Artec Group OÜ, Türi 10c Tallinn,
11313 Estonia European Union
Tel: (+372) 6718 550
Fax: (+372) 6718 555
www.artecgroup.com
info@artecgroup.com

Programmable LPC Dongle

DATA SHEET

FPGA based hardware debugger platform

Ver. 5

Dongle version 5 Features:

- Supports LPC memory read (can be disabled),LPC Firmware Hub memory read and IO write for POST Code capture
- Postcode only mode (LPC reads from dongle are disabled)
- Postcode logger (sends all postcodes to USB serial port as hexadecimal bytes in ASCII)
- 4 segment LED display for debug purposes
- 10 bit GPIO header for debug purposes
- Selectable voltage source either USB or LPC bus
- 4 jumper selectable boot images for LPC
- Third party free USB drivers for
  - Windows (98,2000,ME,XP)
  - Linux 2.4 and greater

Included:

- LPC Dongle
- Python based software with source (Linux and Windows supported)
- USB cable
- LPC cable

Index

Dongle version

Dongle version number can be seen on the postcode byte if reset button is held down while USB cable is connected or dongle is connected to a host system by LPC bus. This document deals with dongle version 5.

Additions from version 3 to 4 were:

- LPC Firmware Hub read capability and disable jumper to disable the new mode.
- Post code peek mode (post code IO writes are shown but read requests are ignored)

Additions from version 4 to 5 are:

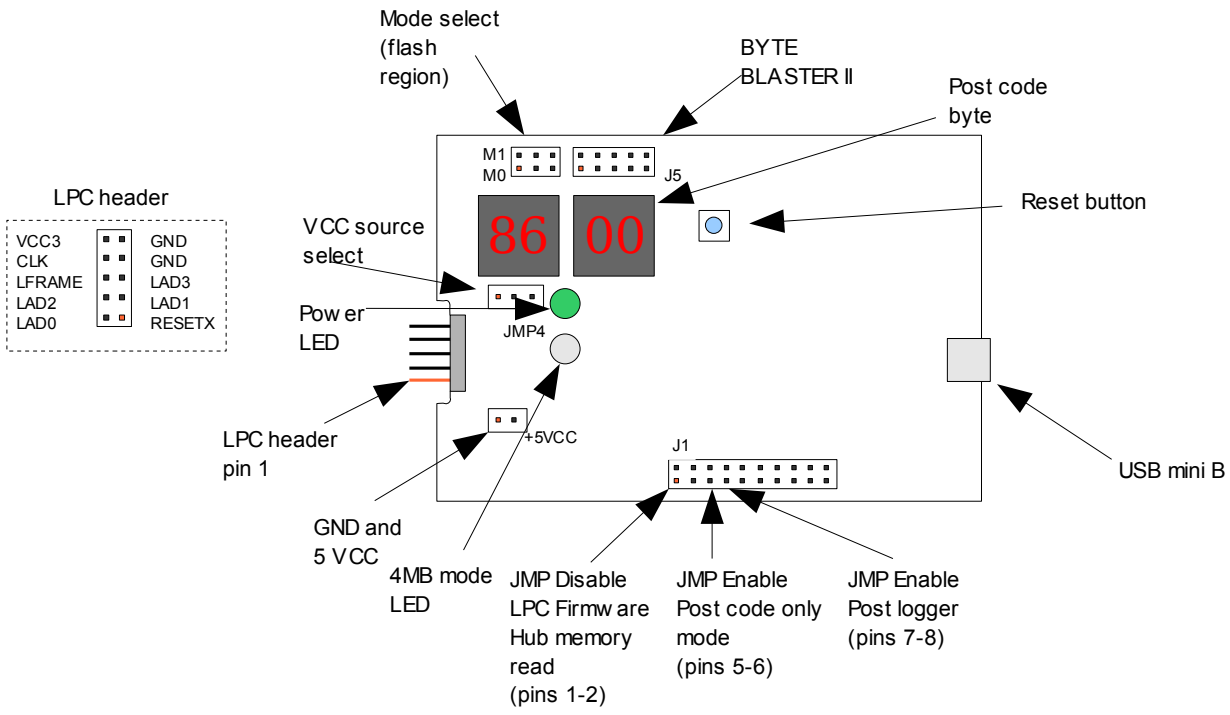
- Changed concurrent access controller to flash to stop LPC cycles interrupting flash programming
- Post code logger mode and jumper to enable the new mode (the jumper at the same time disables USB programming interface)
- Added fast block read hardware (block size is 64K bytes) and improved write hardware implementation to speed-up write process with following approximate results:

  (Gentoo Linux 4096K write time 59s includes erase time 27s, read time 10s)

  (Debian Linux 4096K write time 59s includes erase time 27s, read time 34s)

  (Windows XP 4096K write time 2m37s includes erase time 27s, read time 7s)

LPC dongle interfaces



Drawing 1: Board top layout

LPC

LPC header provides 7 IO pins used for LPC signal (see Dongle schematic). By default 1MB on flash can be accessed over LPC bus (highest MB of 4 MB available in each mode configuration). This can be changed by IO command to full 4 MB.

LPC IO write to address 0x0088 data 0xF4 turn on 4MB addressing

LPC IO write to address 0x0088 data 0xF1 turn off 4MB addressing

LPC IO write to address 0x0080 will be displayed on Post code byte

Mode Select

Mode select jumpers are intended for internal image selection from dongle flash divided into 4 separate 4 MB sections.

Jumper settings on GPIO header

On GPIO header pin pairs 1-2, 5-6 and 7-8 are used as jumper pins as shown on Drawing 1. Jumper on GPIO header J1 pin pair:

- 1-2 is “Enable LPC Memory read” this is legacy option to allow booting from devices that don't support LPC Firmware Hub read cycles although LPC Firmware read cycles are also responded in this configuration. If pins 1-2 are left open only LPC Firmware read cycles are responded.
- 5-6 is Post code peek mode enable jumper. In this mode dongle is not a memory device and accepts only IO write commands to addresses 0x0080 (post code) and 0x0088 (memory size command register) the post code bytes are displayed on lower two segments of 4 segment display.
- 7-8 is Post code logger mode enable jumper. In this mode all post codes are sent to USB serial port. Open serial terminal to the FTDI USB serial port with settings: baud=115200, data=8 bit, parity=none, stop=1 bit, Flow=none. Post codes are displayed in hexadecimal format 16 bytes per line (\r\n as line separators) all bytes are prefixed by “x” and byte separator is space character. For example:

  “x80 x11 x55 x10 x80 x11 x55 x10 x80 x11 x55 x10 x80 x11 x55 x10\r\n”

  The 4 segment display in this mode will show internal FIFO byte count. If the byte count reaches “x1F40” the dongle will start to flow control LPC IO write cycles to avoid data loss. This may affect boot time if post code IO writes are heavily used (no data loss should occur).

GPIO pins are platform pins that are not used in the standard LPC version of the dongle but are target for tailored development for specific functions.

VCC source select

Jumper is used to define VCC source. When USB cable is used at the same time with target board LPC connection, JMP4 should be off. If jumper is on pins 1,2 (marked with thick white line under the pins) the VCC3V is taken from target system, but USB cable must be disconnected.

When target board is used as voltage source the POR (power on reset) time along with the FPGA configuration time has to be taken into account when using the dongle as a boot device. It is recommended to use target board reset to initiate clean boot after power up.

USB

USB interface is used to program dongle flash or read back flash content. This can be done by Artec USB Dongle programming utility.

Usage:

Write file	dongle.py [-vq] -c <name> <file> <offset>
Readback file	dongle.py [-vq] -c <name> [-vq] -r <offset> <length> <file>

Options:

<file> <offset>	When file and offset are given file will be written to dongle
file:	File name to be written to dongle
offset:	Specifies data writing starting point in bytes to 4M window
	For ThinCan boot code the offset = 4M - filesize. To write
	256K file the offset must be 3840K
-c <name>	Indicate port name where the USB Serial Device is
name:	COM port name in Windows or Linux Examples: COM3,/dev/ttyS3
	See Device Manager in windows for USB Serial Port number
-v	Enable verbose mode. Displays more progress information
-q	Perform flash query to see if dongle flash is responding
-r <offset> <length> <file>	Readback data. Available window size is 4MB
offset:	Offset byte addres inside 4MB window. Example: 1M
	use M for MegaBytes, K for KiloBytes, none for bytes
	use 0x prefix to indicate hexadecimal number format.
length:	Amount in bytes to read starting from offset. Example: 1M
	use M for MegaBytes, K for KiloBytes, none for bytes
file:	Filename where data will be written
-e	Erase device. Erases Full 4 MegaBytes

Board test options:

-t	Marching one and zero test. Device must be empty
	To test dongle erase the flash with command -e
	Enables dongle memory tests to be executed by user
-b	Leave flash blank after test. Used with option -t

Examples:

```
dongle.py  -c COM3 loader.bin 0
dongle.py  -c /dev/ttyS3 boot.bin 3840K
dongle.py  -c COM3 -r 0x3C0000 256K flashcontent.bin
```

USB programming utility can address all of the 4 MB available in any mode that is selected. It must be remembered that only 3072KB to 4096 KB in each mode configuration is directly accessible over LPC and is repeated in LPC memory address space (x86 legacy 1M window) without using the 0xF4 command first (see LPC paragraph).

After programming press reset button on dongle board to unlock flash interface for LPC reads.

FAQ

Q: Why does the dongle give error code when opening serial port in Ubuntu Linux?

A: If you use Ubuntu, make sure to uninstall brltty (apt-get remove brltty --purge) before you hook up the dongle; otherwise it will hijack the dongle and you won't be able to talk to it. Brltty is a software Braille terminal. (http://www.coreboot.org/Artecgroup_programmable_LPC_dongle)

Q: Why my dongle does not boot immediately after programming.

A: When dongle is connected to USB and LPC interfaces at the same time LPC interface will be locked when USB is used to program on-board flash. This stops LPC interface from interfering with flash programming. To indicate that programming is finished press reset button on dongle board.

Source code

Source code is available at
http://www.opencores.org/projects.cgi/web/usb_dongle_fpga/overview

Contains:

- Python USB application source code
- VHDL LPC slave (supporting IO write and Memory read)
- VHDL Flash Waveform generator
- VHDL FTDI parallel interface to on-board flash (supports block write)
- VHDL Scanning LED segment display coder
- Project files with pin location constraints (for Altera Quartus™ Web Edition)