README.mvme5500

T. Straumann, 2005/4/26, 2005/10/14

All instructions below also apply to the MVME6100 combined (AKA 'beatnik') BSP
(i.e., the beatnik BSP supports both, the 6100 and the 5500 boards).
The same 'netboot' binary supports both boards and can be found
at '<top>/target/ssrlApps/powerpc-rtems/beatnik/img/netboot.flashimg.bin'.

The 'netboot' loader is now also available for the MVME5500 board.
A patch to the mvme5500 BSP (but not the mvme6100 combo BSP) is necessary
so that 'netboot' can pass config parameters to the bootee.
The necessary patch is applied in the
'package/rtems/4.6.2-devel' area.
Furthermore, more patches are needed to fix bugs which prevent
this BSP (i.e., the mvme5500 BSP) from being booted properly
(by 'chance' it works with MotLoad but not e.g., with motLoad's
'execProgram').

The general description from README.vgmflash still applies
EXCEPT for the sections about burning/programming the flash
and starting the netboot program.

The respective procedures for MVME5500/MVME6100 are as follows:

I) Programming the Flash.

```
a) Image File Creation
- - - - - - - - - - - -
```
The image file 'netboot.flashimg.bin' is created as
a result of the automatic build process.

```
b) Burning
- - - - - -
b1) Burning from RTEMS (cexp prompt)
- - - - - - - - - - - - -

An old version of netboot can easily be overwritten
using the
'BSP_flashWriteFile(int bank, unsigned offset, char *path)'
command:
- make sure image file is accessible from NFS server on
the target board.
- issue

Cexp> BSP_flashWriteFile(0, 0, "<path_to_netboot.flashimg.bin>")

b2) TFTP download and burning from MotLoad
- - - - - - - - - - - - - -

Downloading the flash image and burning it is done
at the MotLoad command line prompt:

tftpGet -c<client> -s<server> <other parms> -f<path>/netboot.flashimg.bin

write down the load address and image size which must
be converted to hex. Note that the image size is also
required by the 'startup script', see below.

Now the flash is ready to be programmed - make sure
the target area is not used by other applications etc.
```

flashProgram -s<load address> -n<image size> -v

This burns the image to the start of flash0.

```
After burning the flash, the startup command must
be checked for a change in image size since this
information is needed for the 'bmw' block copy command.

BTW: Eric Norum wisely recommends to write-protect the
MotLoad flash by removing the respective jumper
(I believe it is J9, RTM). This is board-dependent,
i.e., the 6100 and 5500 are different - RTM.

b3) Copying from master over VME backplane and burning from MotLoad
- - - - - - - - - - - - - - - - - - - - - - - -
```

This method is particularly useful if a number of boards has
to be programmed:

Install a 'master' (a board with netboot in its flash) and
any number of 'slaves' (boards with blank flash) into a VME
crate.

Boot RTEMS on the master. At the prompt, issue the commands

```
Cexp> BSP_VMEInboundPortCfg(0,0xd,0x01000000,0,0x20000000)
```

This makes the master RAM visible on the VME bus (flash is
not on PCI and can not directly be mapped to VME).

copy to buffer

```
Cexp> buf = malloc(0x200000)
0x009cd478 (10278008)
Cexp> memcpy(buf, 0xf2000000, 0x100000)
0x009cd478 (10278008)
```

(replace the 0xf2000000 flash address by 0xf4000000 on
a MVME6100 master board)

calculate VME address for slave

```
Cexp> buf + 0x91000000
0x919cd478 (-1851992968)
```

write down this address.

On each slave at the MotLoad prompt issue (no leading '0x' for
hex numbers)

```
MVME6100> mdw -a<translated_buf_addr>
919CD478 48000005 7C000278 7C000124 7CA802A6 H...|..x|..$|...
919CD488 38A5FFFC 3F60000B 637BE7E0 7C661B78 8...?`..c{..|f.x
919CD498 7E232051 40810034 3E40000D 7E05DA14 # Q@..4>@..···
919CD4A8 7C109000 40800008 7E509378 7C101800 |...@...~P.x|...
```

to verify successful mapping. You should see non-FFFFFFFF
hex data, starting with 48000005. If this test succeeds
then you are ready for burning:

MVME6100> flashProgram -s<translated_buf_addr> -n100000 -v

Repeat this last step on all slaves.

Don't forget to program a 'mot-script-boot'.

II) Starting 'netboot'

It is recommended to set the 'mot-scipt-boot' environment
variable in order to automatically start netboot:

```
gevEdit mot-script-boot

netShut
bmw -af2000000 -b<f2000000 + image size> -c04000000
rs -nr3 -d0
rs -nr4 -d0
go -a04000000

++++++++++++++++=
```

NOTE: on the mvme6100, the flash0 is at f4000000,
hence the 'bmw' command needs to be adapted.

```
++++++++++++++++=
```

This chooses the start of the flash0 area. The image size
must match the real image size (or exceed it). The target
memory area (-c) must be available.

Note that the 'netShut' command is important: Modern
interface devices usually feature DMA busmasters and
such a busmaster - if not stopped prior to starting RTEMS -
may corrupt a memory area that was designated as buffer
space by MotLoad but has been taken over by the newly
started application.

III) NOTES

All comments about mvme5500 BSP bugs do not apply to the mvme6100 BSP
(which also supports the mvme5500 board).

MVME5500 BSP bug: A bug in the BSP causes applications loaded
by 'netboot' instead of 'motLoad' to hang/freeze
very early during the boot process [of the application,
not netboot's].
The bug has been fixed in 'package/rtems/4.6.2-devel'
area but if you suspect you experience this problem
you should contact me.

MotLoad's 'execProgram' and 'netBoot':
Although these commands could theoretically be
used to boot RTEMS/mvme5500 this is not recommended.
These commands disable the caches. The BSP,
however was designed to inherit MotLoad's cache
settings and a very serious performance hit
is the consequence of using 'execProgram/netBoot'.

NVRAM usage: 'netboot' reserves/uses the first 4k block
of the NVRAM to store parameters. If this
is to be changed, the setting in 'nvram.c'
can be changed.
NOTE: 'nvram.c' is also used by GeSys to
allow for reading/modifying the NVRAM from
the running GeSys prompt – you must make
sure, GeSys and netboot share the same
version of 'nvram.c'!