An Interim report on connecting a USB3343 USB transceiver to the LPC4357 Microcontroller

Abstract:

This report describes the difficulties that the report's author has had in configuring the LPC4537 Microcontroller to work with Microchip's USB transceiver.

Background:

The LPC43xx Microcontrollers from NXP mostly all feature 2 High-Speed USB Controller logic units, however the these 2 controllers are furnished with 1 High-Speed on-chip Phy and 1 Full-Speed on-chip Phy. If the user wishes to employ 2 high-speed Phys then a second High-Speed Phy can be attached to selected MCUs in the family line via an industry standard ULPI bus. When appropriately configured the MCU will direct all USB traffic across the bus to the external Phy. Such configuration is detailed in the Application note AN11309 from NXP.

The author is developing with a LPC4357-Xplorer++ kit from NGX Technologies complimented by a breakout board designed in house fitted with a USB3343 USB transceiver.

The LPC4357 MCU has been configured as per the application note and communication with the transceiver has been established, however normal USB communications are NOT yet working.

We can be fairly confident that the MCU is properly configured for ULPI communications as we are able to read and write the internal registers of the attached Phy and the contents of those registers are as we expect from the data-sheet.

After the USB controller has been initialized and at the moment of connection to a USB Host the contents of those registers are thus:

```
UlpiReg[0x00]=0x24 ; Vendor IDL
UlpiReg[0x01]=0x04 ; Vendor IDH
UlpiReg[0x02]=0x09 ; Product ID L
UlpiReg[0x03]=0x00 ; Product ID H
UlpiReg[0x04]=0x51
UlpiReg[0x07]=0x08
UlpiReg[0x0A]=0x0E
UlpiReg[0x0D]=0x1F
UlpiReg[0x10]=0x1F
UlpiReg[0x13]=0x18
UlpiReg[0x14]=0x0C
UlpiReg[0x15]=0x00
UlpiReg[0x16]=0x55 ; ScratchPad, 55 written by us
UlpiReg[0x19]=0x00
UlpiReg[0x1D]=0x00
UlpiReg[0x20]=0x00
UlpiReg[0x21]=0x00
UlpiReg[0x31]=0x00
UlpiReg[0x32]=0x00
UlpiReg[0x33]=0x00
UlpiReg[0x36]=0x00
UlpiReg[0x39]=0x04
```

Observations:

The most immediate problem with the commencement of a USB connection is the the proper configuration of the USB bus pull-up/pull-down and termination resistors is not taking place and hence the USB device (the LPC dev board) is not being detected by the host.

I have been able to monitor the ULPI bus between the MCU and transceiver and log the transactions that occur there while the transceiver is configured for operation. A transaction log follows below and it shows that the transceiver is misconfigured for the desired operating mode.

The MCU has been configured to operate USB1 in High-Speed Device mode in which the pre-connection state requires D+ to be pulled up but we can see from the logs that D+ is in fact configured for Pull-Down, additionally we note that in fact the configuration of Pull-Up Pull-down and termination resistors meets the criteria of an invalid configuration as defined by the USB3343 data sheet and the behavior of the transceiver in that state is undefined.

In the logs that follow(Table 1) the first column is headed ULPI TxCMD and shows the commands issued by the LPC4357 ROM/Microcode to the Phy. The format of the commands is described well in the datasheet for the Phy but in the logs below we see reads and writes(mostly) to the Phy's internal registers. A write is indicated by MSB[7:6] = 10b, a read by 11b. Bits[5:0] give the register address. The LSB is data to be written to the register or read from the register when the 'dir' signal is asserted.

Many registers have aliased addresses for Read/Write/Set/Clear operations and the effective operation is marked in the second column. The name of the register is given in the third and the effect of the operation or other pertinent information is given in the 4th

The logs capture all the ULPI traffic between board power-on and being ready for USB connection, with the specific MCU commands that elicit ULPI traffic marked down as 'Actions:'

The first burst of activity comes after the GPIO pins are configured for the ULPI interface when the MCU is configured to use the external Phy.

The second burst comes on a USB controller 'reset' during the USB stack Init.

The last burst happens during the Init of the USB software stack.

ULPI TxCMD	R/W/S/C	Register Name	Deviation from Power on default
Action:			
LPC_USB1->PORTSC1_D = 0x80000004			
89 07	C	Interface Control	!6pin serial,!3pin serial, !carkit (defaults)
85 20	S	Function Control	Reset transceiver
84 51	W	Function Control	!SuspendM, Disable bit-stuff & NRZI, Enable FS Xcvr
8C 38	C	OTG Control	Do not drive VBUS, Disconnect resistor VBUS-VD33, disconnect resistor VBUS-GN
8B 07	S	OTG Control	Pull down on DM, Pull down on DP, Pull up on ID
89 07	C	Interface Control	!6pin serial,!3pin serial, !carkit (defaults)
88 08	S	Interface Control	Enable CLKOUT in Serial or CarKit mode
D3 1C	R	USB Interrupt Status	idGnd is high, SessionEnd, SessionValid
D5 00	R	Debug	Line state = 00b
84 41	W	Function Control	!SuspendM, Normal Operation, Enable FS Xcvr
Action:			
LPC_USB1->USBCMD_D = (1 << 1)			
85 20	S	Function Control	Reset transceiver
8C 31	C	OTG Control	Do not drive VBUS, Disconnect resistor VBUS-VD33, Disable Pull up ID
8B 0E	S	OTG Control	Pulldown on VBUS, Pull down on DM, Pull down on DP,
84 D5 (44)	W	Function Control	LPM Enable,!Suspend,!Reset,Disable BS&NRZI, TermSel=1, Xcvr=FS (#funky wave
84 51	W	Function Control	!LPM, !Suspend, !Reset, Disable BS&NRZI, TermSel=0, Xcvr=FS
89 07	C	Interface Control	!6pin serial,!3pin serial, !carkit (defaults)
88 08	S	Interface Control	Enable CLKOUT in Serial or CarKit mode
D3 10	R	USB Interrupt Status	IdGnd
D5 00	R	Debug	Line state = 00b
84 41	W	Function Control	!SuspendM, Normal Operation, Enable FS Xcvr
Action:			
LPC_USB1->USBMODE_D = 0x02 | (1 << 3)			**Setup Lockouts off | Mode = DeviceMode**
no ULPI activity			
Action:			
USBD_API->hw->Init()			
85 20	S	Function Control	Reset transceiver
8C 31	C	OTG Control	Do not drive VBUS, Disconnect resistor VBUS-VD33, Disable Pull up ID
8B 0E	S	OTG Control	Pulldown on VBUS, Pull down on DM, Pull down on DP
84 44	W	Function Control	!SuspendM, Normal Operation, Termination selected, Enable HS Xcvr
84 51	W	Function Control	!LPM, !Suspend, !Reset, Disable BS&NRZI, TermSel=0, Xcvr=FS
89 07	C	Interface Control	!6pin serial,!3pin serial, !carkit (defaults)
88 08	S	Interface Control	Enable CLKOUT in Serial or CarKit mode
D3 10	R	USB Interrupt Status	IdGnd is high
D5 00	R	Debug	Line state = 00b

Table 1: Captured ULPI transactions

Looking at this last section we see that ULPI commands '8B 0E' and '84 51' leave the transceiver configured with XcvrSelect[1:0] = 01, TermSelect[2] = 0, OpMode[4:3] = 10, DpPulldown[1] = 1, DmPulldown[2] = 1 and we note that this configuration does not match any of the valid configurations list in Table 5.1 of the transceiver's datasheet and hence the transceiver behavior is undefined.

As noted adjacent to the '84 D5 (44)' command the waveform(Figure 1) during this Write command shows some anomaly with the 'dir' signal being asserted which would normally indicate a Read from the Phy register. In any case a normal looking write command follows immediately afterwards.

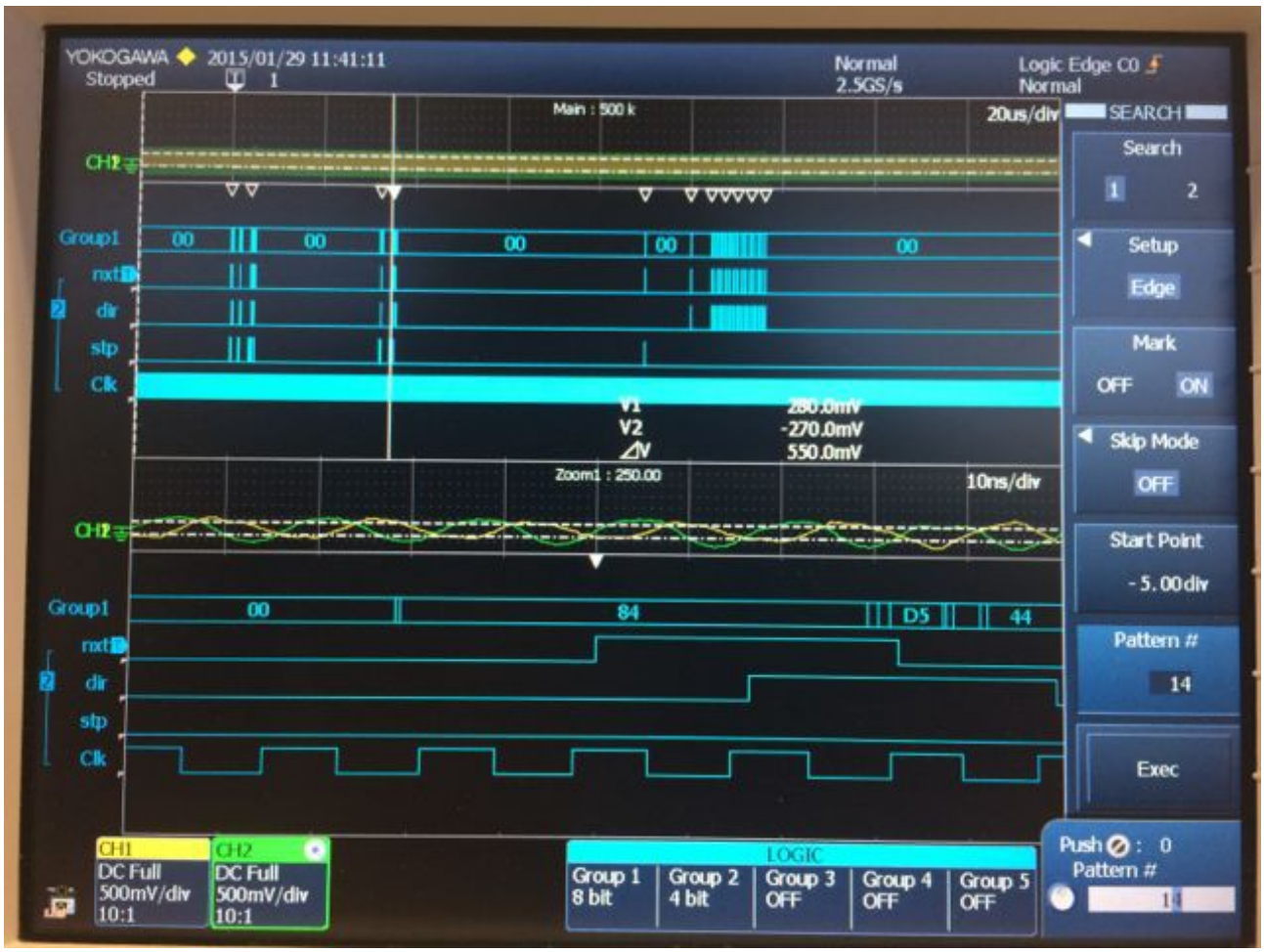

Figure 1: Anomalous 'dir' assertion

Notes:

In searching for information regarding this issue I came across another NXP user who encountered a problem whose symptoms match these. In his case his issue disappeared when he up-reved his silicon from LPC4350 RevA to LPC4350 RevC. In the case of the LPC4357 only 1 Rev exists to my knowledge. His issues are detailed in this forum post:
http://www.lpcware.com/content/forum/high-speed-usb-device-usb1-external-phy

There is no published errata concerning this issue for the LPC4350 or the LPC4357

Further Work:

Pending advice from NXP or other parties I will manually configure the ULPI registers in the Phy setting the pull-ups and termination resistors into valid state and hope that the LPC's USB controller can be coaxed into life.

Feedback:

The author can be contacted at:

chris_bayley@trimble.com