

Commissioning a Balloon 3 Board

D.L.Bisset

iTechnic Ltd

Friday, 28 July 2006

Version 0.1 : Preliminary Draft 11/07/2006

Version 0.2 : Corrections and addition of software details. 27/07/2006

Known issues with the current release of hardware are documented as follows:

Note there is an issue with....

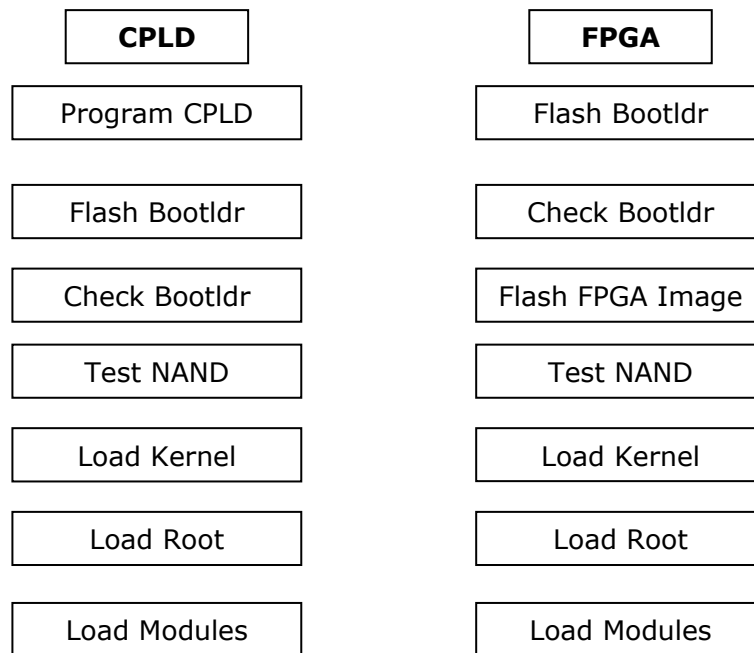
Commissioning a Balloon 3 Board

The purpose of this document is to bring together all of the different information required to commission a new Balloon 3 board without any previous configuration.

This document will give an overview of the process and the appendices will describe the cable sets needed for the different connections to the board.

If you have a partially configured board it is possible to start mid way through this document in accordance with the level of configuration your board has.

The sequence of configuration is different between the CPLD and FPGA versions of the board. The main steps in the configuration of both types of Balloon board are as follows:



In order to carry out these steps you will need a means of connecting to the JTAG FFC port on the Balloon Board Appendix A gives details of how to do this, as well as various software tools which are detailed in Appendix B.

The software that is needed for this process can be found on husaberg.toby-churchill.com in the following directory tree:

```
/balloon/releases/development/balloon3/distro/test-0v1/
```

In the remaining documentation this directory path will simply be referred to as <dist>

This tree will contain the released test versions of the software suitable for use on the different prototype builds. A README file indicates which test build relates to which physical build of hardware. The following items are stored in this tree:

- a) `bootldr` suitable for Balloon 3.
- b) FPGA/CPLD images and VHDL release code.
- c) `bflash` utility for initial programming of the NOR

- d) Kernel image.
- e) Root tree package
- f) Additional `ipkg` modules that are specific to Balloon3.

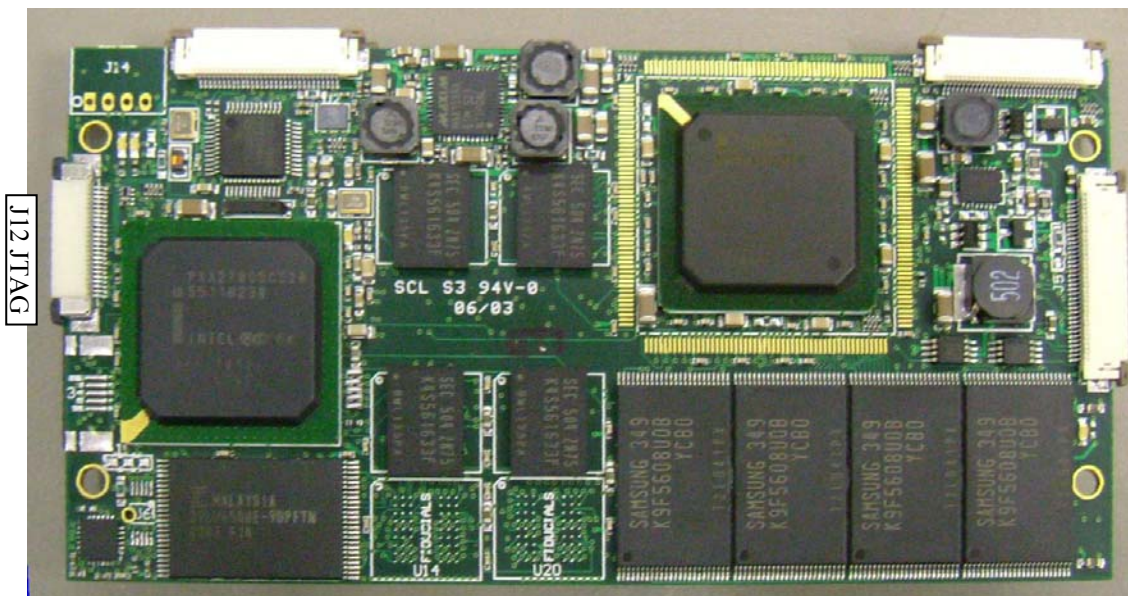
The following sections detail each of the configuration steps shown above.

Hardware Setup

Both JTAG and RS232 connections are required to fully configure the Balloon Board.

The Balloon board needs to be powered from between 3.6v to 5v with a current capability of at least 1A. This is best done through the power connector but for configuration can be done through the JTAG port.

It is not advisable to use a battery to supply the Balloon board during the setup process unless there is an external adjustable current limit in series.



On first power-up it is advisable to limit the current to 100mA or less from the PSU in case the board has a fault, after the first application of power has succeeded this limit will need to be raised to 600mA.

The CPLD and the initial testing of the board are all carried out via JTAG interfaces which are available on the board. There are two separate JTAG chains one for the PXA processor and the other for the FPGA/CPLD. Each of these chains is brought out to the JTAG connector together with other useful signals such as the console RS232 and the USB Client interface.

Many of these signals are also available on probe pads on the bottom of the board so that the configuration process described in this document can also be carried out using a “bed of nails” This means that boards without a JTAG connector can still be programmed.

Appendix A details the connections from these connectors to the different types of JTAG pod that can be used, and to the console RS232 connection which will also be needed.

Programming the CPLD

Programming the CPLD is done via the CPLD JTAG chain and either the Windows or Linux based Xilinx WebPack software (or equivalent) alternatively the Linux tool `playsvx` can be used.

Connect a JTAG programming pod to the CPLD JTAG chain and program the device using the `l3cpld.jed` file that can be found in:

```
<dist>/vhdl/cpld/images.
```

Flash the Bootldr

The `Bootldr` software needs to be loaded into the NOR Flash on the Balloon board via the JTAG port to the PXA. This uses the boundary scan on the PXA and so can take a considerable amount of time. Once the `Bootldr` is loaded and running it can load new versions of itself via the serial port which much faster.

Programming the `Bootldr` for the first time requires either XJTAG or the Linux based `bflash` utility.

Connect the JTAG programming pod to the PXA chain.

The `bflash` command line to program the boot loader for the FPGA version is:

```
bflash Balloon3 -p bootldrpxa-fpga.fast
```

and for the CPLD version:

```
bflash Balloon3 -p bootldrpxa-cpld.fast
```

Note the separation into FPGA and CPLD boot loaders is temporary as a lack of P1 prototypes has made it impossible to test a combined boot loader on a realistic timescale. These will be fixed once P2 machines become available for further testing.

These can be found at:

```
<dist>/bootldr
```

Note that this is **not** a `gzip` file.

This should print out various information as it processes the command, firstly information about the PXA version then a dump of the CFI data in the NOR device, and finally a running commentary on the erasure and programming of the NOR.

If this fails at any point the error messages should provide enough information to identify the problem.

Bear in mind that if this is the first time a device has been powered up it is possible that there may be undetected hardware faults that cause this process to fail. The programming of the NOR uses a significant fraction of the functionality of the Balloon board Bus and even probes the CPLD on CPLD boards.

Further debugging may be carried out using the `bflash` program to read memory and set GPIO lines.

NOTE as of the initial release of bflash there are a number of known issues. The fast option (-f) should only be used with -p and cannot be used in conjunction with -g. Some functions displayed by the (-h) are not implemented.

Checking the Bootldr

There is currently nothing that needs to be configured in the `Bootldr`. In the future it may be necessary to set the board type and save the parameter file.

It is worth checking the operation of the `Bootldr` at this point.

Reset or power cycle the Balloon board. Make sure you have a working RS232 connection to the console (Most perceived failures at this stage stem from not having a working terminal program connected to the right serial port with the right baud rate setting which should be 115200 baud 8 bits with no hardware flow control and no modem parameters set).

Once the board starts up a stream of data should be visible on the terminal (the exact details of this will depend on the `Bootldr` version loaded, but in early versions of `Bootldr` there are a few screens of low level debug info).

If all is well this should result in the `boot>` prompt appearing.

At this point things differ between the FPGA and CPLD versions.

If you have an FPGA version you now need to load the default FPGA image into the NOR so that the `bootldr` can program the FPGA as it starts. For CPLD versions you should be able to test the NAND memory.

Flash the FPGA Image

The FPGA does not retain its configuration during a power cycle and so must be reprogrammed each time the Balloon board starts up. Since NAND is not available until after the FPGA is programmed the NOR must contain a default image for the FPGA to allow `Bootldr` to access NAND memory and other functions.

This image is currently held at a fixed address in NOR. The `Bootldr` can recognize if the image is absent and can recognize the bit sense of the image (different compilers for the FPGA will produce different bit orders per byte in the images).

The image can be loaded using `bootldr` and the serial port via `xmodem` through your terminal emulator.

The command for loading the image is:

```
Load flash 0x200000
```

Followed by the usual sequence to activate an `xmodem` download from whatever terminal program you are using.

The file to download should be `l3fpga.bin` (or `l3fpga.bit`).

This can be found in:

```
<dist>/vhdl/fpga/Images
```

Under certain circumstances (for example when debugging the FPGA loading code in the `Bootldr`) it may be necessary to use `bflash` to load the FPGA image. In this case the command:

```
bflash Balloon3 -p L3fpga.bin 0x200000
```

Will load the image directly to NOR via the PXA JTAG chain, as usual this takes a long time and is *not* the normal route for loading new default FPGA images.

Once the FPGA image is loaded it is important to restart the `bootldr` as the FPGA image is only loaded into the FPGA at initialization of the `bootldr`.

This can be done either by resetting or power cycling the Balloon board or by using the command:

```
boot boot.
```

There should be a slight pause with the message

```
Programming Device
```

As the FPGA loads.

The FPGA may fail to load in which case a timeout message will be displayed. This should either say that the timeout was on `init` or `done`. In either case there is a problem with programming or finding the FPGA.

If the output on the console stops at this point or a `DABT` message occurs or equivalent failure then the FPGA has a bad image in it and has now crashed the processor bus, it is also likely that the current consumption of the board has risen.

This can be caused by a number of things:

- The PSU may have dipped during programming or at startup when the current drawn by the FPGA is greatest. Power cycle the board and observe the voltage delivered by the PSU or any current limit indicators on the PSU, if the voltage dips or the current limits then either raise the current limit or get a better PSU.
- The FPGA image may not have programmed fully into the NOR, re-flash the FPGA image and try again carefully checking the console messages.
- There is a board fault that has been exposed by activation of the FPGA. The FPGA will drive some parts of the system that will not have been tested using the XJTAG tool (used post manufacture to verify the board) and so it is possible that faults on these parts will not be seen until the FPGA is programmed.

Test NAND

The NAND sits on the other side of the CPLD/FPGA to the processor and so cannot be accessed until the FPGA or CPLD have been correctly programmed.

Once you are successfully running `bootldr` with the FPGA or CPLD then it is possible to test the NAND. Simply using the `yaffs` commands built into the `Bootldr` is sufficient at this stage.

```
yaffs ls
```

Should produce the following output.

```
drw-rw-rw- 0      0      512  lost+found
```

Load Kernel

The kernel is loaded using the following bootldr command and xmodem from the terminal emulator

```
yaffs write zImage
```

The zImage file should be found in the file tree. During development this is at:

```
<dev>/kernel/zImage
```

To check it has loaded use:

```
yaffs ls
```

This should show that the zImage file is now loaded into the filesystem.

NOTE

Now that the kernel is loaded power cycling and rebooting the system will automatically try to boot the kernel. To get back to the boot> prompt you need to press the space bar a few times as the bootldr starts up.

Load Root

Before the system can be booted for the first time the root filing system needs to be loaded. This is done using the following bootldr command and xmodem and will take while to complete.

```
load root
```

Followed by the appropriate sequence of xmodem download commands from your terminal emulator.

During development the gzip'd root file can be found at:

```
<dev>/rootfs/balloon3-test.yaffs.gz
```

Once the root is loaded the first boot of the system can be attempted with :

```
boot.
```

Since this is the first real test of the whole processor and bus system of the machine there is always the possibility that this will fail.

Load Modules

Before features such as the PCMCIA can be activated it may be necessary to load additional modules into the root tree. This must be done by downloading the .ipg files from:

<dev>/feed/arv41/

and using the `ipkg` command to install them

Note that by now `zmodem` may be active and files can be downloaded using this.

The exact list of packages required at this stage will depend on what is already loaded into the root image and what sub systems are built into the Balloon board you have. It is likely in the early stages of release that you will need to make sure that the following are installed:

- PCMCIA
- wireless-tools
- udev

A README file will be held in this directory to help you decide which, if any, module packages you need to download.

Note there are known issues with using `cardctl eject/insert` with PCMCIA cards. It is currently recommended that you insert cards and then power up the system.

Final Note

Once you have completed the configuration in this document you should complete the testing of the board by following the instructions in the document "Testing Balloon 3".

Appendix A

This Appendix details the hardware that is needed to support the configuration of a Balloon3 board that has not been programmed.

Hardware is needed to provide a connection between a PC (running Linux or Windows) and the Balloon board. Connection to the JTAG chains on the Balloon board and the RS232 console port are needed.

There are a number of different JTAG connection solutions each of which is detailed as follows.

XJTAG Connection

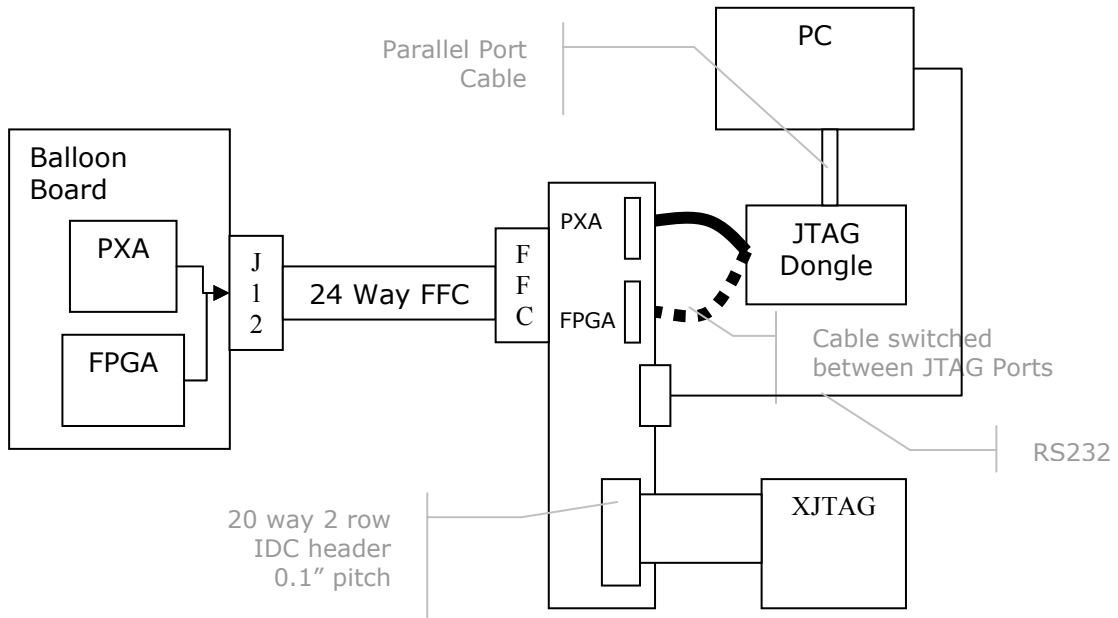
The XJTAG pod uses a standard 0.1" pitch double row IDC connector to deliver the JTAG signals to the target board. This must be connected to the JTAG FFC connector on the Balloon board in such a way that the two JTAG chains on the Board can be accessed independently as well as being linked together.

The pin-out from J12 the JTAG connector on the Balloon 3 board is as follows:

Pin	Signal	Group	Flow
1	GND		
2	NTRST	PXA JTAG	
3	TMS		To Board
4	TDO		From Board
5	TDI		To Board
6	TCLK		To Board
7	GND		
8	GND		
9	TCLK	CPLD/FPGA JTAG	To Board
10	TDI		To Board
11	TDO		From Board
12	TMS		To Board
13	PORT_EN		To Board
14	VDD	3v3	From Board
15	TXD	Console RS232	From Board
16	RXD		To Board
17	GND		
18	UDC N	Client USB (PXA)	Bidirectional
19	UDC P		Bidirectional
20	RESET		To Board

21	GND		
22	GND		
23	VDD_RAW		To Board
24	VDD_RAW		To Board

The system connections are as follows:



There are a number of different ways to terminate the different connections.

The XJTAG expects a 0.1" 20 way double row (10 x 2) IDC header which must then be connected via jumpers to the two JTAG chains.

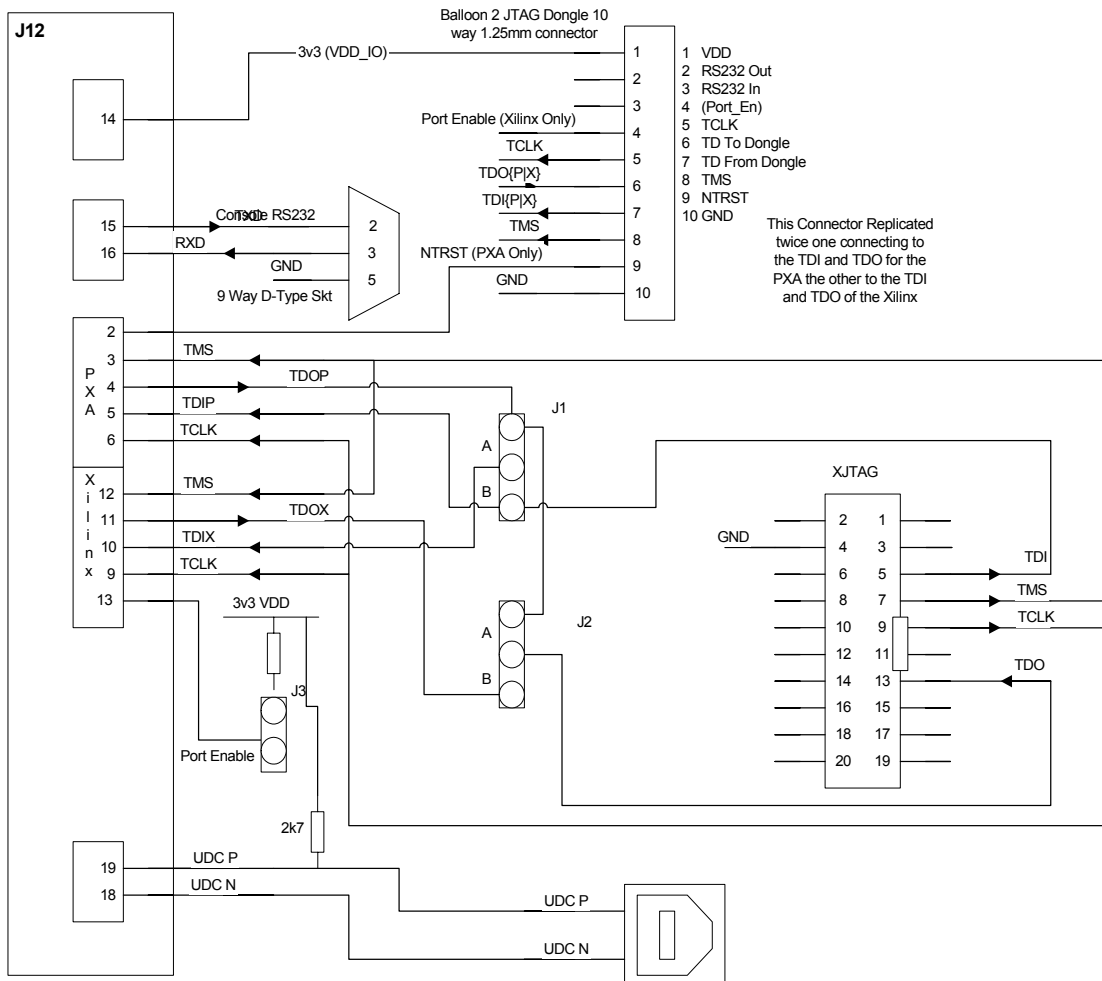
If a standard Balloon 2 JTAG parallel port JTAG dongle is used to provide JTAG output from the PC then this terminates in a 10 way 1.25mm pitch Molex header.

A number of different means can be used to connect this header to the FFC connector from providing headers on the breakout board to flying wires.

The following diagram shows the connections required for connection of the XJTAG header to the J12 FFC connector. This requires two Jumpers to allow the two JTAG chains to be linked or used separately.

The headers for the Balloon 2 JTAG dongle must be wired separately to two headers.

If all three JTAG connections are wired to the same breakout board then they must only be used one at a time and the unused connectors must be unplugged.



The jumpers allow the chains to be joined or accessed independently by the XJTAG driver.

The following table lists the link positions for particular configurations:

J1 Position	J2 Position	Function
No Jumper	A	Access PXA Chain Only
B	B	Access Xilinx Chain Only
A	B	Link Both chains PXA then Xilinx

Appendix B

Bootldr FPGA Images

The default FPGA image is stored in NOR. Currently this is at physical address 0x200000. However it is intended that multiple images be stored and that a mechanism for selecting the boot image is created using the `params` mechanism.

It is also possible that once a default image is loaded and the NAND can be accessed an image stored in the root file tree can also be used to program the FPGA prior to Linux being booted.

Bflash Command

The `bflash` command has been specifically written to accommodate the JTAG programming and testing of Balloon boards it is compatible with both Balloon 2 and Balloon3 and provides a limited set of debug functions that can be accessed from its command line. In particular memory addresses can be read or written and GPIO lines set prior to executing a Flash load.

Note the initial version of `bflash` provided has not been fully tested with Balloon2.