Peracom

USB to Serial Dongle Specification

Preliminary

Version 0.3

Author: Ken Rivage

Contacts: Jack Chorpenning
John Beidl
Ken Rivage

SEPTEMBER 1, 1997

Peracom • 3800 Gateway Center • Suite 301 • Morrisville NC, 27560 • (919) 469-9320

PREFACE

This document describes the Peracom USB to Serial Dongle from a hardware perspective. The major devices used in the design are not described in detail and the user should reference the appropriate vendor specification. Please refer to the USB Specification on the overall description of how USB operates.

INTRODUCTION

The Peracom USB to Serial Dongle (referred to herein as Dongle) is a standalone device that allows a legacy RS232 peripheral to attach to a USB Host controller on a PC or a USB hub. The Dongle will consist of one USB Device controller, a 16C450/16C550 compatible UART, and RS232 Transceivers. The device will use a USB type B connector for connection to the USB host and a DB 25 pin connector for connection to the RS232 peripheral. Figure 1 below shows the high level architecture of the device.

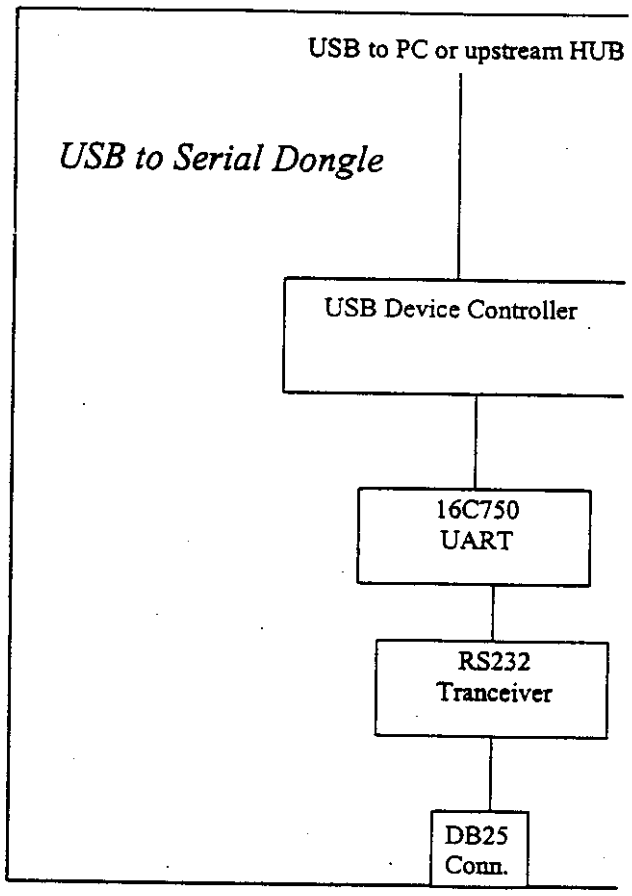


Figure 1

FUNCTIONAL OVERVIEW

The Dongle will be fully compatible with today RS232 designs using 16C550 or comparable devices. The unit will initially require a device driver at the host to translate serial host commands into USB specific commands. The USB device controller will then convert these host USB commands into UART functions.

RS232 SUPPORT

The Dongle will support all popular baud rates up to 115Kbps. The Dongle will contain transmit and receive buffers to allow the maximum baud rate and synchronization with the 1ms USB frame.

USB INTERFACE

The Dongle will be a low bus powered device (less than 100ma) that attaches to an upstream USB port. The interface will have separate endpoints for control, transmit and receive.

Production versus Prototype Design

The production unit will consist of one Intel 8x930Ax USB device controller, a 16C750 UART, a RS232 transceiver, an upstream type B USB connector and a DB 25 pin RS232 connector. However, since the Intel USB device controller contains a MASK only ROM area, the prototype will require an external ROM for debug purposes. Initial production versions will contain the microcode in an external PROM until it is cost justified to produce a Mask ROM version of the 8x930Ax.

Mechanical Packaging and Covers

The Dongle will be packaged in a small plastic housing similar to popular gender changers for parallel or serial connections. The design should not require any EMI coatings or protective barriers. All logo should be silkscreen to allow customization for potential OEM customers. The figure 2 shows the mechanical representation of the device.

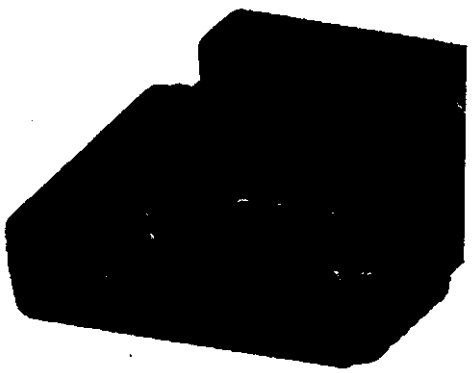


Figure 2

Product Packaging

The product will need to be packaged for both OEM and brand name retail sales.

USB Device Controller Configuration

The Dongle uses an Intel 8x930AE USB Device Controller. Refer to the Intel 8x930Ax specification for detailed information on the USB Device Controllers.

General 8x930AE Description

The Intel 8x930AE is a USB device controller that contains an integrated MCS251 microcontroller, 1 KB RAM, 16KB MASK ROM, up to six endpoint pairs with a combined receive and transmit FIFO of 1KB. The microcontroller is responsible for data transfer between the UART and the USB FIFOs. The microcontroller is also responsible for setting up the control registers in the UART via host USB commands.

Endpoint Description

The USB controller (8x930AE) supports up to six endpoint pairs for communication with attached devices. The table below describes the endpoint configuration for the Dongle.

End Point #	Direction	End Point Type	FIFO size (bytes)	Description
0	In/Out	Control	8	USB Device Setup & Register Access
1	In	Bulk	16	UART Data Receive
1	Out	Bulk	16	UART Data Transmit
2	In	Interrupt	16	Interrupt for UART & USB Controller
3		N/A	N/A	Not Used
4		N/A	N/A	Not Used
5		N/A	N/A	Not Used

Table 1

UART Register Access

UART setup data is encapsulated in a USB Setup packet. The following is the layout of the setup packet with information to access the UART control registers.

Get Register

This request returns the specified 16C550 register contents.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000000B	GET_REGISTER (0x40)	0	Register Index	1	Register Data

Set Register

This request sets the specified 16C550 register to the value supplied under mask.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000000B	SET_REGISTER (0x41)	Register Mask and Register Value	Register Index	0	None

bmRequestType

The bmRequestType is used to define the direction of the transfer, the type of transfer and the recipient of the request. The below table shows the values for the bmRequestType.

Data bit	Value	Comments
D7	Data transfer direction 0 = Host to device 1 = Device to host	Sets direction for read or write of register data to/from host or UART.
D6..5	Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved	Value should be Vendor (2) to select the UART registers.
D4..0	Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4..31= Reserved	Value should be Device (0) to select the UART registers.

Table 2

bRequest

This value should be set to 0x40 for UART register reads and 0x41 for UART register writes.

wValue

For UART register writes (from host to device) this field specifies the mask to be used (high byte) and the value to be set (low byte). Only the bits specified by binary 1 mask bits will be altered. All unmasked bits will remain unaltered.

For UART register reads this field is reserved and must contain 0

wIndex

This value specifies the UART register to be accessed.

Value	Modem Register Name
0	THR – Transmitter Buffer Register (write only) RBR – Receiver Buffer Register (read only)
1	IER – Interrupt Enable Register
2	IIR – Interrupt Identify Register (read only)
3	LCR – Line Control Register
4	MCR – Modem Control Register
5	LSR – Line Status Register
6	MSR – Modem Status Register
7	SCR – Scratch Register
8	DLM – Divisor Latch MSB Register
9	DLL – Divisor Latch LSB Register

Table 3

wLength

For UART register reads, this values specifies the amount of data returned in the data phase and should be one. For UART register writes, this value should be zero.

UART Data Transfer

Data is transferred from the host to the UART via dedicated endpoints for each direction. Table 1 shows the list of endpoints associated with the data pipes for the UART. The endpoints associated with data transfer are Bulk endpoints. Transmit data transfers are Out token packets followed by OUT data packets. Receive data transfers are IN token packets followed by IN data packets. The actual data is transferred during the data phase (IN or OUT) of the bulk transfer.

Interrupt Transfer

The interrupt transfer are used to indicate status changes and event information to the UART. An interrupt transfer is an IN transfer. The 8x930AE can respond with a NAK (no interrupt pending), data (interrupt), or a Stall (problem). The data will specify an interrupt value, and optional parameters depending on the interrupt value. The interrupt values and parameters are TBD.

DONGLE INTERNAL CONFIGURATION

The previous section described the Dongle from the USB host perspective. This section will describe the Dongle from the 8x930AE point of view.

USB Device Controller Configuration

The Dongle uses an 8x930AE USB Device Controller. Refer to the Intel 8x930Ax specification for detailed information on the USB Device Controllers.

General 8x930AE Description

The Intel 8x930AE is a USB device controller that contains an integrated MCS251 microcontroller, 1 KB internal RAM, 16KB internal ROM, and up to six endpoint pairs with a combined receive and transmit internal FIFO of 1KB. The microcontroller is responsible for data transfer between the UART and the USB FIFOs. The microcontroller is also responsible for setting up the control registers in the UART via host USB commands.

8x930Ax and UART Communication

The 8x930Ax microcode will poll the UART to determine transmit and receive status. There will an interrupt between the UART and the 8x930Ax for remote wake up via Ring Indicate.

UART Reset

The UART Reset is controlled by microcode from the 8x930. The 8x930 programmable port bit P1.5/CEX2 is used for UART Reset. Reset is asserted when P1.5/CEX2 = 1.

The P1.5/CEX2 pin which is programmable via control registers in the 8x930. When the system is reset, the software will need to reset the UART.

USB Suspend and Resume

When the USB enters suspend mode, the 8x930 will shut down the RS232 tranceivers, the UART, and then put the 8x930 into sleep mode. An external Ring Indicate signal can wake up the 8x930 via an interrupt.

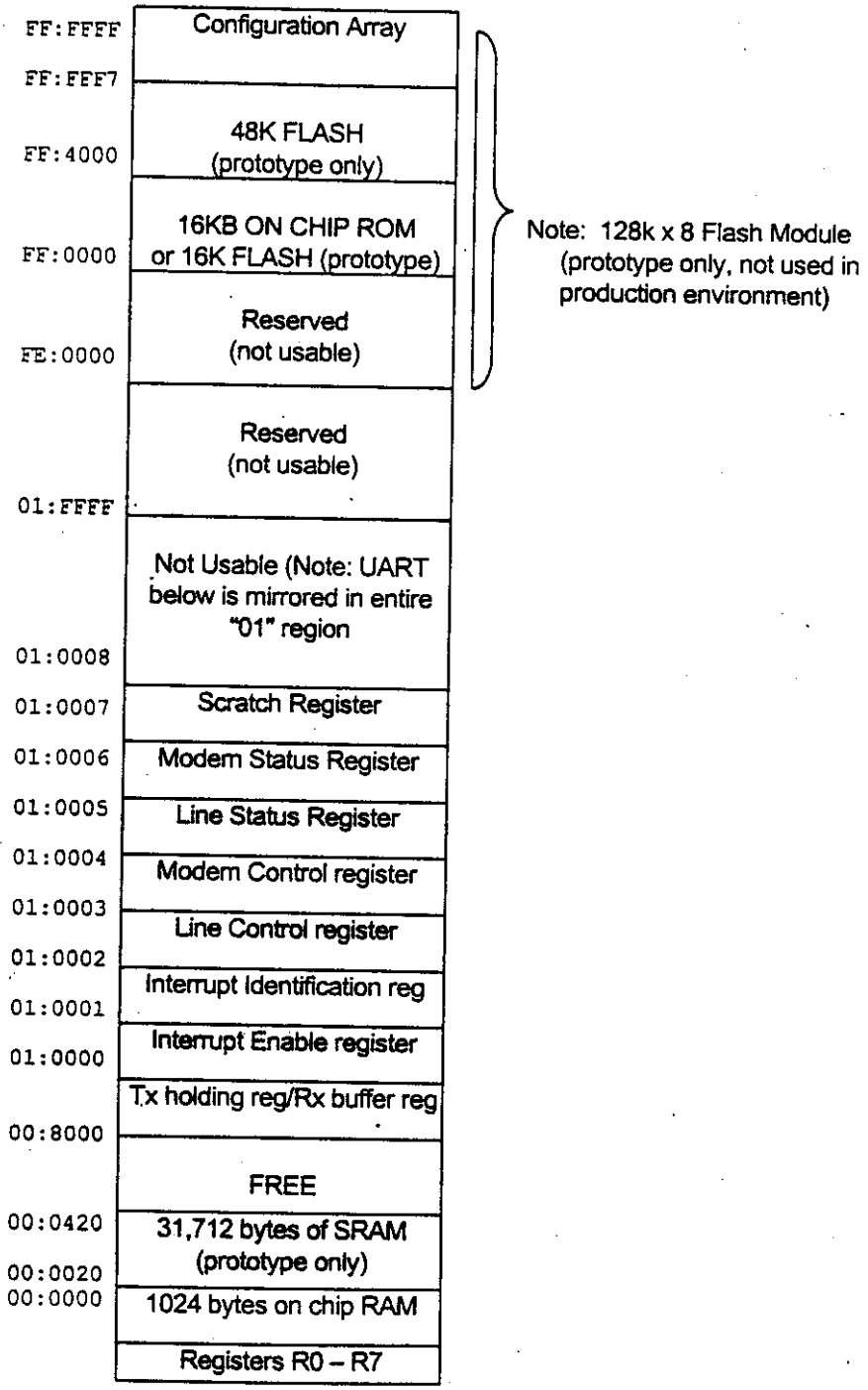
Memory and UART Wait States

The 8x930Ax will execute in low clock mode to meet the power requirements for attaching to a bus powered hub. The low clock mode extends the memory cycles by a factor of four, so no wait states are required for memory or the UART.

8x930AE Memory Map

The microcontroller contains three distinct address spaces for accessing the Register File, Special Function Registers (SFRs), and memory. Please refer to the Intel 8x930Ax specification for detailed information on the Register File and SFRs.

The following figure outlines the memory map of the 8x930AE.



Host Programming Model

The serial dongle closely represents the proposed implementation of a USB Communication Class device. A vendor specific solution is required since this Class will not contain any host driver support in the near future and has not been approved by the USB-IF Device Working Group. This Communication Class, however, should be used as a architectural framework for implementation of the Serial Dongle to enable future models to take advantage of any available class drivers. The following describes the Host USB commands and serial data encapsulation.

Communication Classes

There are currently three different types of Communication Classes defined in the Communication Class Specification version 0.9. These are Direct Line Interface, Abstract Control Interface, and Telephone Control Interface. The serial dongle will implement the Abstract Control Interface along with additional vendor specific commands to achieve the appropriate serial function.

Abstract Control Interface Commands

The table below represents the valid commands for the Abstract control interface, all other commands sent to this interface should generate a stall.

Communication Class-Specific Requests - Abstract Control Interface

Request	Code	Description	Req /Opt
SEND_ENCAPSULATED_ COMMAND	00h	Issue a command in the format of the supported control protocol.	Required
GET_ENCAPSULATED_ RESPONSE	01h	Request a response in the format of the supported control protocol.	Required
UART_SET_LINE_CODING	14h	Configure DTE rate, stopbits, parity, and number of character bits.	Optional *
UART_SET_CONTROL_ LINE_STATE	15h	RS232 signal used to tell the DTE device the DCE device is now present.	Optional
UART_SEND_BREAK	16h	Send special carrier modulation used to specified RS232 style break.	Optional
UART_GET_LINE_CODING	17h	Request current DTE rate, stopbits, parity and number of character bits.	Optional *

- * It is strongly recommended to support these requests for an analog modem.

Communication Class-Specific Notifications - Abstract Control Interface

Notification	Code	Description	Req /Opt
NETWORK_CONNECTION	00h	Notification to host of network connection status	Optional *
RESPONSE_AVAILABLE	01h	Notification to host that a bulk request to retrieve a response should be issued.	Required
UART_STATE	14h	Returns current state of carrier detect, DSR, break and ring signal.	Optional *

- * It is strongly recommended to support these requests for an analog modem.

Abstract Control Interface Command Messages

GetEncapsulatedResponse

This request is used to request a response in the format of the supported control protocol of the Communication Class interface. It can be sent to the device, a Communication Class interface or an association.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100000B 10100001B 10100011B 10100100B	GET_ENCAPSULATED_ RESPONSE	Zero (unused)	Zero Interface Other Association	Amount of Data in Bytes associated with this recipient.	Protocol dependent data response

SendEncapsulatedCommand

This request is used to issue a command in the format of the supported control protocol of the Communication Class interface. It can be sent to the device, a Communication Class interface or an association.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B 00100001B 00100011B 00100100B	SEND_ENCAPSULATED _COMMAND	Zero	Zero Interface Other Association	Amount of Data in Bytes associated with this recipient.	Control protocol based command

UARTSetControlLineState

This is used to generate an RS232 style signal used to tell the DCE that the DTE device is now present.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B 00100001B 00100011B 00100100B	UART_SET_CONTROL_ LINE_STATE	0 - disconnect 1 - connect	Zero Interface Other Association	Zero	None

UARTGetLineCoding

This allows the host to find out the currently configured line coding.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100001B 10100010B	UART_GET_LINE_ CODING	Zero	Interface Endpoint	4	Properties

The Line Coding Properties are defined as follows:

UART Configuration Bitmap Values

Field	Bits	Description
dwDTERate	D0-D17	Data terminal rate in bits per second
bCharFormat	D18-D21	Stop bits (1, 1.5, 2)
bParityType	D22-D24	parity (none, odd, even, mark, space)
bDataBits	D25-D27	data bits (5, 6, 7, 8 or 16)

UARTSetLineCoding

This request allows the host to specify typical asynchronous line character formatting properties which may be required by some applications. This request applies to asynchronous octet stream data class interfaces and endpoints; and applies to data transfers both from the host to the device and from the device to the host.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B 00100010B	UART_SET_LINE_CODING	Zero	Interface Endpoint	4	LineCoding Properties

See Section 6.2.14 for the definition of the valid properties.

UARTSendBreak

Send special carrier modulation used to generate a RS232 style break.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100000B 00100001B 00100011B 00100100B	UART_SEND_BREAK	Duration of Break	Zero Interface Other Association	Zero	None

The wValue field will contain the length of time in milliseconds of the break signal.

NetworkConnection

Allows the device to notify the host that it is or is not connected to a network.

bmRequestType	bNotification	wValue	wIndex	wLength	Data
10100000B 10100001B 10100010B 10100011B	NETWORK_ CONNECTION	0 - Disconnected 1 - Connected	Zero Interface Other Association	Zero	None

ResponseAvailable

Allows the device to notify the host that a response is available which may be retrieved with *GetEncapsulatedResponse* request.

bmRequestType	bNotification	wValue	wIndex	wLength	Data
10100000B 10100001B 10100010B 10100011B	RESPONSE_AVAILABLE	Zero	Zero Interface Other Association	Zero	None

UARTState

Sends asynchronous notification of UART status.

bmRequestType	bNotification	wValue	wIndex	wLength	Data
10100001B 10100100B 10100010B	UART_STATE	UART Status Bitmap	Interface Association Endpoint	1	None

The *Data* field is a bit mapped value that contains the current state of carrier detect, DSR, break, ring signal and device overrun error. These signals are normally found on a UART and are used for communication status reporting. A state is considered enabled if its respective bit is set to 1.

This notification is used like a real interrupt status register. Once a notification has been sent, the device will reset and re-evaluate the different signals. For the consistent signals like carrier detect or DSR, this will mean another notification won be generated until their state changes. For the irregular signals like break, the incoming ring signal or the overrun error state, this will reset their values to zero and again won send another notification until their state changes.

UART State Bitmap Values

Field	Bits	Description
bCarrierDetect	D0	State of carrier detection mechanism of device
bDSR	D1	Current state of DSR signal
bBreak	D2	State of break detection mechanism of device
bRingSignal	D3	State of ring signal detection of device
bOverRun	D4	Overrun error has occurred in device.
	D5-D7	RESERVED

LIST OF EXHIBITS

Fee Payment Check

FCC Form 731

Letter of Authorization

List of Exhibits

Exhibits A - Technical Document,Diagram

Exhibits B - Photographs

Exhibits C - Measurement Report

Exhibits D - Equipment FCC ID Label

Exhibits E - User's Manual