *(default)*


Settings 6:43
SNMP
Read Only Community, <public>
Read/Write Community, <private>
Read Encryption, <Off>
Write Encryption, <Off>
Encryption Key, <>
Change 'Read Encryption' Setting
On [✓ Off]

Write Encryption

Sets the packet-level mode of security for SNMP read/write requests. If you enable write encryption, all SNMP packets that are received with the read/write community string have to be encrypted or the packet will not be authorized. You need to use software from Intermec Technologies that supports encryption.

Note: To enable security encryption, you also need to set the Security Encryption Key (page 335).

Action

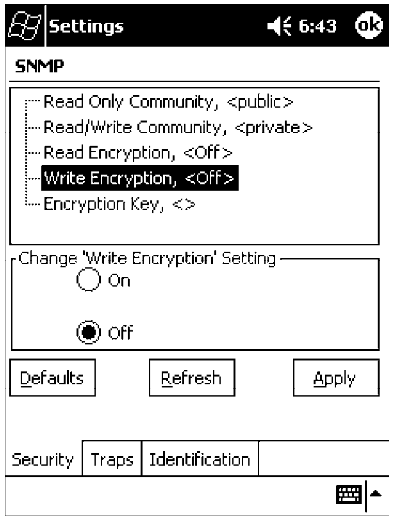
Tap the **Write Encryption** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.5.0

Options

1	On	SNMP packets must be encrypted
2	Off	SNMP packets do not have to be encrypted *(default)*


Settings 6:43 ok
SNMP
Read Only Community, <public>
Read/Write Community, <private>
Read Encryption, <Off>
Write Encryption, <Off>
Encryption Key, <>
Change 'Write Encryption' Setting
On [✓ Off]

Encryption Key

Identifies the key that this 700 Series Computer uses to encrypt or decipher SNMP packets. Encryption is used only by software provided by Intermec Technologies. If encryption is enabled, SNMP management platforms will not be able to communicate with the 700 Series Computer. The encryption key is returned encrypted.

Action

Tap the **Encryption Key** parameter, then enter a security encryption key value to change this parameter setting.

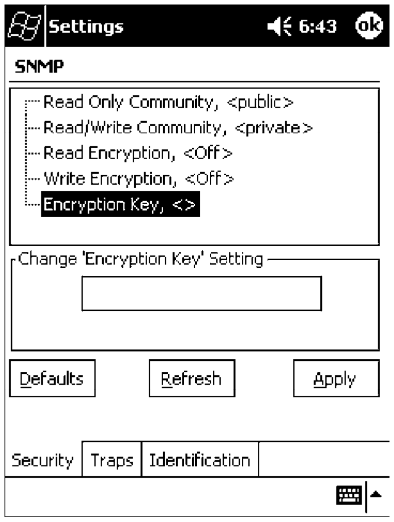
Note: You also need to set either **Read Encryption** (page 333) or **Write Encryption** (page 334) or both.

SNMP OID

1.3.6.1.4.1.1963.10.5.1.6.0

Options

The encryption key can be from 4 to 20 ASCII characters. *Default is NULL.*



Read Only Community, <public>
Read/Write Community, <private>
Read Encryption, <Off>
Write Encryption, <Off>
Encryption Key, <>
Change 'Encryption Key' Setting

Traps

SNMP

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **SNMP** icon → the **Traps** tab to access its parameters.

The following are authentication and threshold parameters for traps. *Note that these are listed in the order of their appearance within the Traps tab.*

Authentication

Determines whether to send authentication traps. When trap authentication is enabled, an authentication trap is sent if an SNMP packet is received by the master agent with an invalid community string.

Action

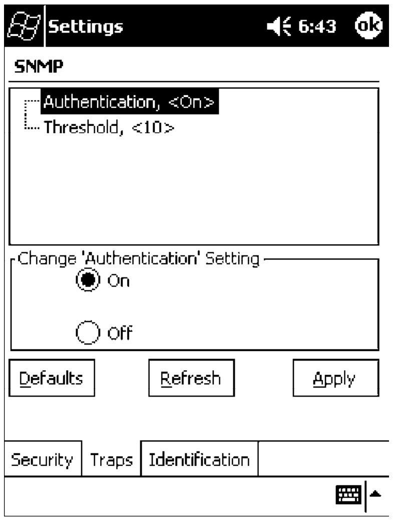
Tap the **Authentication** parameter, then select an option to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.2.2.0

Options

1	On *(default)*
2	Off


Settings 6:43 ok
SNMP
Authentication, <On>
Threshold, <10>
Change 'Authentication' Setting
[✓ On] Off

Threshold

Determines the maximum number of traps per second that the master agent generates. If the threshold is reached, the trap will not be sent.

Action

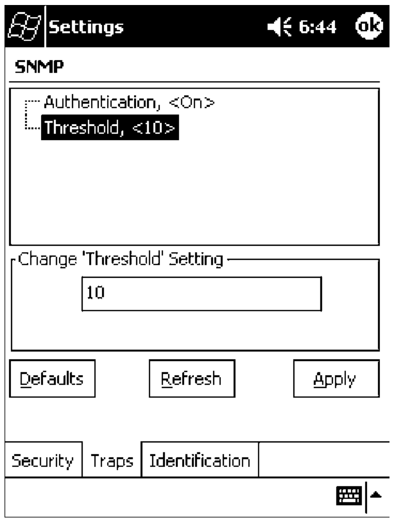
Tap the **Threshold** parameter, then enter a threshold value to change this parameter setting.

SNMP OID

1.3.6.1.4.1.1963.10.5.2.3.0

Options

Any positive integer value. *Default is 10.*



Authentication, <On>
Threshold, <10>
Change 'Threshold' Setting
10

Identification

SNMP

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **SNMP** icon → the **Identification** tab to access its parameters.

The following are parameters for contact, location, and name information for support purposes. *Note that these are listed in the order of their appearance within the Identification tab.*

Contact

Sets the contact information for the person responsible for this 700 Series Computer.

Action

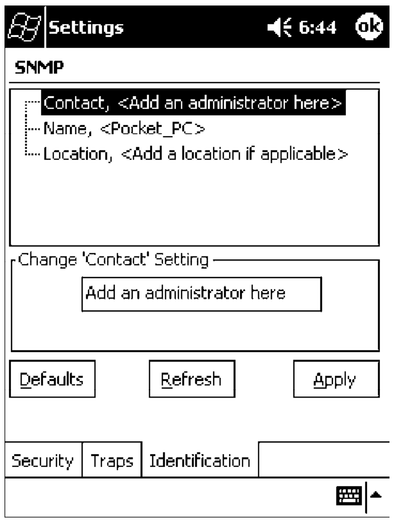
Tap the **Contact** parameter, then enter the name of your contact representative to change this parameter setting.

SNMP OID

1.3.6.1.2.1.1.4.0

Options

The identification contact may be up to 255 ASCII characters. *Default is no characters or blank.*



Contact, <Add an administrator here>
Name, <Pocket_PC>
Location, <Add a location if applicable>
Change 'Contact' Setting
Add an administrator here

Name

Sets the assigned name for this 700 Series Computer.

Action

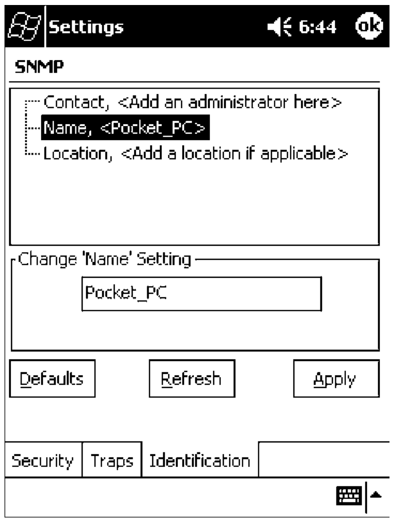
Tap the **Name** parameter, then enter the name of your 700 Series Computer to change this parameter setting.

SNMP OID

1.3.6.1.2.1.1.5.0

Options

The identification name may be up to 255 ASCII characters. *Default is no characters or blank.*


Settings 6:44
SNMP
Contact, <Add an administrator here>
Name, <Pocket_PC>
Location, <Add a location if applicable>
Change 'Name' Setting
Pocket_PC

Location

Sets the identification location for this 700 Series Computer, such as “Shipping.”

Action

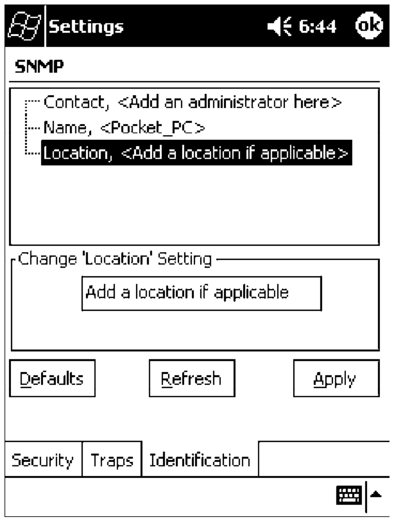
Tap the **Location** parameter, then enter the location of where your 700 Series Computer to change this parameter setting.

SNMP OID

1.3.6.1.2.1.1.6.0

Options

The identification location may be up to 255 ASCII characters. *Default is no characters or blank.*


Settings 6:44 ok

SNMP

Contact, <Add an administrator here>
Name, <Pocket_PC>
Location, <Add a location if applicable>

Change 'Location' Setting

Add a location if applicable

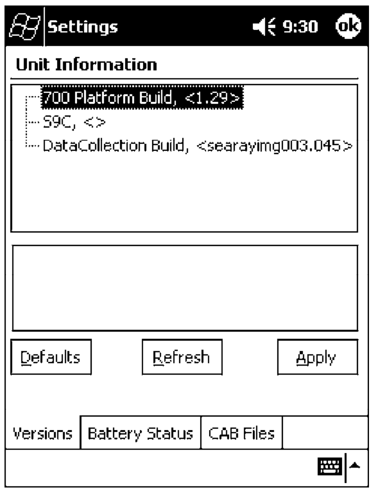
Unit Information Control Panel Applet

Unit Information is a read-only control panel applet that provides information about your 700 Series Computer, such as software version builds, available CAB files, and the internal battery status.

This control panel applet is only available in the 700 Series Computer if Intermec Content is enabled, the Plus region is enabled and installed, and a laser scanner is installed.

Unit Information

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Unit Information** icon to access its control panel applet.



```
700 Platform Build, <1.29>
S9C, <>
DataCollection Build, <searayimg003.045>
```

Tap a tab to access its menus. These tabs represent three groups of settings or parameters:

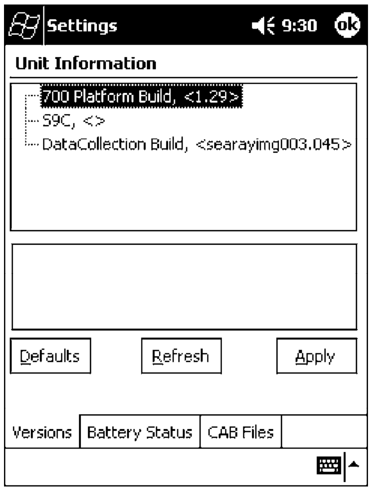
- **Versions** *(starting on the next page)*
- **Battery Status** *(starting on page 343)*
- **CAB Files** *(starting on page 344)*

Versions

You can view the latest software build version on your 700 Series Computer by accessing the **Unit Information** control panel applet.

Unit Information

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Unit Information** icon → the **Versions** tab to view the latest software build version. Tap **ok** to exit this information.



700 Platform Build, <1.29>
S9C, <>
DataCollection Build, <searayimg003.045>

Below are some of the software applications you may find on this screen:

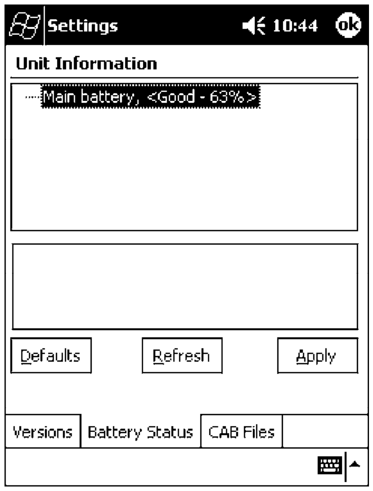
- **700 Platform Build**:
  Shows the latest development or released version of the software build for the 700 Series Computer.
- S9C:
  Provides the name and version of the scanner file built into this 700 Series Computer, along with the current CPU version.
- **DataCollection Build**:
  Shows the latest development or released version of the software build for the Data Collection control panel applet.

Battery Status

You can view the battery status for your 700 Series Computer by accessing the **Unit Information** control panel applet. Unit Manager applications are available on the *700 Series Color Unit Manager CD-ROM*. For more information, consult your Intermec sales representative.

Unit Information

To access the settings from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Unit Information** icon → the **Battery Status** tab to view the current status. Tap **ok** to exit this information.



Settings 10:44 ok

Unit Information

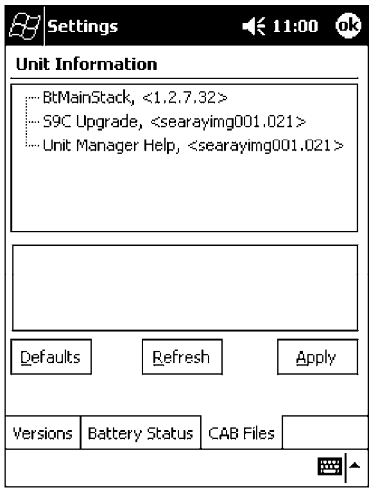
Main battery, <Good - 63%>

CAB Files

You can view the latest developer or released version of each CAB file from Intermec Technologies Corporation that are installed in your 700 Series Computer via the **Unit Information** control panel applet. *Custom CAB files are not displayed in this applet.* See the *Software Tools User's Manual* for more information about these files.

Unit Information

To access the information from the 700 Series Computer, tap **Start** → **Settings** → the **System** tab → the **Unit Information** icon → the **CAB Files** tab to view the current CAB file versions. Tap **ok** to exit this information.



```
BtMainStack, <1.2.7.32>
S9C Upgrade, <searayimg001.021>
Unit Manager Help, <searayimg001.021>
```

When a CAB file is built, a registry entry is created with a build number for that file. This CAB Files control panel applet looks for a registry key for each CAB file installed. When the registry entry is found, the CAB file name and version number information are displayed. If a CAB file has not been installed, then its information is not displayed.

Below is a list of CAB files from Intermec Technologies that are available for your 700 Series Computer with their latest developer or released version of the software build. Should you need to add any of these to your 700 Series Computer, contact an Intermec representative.

- **BtMainStack**:
  Installation of the Main Bluetooth Stack is handled automatically as part of the operating system boot-up procedure. *See Chapter 4, "Network Support," for more information about Bluetooth wireless printing.*
- **Comm Port Wedge**:
  The software build for the Comm Port Wedge. *Note that the Comm Port Wedge CAB file is available on the 700C Tools CD.*
- **NPCPTest**:
  This installs a Norand® Portable Communications Protocol (NPCP) Printing test application which will print to an Intermec® 4815, 4820, or 6820 Printer. *See Chapter 5, "Printer Support," for more information.*

Wireless Printing Demo

- PDWPM0C:
  This is the installer for the Wireless Printing Demo application. To run this demonstration, tap **Start** → **Programs** → the **Wireless Printing Demo** icon. *Press Help in the demo application for more information.*
- **S9C Upgrade**:
  Installs the files needed to upgrade the S9C scanner firmware. *See the Recovery CD Help for more information about upgrading the firmware.*
- **SDK**:
  Installs the Intermec Software Developer' s Kit (SDK). *See the SDK User' s Manual for more information.*
- **Unit Manager**:
  Installs the Unit Manager application which provides tools for remotely managing the 700 Series Computer. Unit Manager applications are available on the *700 Series Color Unit Manager CD-ROM*. For more information, consult your Intermec sales representative.
- **Unit Manager Help**:
  Installs the online help for the Unit Manager application.
- **WinCfg**:
  Configures the NRINET.INI file, launches the NRINet client, and loads and unloads the LAN and WLAN device drivers. *See the Windows 95 and Windows CE Configuration Utilities Reference Manual (P/N: 978-054-010) for more information.*
- **Wireless Printing Sample**:
  Installs a sample application that developers can use for reference when they are developing their own Wireless Printing applications. The source code for this application is included as part of the Wireless Printing SDK on the 700C Tools CD. *See the SDK User' s Manual for more information.*
- **ActiveX Control Tools**:
  This lists some of the CAB files that may be available with which to install ActiveX Control Tools. *See the SDK Online Help for more information.*
  - **AXCommunication**:
    Communication controls that transmit or receive messages from input connections.
  - **AXFileTransfer**:
    File transfer controls that transmit and receive files using the Trivial File Transfer Protocol (TFTP).
  - **AXReaderCommand**:
    Reader command functions that modify and retrieve configuration information from your 700 Series Computer.
  - **AXVWedge**:
    The virtual wedge control that retrieves scanned ADC data and sends it to the keyboard driver to interpret data as keyboard input.

B Unit Manager

Configuration parameters are also configurable using a Unit Manager application which accesses the 700 Series Computer through a web browser on your desktop PC via the SRDEVMGMT.CAB file.

Unit Manager applications are available on the *700 Series Color Unit Manager CD-ROM*. For more information, consult your Intermec sales representative.

Note: Parameter information, such as SNMP OID and options, is detailed in Appendix A, "*Control Panel Applets*."

Data Collection

Data Collection

Within the Unit Manager, click **Configuration** from the left navigation bar, then click the **Data Collection** icon to access any of these tabs: Symbologies, Symbology ID, Beeper/LED, or Virtual Wedge.

Symbologies

Within the Unit Manager, select **Configuration Management → Data Collection**, then click the **Symbologies** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Codabar *(page 292)*
- Code 11 *(page 306)*
- Code 128 *(page 295)*
  - Code 128 Options *(page 296)*
  - Code 128 FNC1 Character *(page 297)*
- Code 39 *(page 290)*
- Code 93 *(page 294)*
  - Code 93 Length *(page 294)*
- Data Matrix *(page 308)*
- Interleaved 2 of 5 *(page 303)*
- Matrix 2 of 5 *(page 304)*
- MSI *(page 299)*
- PDF 417 *(page 300)*
  - Macro PDF *(page 300)*
  - Micro PDF 417 *(page 302)*
- Plessey *(page 298)*
- QR Code *(page 307)*
- Standard 2 of 5 *(page 291)*
- Telepen *(page 305)*
- UPC/EAN *(page 293)*

Symbology ID

Within the Unit Manager, select **Configuration Management** → **Data Collection**, then click the **Symbology ID** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Prefix *(page 315)*
- Suffix *(page 316)*
- Symbology ID *(page 309)*
  - Codabar user ID *(page 310)*
  - Code 11 user ID *(page 314)*
  - Code 128 user ID *(page 310)*
  - Code 39 user ID *(page 310)*
  - Code 93 user ID *(page 311)*
  - EAN-13 user ID *(page 313)*
  - EAN-8 user ID *(page 313)*
  - Interleaved 2 of 5 user ID *(page 311)*
  - Matrix 2 of 5 user ID *(page 313)*
  - MSI user ID *(page 311)*
  - PDF 417 user ID *(page 311)*
  - Plessey user ID *(page 312)*
  - Standard 2 of 5 user ID *(page 312)*
  - Telepen user ID *(page 313)*
  - UPC-A user ID *(page 312)*
  - UPC-E user ID *(page 312)*

Beeper/LED

Within the Unit Manager, select **Configuration Management** → **Data Collection**, then click the **Beeper/LED** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Beeper Frequency *(page 320)*
- Beeper Volume *(page 318)*
- Good Read Beep Duration *(page 322)*
- Good Read Beeps *(page 321)*

Imager

Within the Unit Manager, select **Configuration Management** → **Data Collection**, then click the **Imager** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Aimer LED Duration *(page 323)*
- Image Dimension *(page 324)*

Virtual Wedge

Within the Unit Manager, select **Configuration Management** → **Data Collection**, then click the **Virtual Wedge** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Code Page *(page 329)*
- Grid *(page 328)*
- Postamble *(page 327)*
- Preamble *(page 326)*
- Virtual Wedge *(page 325)*

SNMP

Within the Unit Manager, click **Configuration** from the left navigation bar, then click the **SNMP** icon to access any of these tabs: Security, Traps, or Identification.

Security

Within the Unit Manager, select **Configuration Management** → **SNMP**, then click the **Security** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Encryption Key *(page 335)*
- Read Encryption *(page 333)*
- Read Only Community *(page 331)*
- Read/Write Community *(page 332)*
- Write Encryption *(page 334)*

Traps

Within the Unit Manager, select **Configuration Management** → SNMP, then click the **Traps** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Authentication *(page 336)*
- Threshold *(page 337)*

Identification

Within the Unit Manager, select **Configuration Management** → SNMP, then click the **Identification** tab to access the following parameters. Options for these parameters are listed on the page provided. *These are listed in alphabetical order.*

- Contact *(page 338)*
- Location *(page 340)*
- Name *(page 339)*

Unit

Unit

Within the Unit Manager, click **Configuration** from the left navigation bar, then click the **Unit** icon to access any of these tabs: Date/Time, Display, Keypad, Power Management, or Speaker.

Date/Time

Sets the current date and time.

Action

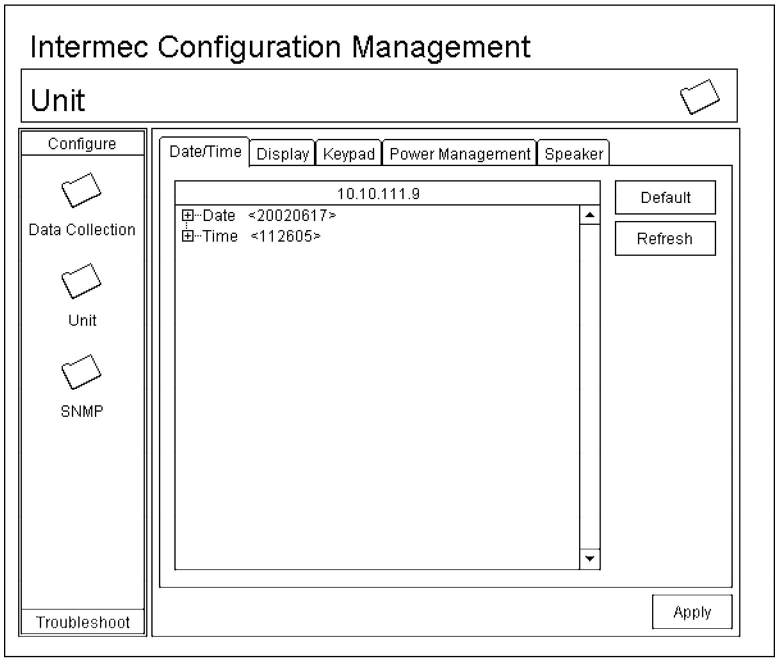
Click the **Date/Time** tab, then select **Date** or **Time** and make changes in the entry field, or tap (+) to expand either the Date or Time parameter, select the setting to be changed, then select a value from the drop-down list or enter a new value to change this setting.

SNMP OID

Date:	1.3.6.1.4.1.1963.15.501.2.1.0
Time:	1.3.6.1.4.1.1963.15.501.2.2.0

Options

Date	Year	0000`999	*(1999)*
	Month	1-12 *(6)*	
	Day	1-31 *(1)*	
Time	Hour	0-23 *(0)*	
	Minute	0-59 *(00)*	
	Second	0-59 *(00)*	



10.10.111.9
Date <20020617>
Time <112605>

Backlight Timeout

Sets the length of time that the display backlight remains on. If you set a longer timeout value, you use the battery power at a faster rate.

Action

Click the **Display** tab, then select an option from the **Backlight timeout** drop-down list.

SNMP OID

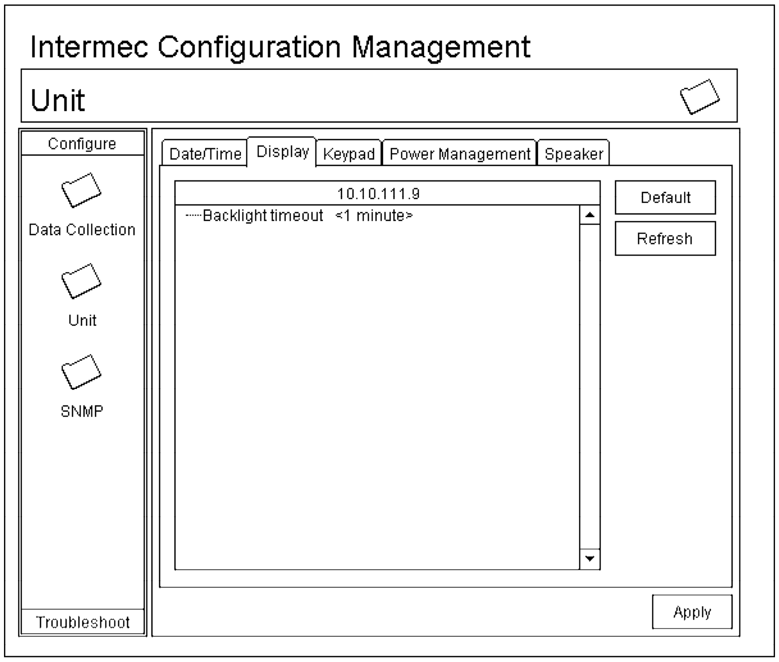
1.3.6.1.4.1.1963.15.13.1.0

Syntax

`DFdata`

where *data* is any of the following:

10	10 seconds
30	30 seconds
60	1 minute *(default)*
120	2 minutes
180	3 minutes
240	4 minutes
300	5 minutes



10.10.111.9
Backlight timeout <1 minute>

Key Clicks

Enables or disables the keypad clicks. The 700 Series Computer emits a click each time you press a key or decode a row of a two-dimensional symbology.

Action

Click the **Keypad** tab, then select an option from the **Key clicks** drop-down list.

SNMP OID

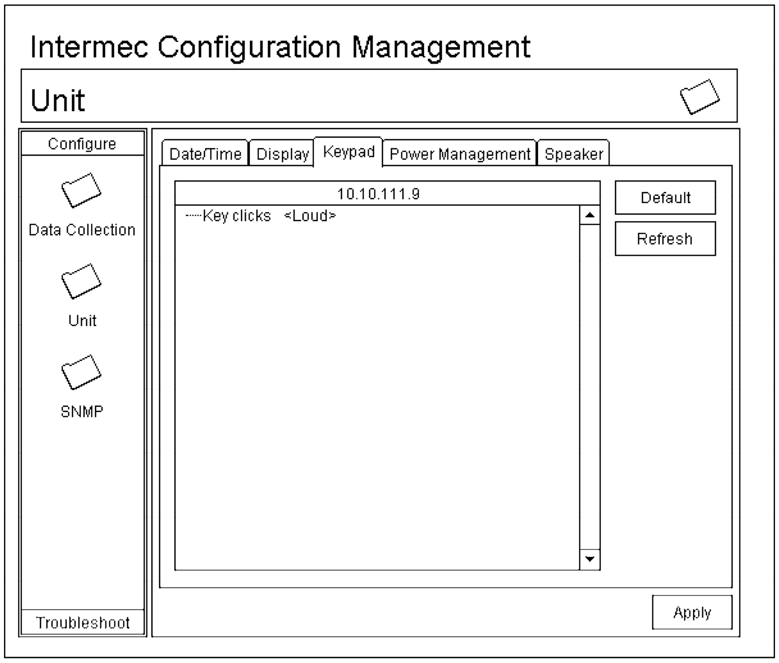
1.3.6.1.4.1.1963.15.12.1.0

Syntax

`KCdata`

where *data* is any of the following:

0	Disable clicks
1	Enable soft key clicks
2	Enable loud key clicks *(default)*



10.10.111.9

Key clicks <Loud>

Automatic Shutoff

Sets the length of time the 700 Series Computer remains on when there is no activity. When you turn on the 700 Computer, it either resumes exactly where it was when you turned it off or boots and restarts your application.

Action

Click the **Power Management** tab, then select an option from the **Automatic shutoff** drop-down list.

SNMP OID

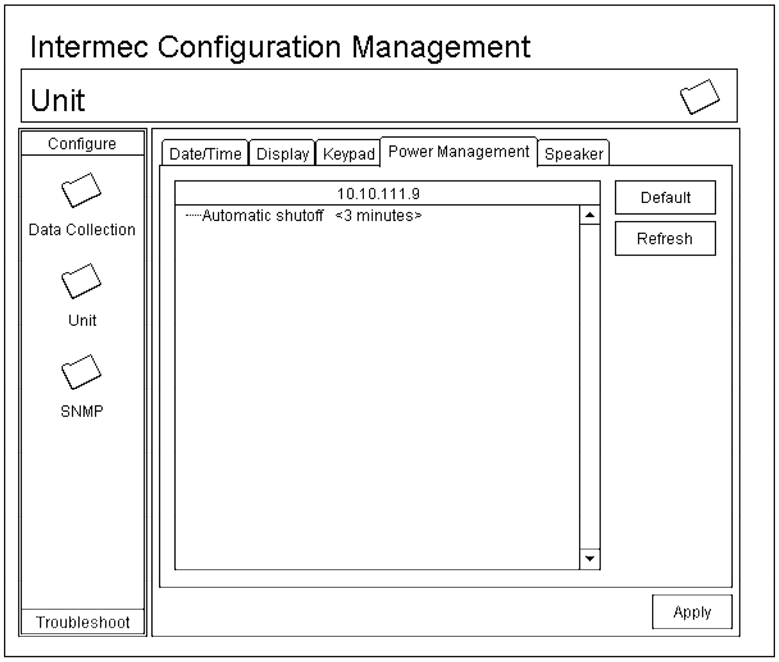
1.3.6.1.4.1.1963.15.11.3.0

Syntax

`EZdata`

where *data* is any of the following:

1	1 minute
2	2 minutes
3	3 minutes *(default)*
4	4 minutes
5	5 minutes



10.10.111.9

Automatic shutoff <3 minutes>

Volume

Changes the volume of all audio signals.

Action

Click the **Speaker** tab, then select an option from the **Volume** drop-down list.

SNMP OID

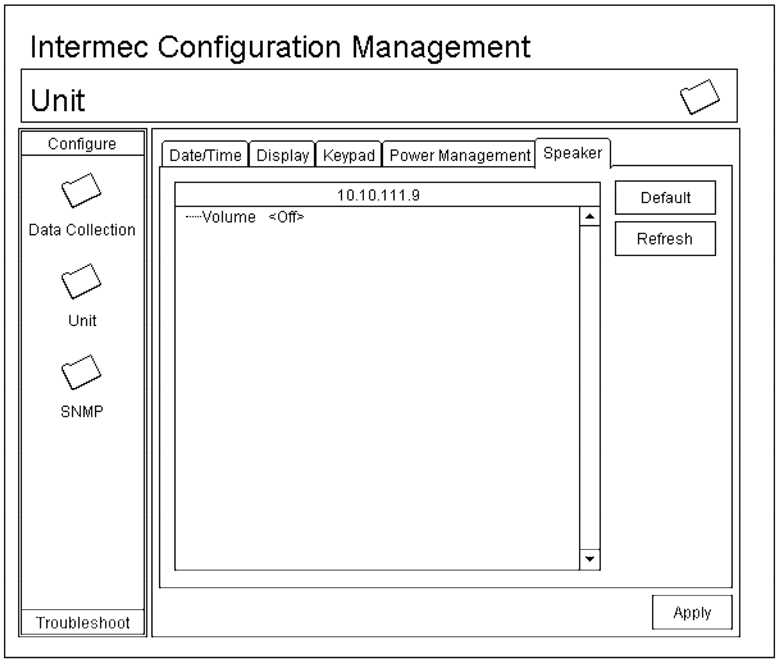
1.3.6.1.4.1.1963.15.3.1.3.0

Syntax

`BVdata`

where *data* is any of the following:

0	Off
1	Very quiet
2	Quiet
3	Normal *(default)*
4	Loud
5	Very loud



10.10.111.9

Volume <Off>

Using Reader Commands

After the 700 Series Computer is connected to your network, you can send the 700 Series Computer a reader command from an application to perform a task, such as changing the time and date. Some reader commands temporarily override the configuration settings and some change the configuration settings.

Change Configuration

The Change Configuration command must precede any configuration command. If you enter a valid string, the 700 Series Computer configuration is modified and the computer emits a high beep. To send the Change Configuration command through the network, use the `$+ [command]` syntax where *command* is the two-letter command syntax for the configuration command followed by the value to be set for that command.

You can also make changes to several different commands by using the `$+ [command]...[command n]` syntax. There are seven configuration command settings that can be changed in this way. *See each command for information on respective acceptable "data" values.*

Command	Syntax
Audio Volume	BV*data*
Automatic Shutoff	EZ*data*
Backlight Timeout	DF*data*
Key Clicks	KC*data*
Virtual Wedge Grid	AF*data*
Virtual Wedge Postamble	AE*data*
Virtual Wedge Preamble	AD*data*

Note: See Appendix A, "*Control Panel Applets*" for more information about the **Virtual Wedge Postamble** and **Virtual Wedge Preamble** commands.

Example 1

To change the Beep Volume to Off, you can send this string to the 700 Series Computer through the network: `$+BV0`
where:

`$+`	Indicates Change Configuration.
`BV`	Specifies the Audio Volume parameter.
`0`	Specifies a value of Off.

Example 2

To change the Beep Volume to Very Quiet and the Virtual Wedge Grid to 123: `$+BV1AF123`
where:

`$+`	Indicates Change Configuration
`BV1`	Specifies Audio Volume, set to Very Quiet (1)
`AF123`	Specifies Virtual Wedge Grid, set to a value of 123.

Set Time and Date

This command sets the date and time on the 700 Series Computer. The default date and time is *June 1, 1999 at 12:00 AM*.

From the network, send the following:

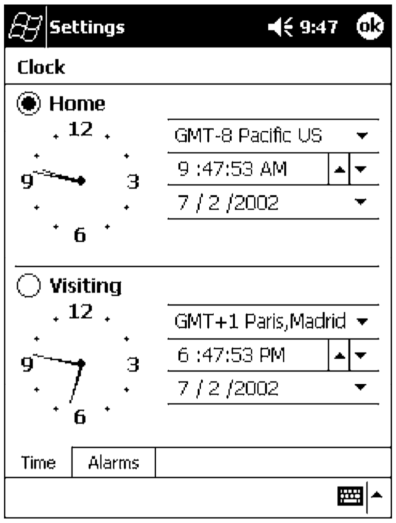
```
/+ yyyymmddhhmmss
```

where acceptable values for the date are:

`yyyy`	0000-9999	Year
`mm`	01-12	Month of the year
`dd`	01-31	Day of the month
`hh`	00-23	Hour
`mm`	00-59	Minutes
`ss`	00-59	Seconds

Clock

You can also set the time and date by using Configuration Management in Unit Manager, or by using the **Clock** control panel applet in the Settings menu. To access this control panel applet, tap **Start** → **Settings** → the **System** tab → the **Clock** icon to access its control panel applet.


Settings 9:47 ok

Clock

● Home
GMT-8 Pacific US
9 :47:53 AM
7 / 2 /2002

○ Visiting
GMT+1 Paris,Madrid
6 :47:53 PM
7 / 2 /2002

Time | Alarms

C Bar Codes

This appendix contains a brief explanation of some of the bar code symbologies that the 700 Series Color (700C) Mobile Computer decodes and explains some of the general characteristics and uses of these bar code types. It also includes several bar code labels that can be scanned into your 700 Series Computer.

Bar Code Symbologies

Specific bar code algorithms can be enabled using the setup menus or the host computer. Once the computer correctly decodes a bar code, the computer encodes data with descriptive information about the symbol. Response time is improved by limiting the computer to the bar codes being used.

Bar Code Data String Formats

Data Bar Code Type	Data Format	Data Length
UPC short (UPC-E)	nddddddc	8
EAN short (EAN-8)	fndddddc	8
UPC long (UPC-A)	nddddddddddc	12
EAN long (EAN-13)	fnddddddddddc	13
UPC short add-on 2	nddddddcaa	10
EAN short add-on 2	fndddddcaa	10
UPC long add-on 2	nddddddddddcaa	14
EAN long add-on 2	fnddddddddddcaa	15
UPC short add-on 5	nddddddcaaaaa	13
EAN short add-on 5	fndddddcaaaaa	13
UPC long add-on 5	nddddddddddcaaaaa	17
EAN long add-on 5	fnddddddddddcaaaaa	18
Interleaved 2 of 5	d......d	Scan device dependent
Standard 2 of 5	d......d	Scan device dependent
Plessey	d......dc	Scan device dependent
Codabar	sd....ds	Scan device dependent
Code 11	d......d	Scan device dependent
Code 39	d......d	Scan device dependent
Extended Code 39	d......d	Scan device dependent
Code 93	d......d	Scan device dependent
Code 128	d......d	Scan device dependent

Note: These bar code data definitions apply to the Data Format column in the previous table

a	Add-on code digits
c	Check digits
d	Bar code digits
f	EAN flag 1 characters
n	Number system digits
s	Start and stop digits

If MOD 10 or MOD 11 check digits are enabled, the digit falls at the end of a bar code data string. Each check digit enabled extends the bar code data string length by one character.

The 700 Series Computer recognizes eleven of the most widely used bar code symbologies. With bar code symbologies, like languages, there are many different types. A bar code symbology provides the required flexibility for a particular inventory tracking system.

A symbology may be for particular industries, such as food and beverage, automotive, railroad, or aircraft. Some of these industries have established their own bar code symbology because other symbologies did not meet their needs.

Without going into great detail on the bar code structure, note that no two products use the same bar code. Each product gets a unique bar code.

Industries that use a particular type of bar code symbology have formed regulating committees or are members of national institutes that issue and keep track of bar codes. This ensures that each organization that contributes to a particular industry conforms to its standard. Without some form of governing body, bar coding would not work.

- UPC (Universal Product Code) with/without add-ons
- EAN (European Article Numbering Code) with/without add-ons
- Codabar
- C11 (Code 11)
- C39 (Code 39)
- C93 (Code 93)
- C128 (Code 128)
- I 2 of 5 (Interleaved 2 of 5 Code)
- S 2 of 5 (Standard 2 of 5)
- Plessey
- MSI (a variant of Plessey)

UPC

The UPC (Universal Product Code) is the symbology used throughout the grocery and retail industries. This bar code symbology contains two pieces of numerical information encoded on the bar code, producer identification, and product identification information.

The UPC symbol is 12 characters long. The first character of the UPC symbol is a number system character, such as “0” for grocery items and “3” for drug- and health-related items.

The UPC symbology is for retail environments such as grocery stores, convenience stores, and general merchandise stores.

Some retail items are so small that a standard UPC bar code cannot fit on the packaging. When this occurs there is a permitted shorter version of the UPC symbology, referred to as UPC-E. UPC-E is six characters long (eight including number system and check digit), approximately half the size of a standard UPC bar code.

EAN

EAN (European Article Numbering) symbology is similar to UPC symbology, except that it contains 13 characters and uses the first two to identify countries.

The EAN symbology is used in the retail environment throughout most of Europe. Though similar to UPC symbology, these are not interchangeable.

Codabar

Codabar was for retail price-labeling systems. Today it is widely accepted by libraries, medical industries, and photo finishing services.

Codabar is a discrete, self-checking code with each character represented by a stand-alone group of four bars and three intervening spaces.

Four different start or stop characters get defined and designated “a”, “b”, “c”, and “d”. These start and stop characters are constructed using one wide bar and two wide spaces. A complete Codabar symbol begins with one of the start or stop characters followed by some number of data characters and ending in one of the start or stop characters.

Any of the start or stop characters may be used on either end of the symbol. It is possible to use the 16 unique start or stop combinations to identify label type or other information.

Since Codabar is variable-length, discrete, and self-checking, it is a versatile symbology. The width of space between characters is not critical and may vary significantly within the same symbol. The character set consists of “0” through “9”, “-”, “$”, “:”, “/”, “.”, and “+”.

The specific dimensions for bars and spaces in Codabar optimize performance of certain early printing and reading equipment. Codabar has 18 different dimensions for bar and space widths. So many different dimensions often result in labels printed out of specification and cause Codabar printing equipment to be more expensive.

Code 11

Code 11 satisfies the requirements for a very high density, discrete numeric bar code. The name Code 11 derives from 11 different data characters that can be represented, in addition to a start or stop character.

The character set includes the 10 digits and the dash symbol. Each character is represented by a stand-alone group of three bars and two intervening spaces. Although Code 11 is discrete, it is not self-checking. A single printing defect can transpose one character into another valid character. One or two check digits obtain data security.

The specifications for Code 11 suggest that this code should have a narrow element width of 7.5 mils. This results in an information density of 15 characters per inch.

Code 39

Code 39 (C39) is the most widely used symbology among the industrial bar codes. Most major companies, trade associations, and the federal government find this code to fit their needs. The main feature of this symbology is the ability to encode messages using the full alphanumeric character set, seven special characters, and ASCII characters.

Programming for this symbology can be for any length that the application requires. The application program for the 700 Series Computer handles symbology at least one character but no more than 32 characters in length.

When programming the computer for Code 39, it is important to set the symbology limit as close as possible (minimum and maximum bar code lengths being scanned). Doing so keeps the computer bar code processing time to a minimum and conserves battery power.

Bar code readers can respond to Uniform Symbology Specification symbols in non-standard ways for particular applications. These methods are not for general applications, because of the extra programming required. Code 39 Full ASCII is one example of non-standard code.

Note: See page 368 to scan several Code 39 bar code labels available to change settings on your 700 Series Computer.

Encoded Code 39 (Concatenation)

If the first data character of a symbol is a space, the reader may be programmed to append the information contained in the remainder of the symbol to a storage buffer. This operation continues for all successive symbols that contain a leading space, with messages being added to the end of previously stored ones. When a message is read which does not contain a leading space, the contents are appended to the buffer, the entire buffer is transmitted, and the buffer is cleared.

Encoded Code 39 (Full ASCII)

If the bar code reader is programmed for the task, the entire ASCII character set (128 characters) could be coded using two character sequences: a symbol (“$”,“.”,“%”,“/”) followed by a letter of the alphabet.

Code 93

The introduction of Code 93 provided a higher density alphanumeric symbology designed to supplement Code 39. The set of data characters in Code 93 is identical with that offered with Code 39. Each character consists of nine modules arranged into three bars and three spaces.

Code 93 uses 48 of the 56 possible combinations. One of these characters, represented by a square, is reserved for a start or stop character, four are used for control characters, and the remaining 43 data characters coincide with the Code 39 character set. An additional single module termination bar after the stop character concludes the final space.

Code 93 is a variable length, continuous code that is not self-checking. Bar and spaces widths may be one, two, three, or four modules wide. Its structure uses edge-to-similar-edge decoding. This makes the bar code immune to uniform ink spread, which allows liberal bar width tolerances.

Code 93 uses two check characters. Its supporters believes this makes it the highest density alphanumeric bar code. The dual check digit scheme provides for high data integrity. All substitution errors in a single character are detected for any message length.

Code 128

Code 128 (C128) is one of the newest symbologies used by the retail and manufacturing industries. It responds to the need for a compact alphanumeric bar code symbol that could encode complex product identification.

The fundamental requirement called for a symbology capable of being printed by existing data processing printers (primarily dot-matrix printers) that produce daily, work-in-progress, job, and product traceability documents. The ability to print identification messages between 10 and 32 characters long, on existing forms and labels deemed an important requirement.

Code 128 uniquely addresses this need as the most compact, complete, alphanumeric symbology available.

Additionally, the Code 128 design with geometric features, improves scanner read performance, does self-checking, and provides data message management function codes.

Code 128 encodes the complete set of 128 ASCII characters without adding extra symbol elements. Code 128 contains a variable-length symbology and the ability to link one message to another for composite message transmission. Code 128, being a double-density field, provides two numeric values in a single character.

Code 128 follows the general bar code format of start zone, data, check digit, stop code, and quiet zone. An absolute minimum bar or space dimension of nine mils (0.010 inch minimum nominal ± 0.001 inch tolerance) must be maintained.

Characters in Code 128 consist of three bars and three spaces so that the total character set includes three different start characters and a stop character.

UCC/EAN-128 Shipping Container Labeling is a versatile tool that can ease movement of products and information. The Shipping Container Labeling bar code can take any form and usually has meaning only within the company or facility where applied.

Because this *random* data can get mistaken later for an industry standard code format, the UCC and EAN chose a symbology uniquely identified from these other bar codes. This standard is for maximum flexibility, to handle the diversity of distribution in global markets by cost efficiency.

The UCC/EAN-128 Container Labeling specification calls for a FUNC1 to immediately follow the bar code' s start character. FUNC1 also follows any variable-length application field. The specification also calls for the computer to send "]C1" for the first FUNC1. The specification requires that the computer send a "<GS>" (hex 1D) for subsequent FUNC1 codes in the bar code.

Because "<GS>" is not compatible with computer emulation data streams, the Uniform Code Council has been asked to change the specification. This change is made to send the same three character sequence "]C1" to identify the embedded FUNC1 codes.

This implementation should provide for clean application coding by identifying the same sequences for the same scanned codes. If the communication of Norand bar code types is enabled, the Shipping Container Label codes precede with a "J". These strings will appear on the computer display. The application may have to allow for strings longer than 48 characters (maximum length indicated in the specification). Actual length variance depends on the number of variable-length data fields. Allowing for 60 characters should be sufficient. Within the Code 128 specification, the computer can link bar codes together. If this is to happen, allow for more characters (computer limit is 100 characters).

The Application Identifier Standard, that is part of the UCC/EAN Shipping Label concept, complements, rather than replaces, other UCC/EAN standards. Most UCC/EAN standards primarily identify products.

Several industries expressed the need to standardize more than product identification. The UCC/EAN Code 128 Application Identifier Standard supplies this tool. The standard adds versatility for inter-enterprise exchanges of perishability dating, lot and batch identification, units of use measure, location codes, and several other information attributes.

For more detailed information on Code 128 UCC/EAN Shipping Label bar code and Application Identifier Standard, refer to the UCC/EAN-128 Application Identifier Standard specification.

I 2 of 5 (Interleaved)

I 2 of 5 (Interleaved 2 of 5 Code) is an all-numeric symbology, widely used for warehouse and heavy industrial applications. Its use has been particularly prevalent in the automobile industry. The I 2 of 5 symbology can be placed on smaller labels than what the standard UPC symbology requires.

I 2 of 5 also provides a little more flexibility on the type of material it can print on. Interleaved 2 of 5 Code has its name because of the way the bar code is configured.

I 2 of 5 bars and spaces both carry information. The bars represent the odd number position digits, while spaces represent the even number position digits. The two characters are interleaved as one. Messages encoded with this symbology have to use an even number of characters since two numeric characters always get interleaved together.

S 2 of 5 (Standard 2 of 5)

The code S 2 of 5 (Standard 2 of 5 Code) is designed primarily for:

- Warehouse inventory handling
- Identification of photo finishing envelopes
- Airline tickets
- Baggage and cargo handling

The code S 2 of 5 is simple and straightforward. All information is contained in the widths of the bars, with the spaces serving only to separate the individual bars.

Bars can either be wide or narrow, and the wide bars are usually three times the widths of the narrow bars. Spaces may be any reasonable width but are typically equal to the narrow bars. Narrow bars are identified as zero bits and wide bars as one bits.

Remember the code structure by associating the bar positions from left to right with weighting factors 1, 2, 4, 7, and parity. Exceptions to this rule are zero, start, and stop. This code is a discrete code, since the white spaces between the characters are not part of the code. Because the white spaces carry no information, their dimensions are not critical.

The S 2 of 5 code is self-checking, meaning a scanner passing through a printing void would detect the proper ratio of wide bars to total bars. When the scanner spots an error, a non-read will occur.

Plessey

Plessey finds its origin in the pulse width modulated (PWM) code developed in England. It is widely used for shelf markings in grocery stores. Pulse width modulated codes represent each bit of information by a bar and space pair. A zero bit consists of a narrow bar followed by a wide space, while a one bit consists of a wide bar followed by a narrow space. It is mainly a numeric symbology (0-9) with six extra characters available for assigning any symbol or letter desired.

Plessey codes are not self-checking and employ a variety of check characters. Plessey employs a polynomial-based Cyclic Redundancy Check (CRC). For start and stop characters, Plessey employs a 1101 and previously used a 0101.

This symbology is very limited about what information can be encoded. It is not considered for new applications.

MSI Code (Variant of Plessey)

In addition to Plessey characteristics, the MSI Code employs a Modulus 10 Check. For start and stop checks, MSI employs a single bit pair of 1 as a start symbol and a single bit pair of 0 as a stop symbol. MSI reverses the 1-2-4-8 BCD pattern for bit pair weighting to 8-6-2-1.

Bar Code Labels

You can change some settings on your 700 Series Computer by scanning the following Code 39 bar code labels.

- You can use the Unit Manager application to set the Automatic Shutoff, Volume, Backlight Timer, or Key Clicks parameters *(starting on page 352).*
- You can use the Unit Manager application or the Data Collection control panel to set the three Virtual Wedge parameters *(starting on page 325).*

Note: When you use a bar code creation utility to make a scannable bar code label, the utility probably adds opening and closing asterisks automatically. Asterisks are included here for translation purposes.

Audio Volume

Note: The Audio Volume parameter information is on page 356.

Turn Audio Off

$+BV0

Set Audio Volume to very quiet

$+VB1

Set Audio Volume to quiet

$+BV2

Set Audio Volume to normal *(default)*

$+BV3

Set Audio Volume to loud

$+BV4

Set Audio Volume to very loud

$+BV5

Automatic Shutoff

Note: The Automatic Shutoff parameter information is on page 355.

Set Automatic Shutoff to 1 minute

$+EZ1

Set Automatic Shutoff to 2 minutes

$+EZ2

Set Automatic Shutoff to 3 minutes *(default)*

$+EZ3

Set Automatic Shutoff to 4 minutes

$+EZ4

Set Automatic Shutoff to 5 minutes

$+EZ5

Backlight Timeout

Note: The Backlight Timeout parameter information is on page 353.

Backlight Timeout 10 seconds

$+DF10

Backlight Timeout 30 seconds

$+DF30

Backlight Timeout 1 minute *(default)*

$+DF60

Backlight Timeout 2 minutes

$+DF120

Backlight Timeout 3 minutes

$+DF180

Backlight Timeout 4 minutes

$+DF240

Backlight Timeout 5 minutes

$+DF300

Key Clicks

Note: The Key Clicks parameter information is on page 354.

Disable key clicks

$+KC0

Enable soft key clicks

$+KC1

Enable loud key clicks *(default)*

$+KC2

Virtual Wedge Grid, Preamble, Postamble

The following parameters are user-configurable strings. Refer to a full ASCII chart for more information.

Grid

For Virtual Wedge Grid, the first part of the bar code would be the following, which can include a string of up to 240 characters. *Parameter information starts on page 328.*

*$+AF

Preamble

For Virtual Wedge Preamble, the first part of the bar code would be below, followed by a string of up to 31 characters *(no <NUL>)* and an asterisk. *Default is no characters. Parameter information is on page 326.*

*$+AD

Postamble

For Virtual Wedge Postamble, the first part of the bar code would be below, followed by a string of up to 31 characters *(no <NUL>)* and an asterisk. *Default is no characters. Parameter information is on page 327.*

*$+AE

I Index

The Classes and Functions Index covers classes and functions for the 700 Series Color Mobile Computer.

The General Index covers all topics. Those in italics are figures, those in bold are tables.

The Files Index is to assist you in locating descriptions for device drivers, applications, utilities, batch files, or other files within this publication.

Classes and Functions

General Index

Files Index

Corporate Headquarters
6001 36th Avenue West
Everett, Washington 98203
tel 425.348.2600
fax 425.355.9551
www.intermec.com

700 Series Color Mobile Computer User's Manual - November 2002

961054031 REV B