S514C Product Update - Programming The B016's Flash-Roms

Bob Green
Updated by Nick Rees for JAC setup - 26 Feb 2002

February 27, 2002

1 Introduction

The IMS B016 contains 64K Bytes of software programmable *read only* memory (ROM). The ROM is organised in four banks, each one being one byte wide. B016s shipped by inmos have already had these ROMs programmed. This is necessary for the correct operation of the B016 device driver supplied as part of the IMS S514C product. It is a simple job for someone using the inmos 'C' or Occam toolsets to write their own program to go in the ROMs.

2 Hardware Requirements

Before the user can program the B016's ROMs, he must have another transputer connected to a host (VxWorks, PC, Sun or Vax). This root transputer is then connected to the B016 via a link and subsystem. Figure 2 shows which links should be connected when the system is setup at the JAC.
The MSC board is the same one used for all the UKIRT VME systems, but is modified with the following:

1. The transputer in slot 0 must be one of the 32 bit ALICE or SCUBA trams, not the 16 bit ones used by the AstroCam systems.

2. The MSC board must be wired so that link 2 of slot 0 is connected to link 4 of the P2(X2) connector. At least on of the MSC boards is wired this way and it is labled "UKIRT MSC TRAM board with B016 programming link".

3. Jumper JP14 should be removed (it is normally installed).

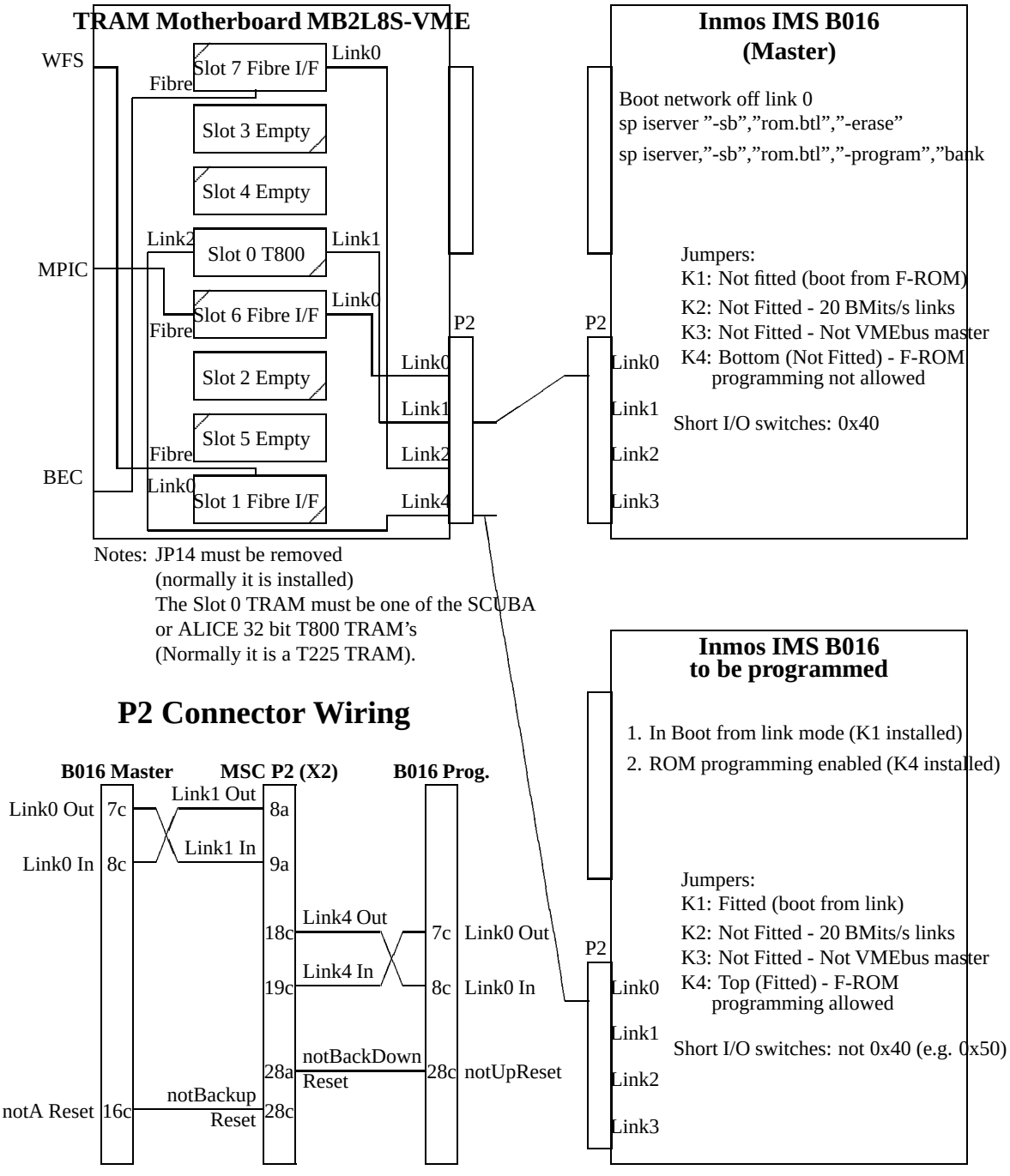
There are also some jumpers on the B016 which must be set correctly for the ROMs to be programmed.

1. The B016 must be in *Boot from link* mode (K1 installed).

2. The B016 must have ROM programming enabled (K4 installed).

3. The short I/O addresses must not conflict with other borads in the crate (at the JAC the master board is set to 0x4000, and so the slave board that is to be programmed should be set to something else - say 0x5000).

B016-MSC Motherboard Configuration for Reprogramming B016's



P2 Connector Wiring

3 Programming the ROM image using the JAC VxWorks systems

The ROM image source code is in `common/acam/b016/occam/b016drv`. However, I haven't been able to build it properly, so the original, as-built binaries are in `common/acam/b016/occam/startup`, and are called `bank0`, `bank1`, `bank2` and `bank3`. (There are four files, since there are four flash banks. These files are copied into the `data` install directory as part of the make process. Also in the startup directory, there is a VxWorks startup script that will program the ROM's of a b016 as part of the boot process. It uses the rom.btl program described in the next few sections. The source code for rom.btl is found in `common/acam/b016/occam/b016r` and it is normally built after checking out the code from CVS. The only changes I have made is to adapt them for the later versions of the INMOS compiler, and to change the link numbers and transputer types in the `rom.pgm` file.
The next few sections are as they were in the original document, and describe the rom.btl program in greater detail.

4 Generating the ROM image

Compile your code in the normal way. If object size is important, it may be sensible to disable symbolic debug information from being generated. Add the following options to the `iconf` command

`-ra -rs 64k -p` *board*

where *board* is the name of the B016 as specified in the configuration (`.cfs`) file. Add the following options to the `icollect` command

`-ra -rs 64k`

Finally, the `ieprom` tool is used to generate four files, one for each bank of ROM data. `ieprom` can be configured to generate several different types of output. A sample configuration file is given below:

```
root.processor.type     t8
eprom.space             20000

bootable.file           drv.btl

output.format           binary
```

```
output.block            bank0
  start.offset          00000
  end.offset            1FFFF
  byte.select           0
  output.address        00000

output.block            bank1
  start.offset          00000
  end.offset            1FFFF
  byte.select           1
  output.address        00000

output.block            bank2
  start.offset          00000
  end.offset            1FFFF
  byte.select           2
  output.address        00000

output.block            bank3
  start.offset          00000
  end.offset            1FFFF
  byte.select           3
  output.address        00000
```

This example takes the file `drv.btl` and generates four files: `bank0`, `bank1`, `bank2` and `bank3`.
Refer to the User Manual supplied with your ‘C’ or occam toolset for full details of the process described above.

5 Using the ROM tool

Now that we have successfully generated the ROM data, we need to get it into the B016. A tool is provided for this purpose, called `rom`. This tool is supplied as a transputer binary called `rom.btl`. It can be run in the following manner
`iserver -sb` */s514c root*`/bin/transputer/rom.btl [` *rom args* `]`

Where *s514c root* is the name of the directory where the S514C was loaded.
With no arguments supplied, a brief help screen will be displayed.
The options that the rom tool understands are described below

`help`	The help screen is displayed.
`dump` *file-root*	This option dumps the contents of the ROMs into files. One file is created for each bank, called *file-root*0, *file-root*1, *file-root*2 and *file-root*3. Files created with this option can be used with the `program` option to program other B016s.
`erase`	This option causes the ROMs to be erased.
`peek` *address*	This option reads the word stored at *address* and prints it on standard output.
`program` *file-root*	This option programs the ROMs. Four files are opened, one for each bank, called *file-root*0, *file-root*1, *file-root*2 and *file-root*3. The data from these files is programmed into the ROMs. The ROMs must have been erased before using this option.
`test`	This option performs a very simple confidence test. It attempts to program one byte in each of the four banks of flash-roms. Any errors are reported. This is useful in detecting general failures in a particular bank. The roms must be erased before using this option.

So, if we wished to program the ROMs with the data files generated in our earlier example we would type

```
iserver -sb rom.btl -erase
iserver -sb rom.btl -program bank
```

Any errors will be reported.
When the ROMs have been programmed, put the B016 into *Boot from ROM* mode (remove K1) and reset the board.