Quick Start Guide

Embedded Performance, Inc.
606 Valley Way, Milpitas, CA 95035
Telephone: (408) 957-0350
FAX: (408) 957-0307
e-mail: support@epitools.com
web: www.epitools.com

Quick Start Guide for Using MAJIC with the X-Hyper250B Evaluation Board

June 26, 2003

Introduction

Thank you for choosing MAJIC as part of your development environment. This quick-start guide will cover the specifics regarding connecting MAJIC to the Hybus X-Hyper250B evaluation Board. It will also cover the process of rebuilding the Linux Kernel to include debug symbols, and the procedure for debugging the Linux Kernel. Using MAJIC to debug your Linux kernel will eliminate the need to rely on a debug agent running on the target. This means that you will be able to debug a kernel at the very early stages of development. You will have full control of the processor via hardware.

MAJIC Configuration Basics

Please refer to the *MAJIC Quick Start Guide – Hardware* for information on connecting the MAJIC to your target and host computer. For this application note it is assumed that the host computer, MAJIC probe, and target board are connected via Ethernet with the same subnet mask. Also, the host computer must be connected to the target board with a serial cable.

If you are using a Windows version of the EPI Development Tools (EDT) then use the *MAJIC Quick Start Guide – Software* for information on creating configuration files for GDB; otherwise see the *Getting Started with MAJIC and Linux* application note. For information on setting up the MDI-Server connection please see the *gdb-readme.txt* file in your manuals directory.

Building the Kernel Image

For this guide it is assumed that the X-Hyper250 tools are installed in the /opt/xhyper250b directory, and the EDT package is installed in the /opt/edta directory.

In order to do source level debugging of the kernel image it is required that the kernel be rebuilt and re-programmed into the target. By doing this it will be ensured that the debug information in the *vmlinux* file will match the *zImage* file. Please reference page 39 in the *X-Hyper250 Manual* for official instructions on rebuilding the kernel, the commands here are only for a quick reference.

Change directories to the location of the Linux source:

```
[root@mc /] cd /opt/xhyper250/kernel/linux-2.4.18-rmk7-pxa1-xhyper250R1.1
```

Then run menuconfig to configure the build for the Linux kernel.

```
[root@mc linux-2.4.18-rmk7-pxa1-xhyper250R1.1] make menuconfig
```

Menuconfig has been shipped by Hybus to default to all correct settings for the X-Hyper250B evaluation board. In order to debug the Linux kernel with the MAJIC though, one setting will need to be changed. Under `kernel hacking` change `Include debugging information in kernel binary` to be enabled. This will cause the *vmlinux* file to be compiled with the –g switch, and therefore have source and symbolic debug information. Then next step is to build the image:

```
[root@mc linux-2.4.18-rmk7-pxa1-xhyper250R1.1] make dep
[root@mc linux-2.4.18-rmk7-pxa1-xhyper250R1.1] make zImage
```

If the compilation works correctly the *vmlinux* file will show up in the current directory, and the *zImage* file will show up in the `/arch/arm/boot` directory, off of the current working directory.

After successfully recompiling the kernel the *zImage* file should be copied to the `/tftpboot` directory in order for TFTP to find it during the re-programming stage.

Programming the Kernel Image

Before reprogramming the Linux kernel ensure that TFTP and BOOTP have been installed and configured on the host computer. Instructions for this can be found in the X-Hyper250B manual on page 68. The original instructions for reprogramming the Linux kernel can be found on page 22.

Connect to the target with Minicom, then power cycle the board so that it will reboot. Instructions for configuring Minicom can be found in the X-Hyper250B manual on page 12.

```
[root@mc /] minicom

X-HYPER250>
--------------------------------------------------------------------------------
X-HYPER250-R1
Copyright (C) 2002 Hybus Co,. ltd.
X-HYPER250 comes with ABSOLUTELY NO WARRANTY; Read the GNU GPL for
details.     This is free software, and you are welcome to redistribute it
under certain conditions; read the GNU GPL for details.
Support: http://www.hybus.net
--------------------------------------------------------------------------------
Autoboot in progress, press any key to stop ...
```

At this point hit any key so that Autoboot will be aborted. You can then type in the commands to have the kernel flashed to the target.

```
Autoboot aborted
Type "help" to get a list of commands
X-HYPER250> tftp zImage
No IP. Bootp start...
Our Ethernet address is 00D0 CAD1 2577.
Sending bootp packet...                                                .
Bootp Packet received.
Host   (server) Ethernet : 0006 5BE5 8F23
Host   (server) IP       : 205.158.243.30
Client (target) Ethernet : 00D0 CAD1 2577
Client (target) IP       : 205.158.243.205
TFTP Start...
Host   (server) IP       : 205.158.243.30
Client (target) IP       : 205.158.243.205
Loading Filename         : zImage-rf-061603-xhyper250
Save Address             : 0xA0008000
Loading start...
0x000D10B4 (856244) bytes received.
tftp done.
X-HYPER250> flash kernel
Saving kernel to flash...
Erase flash blocks from 0x000C0000 to 0x001BFFFF.
Erase Block at : 0x00180000.
Done.
Write to flash...
Writing at 0x001C0000...
Done
X-HYPER250>
```

Be sure to try rebooting the kernel image to ensure that it programmed successfully:

```
X-HYPER250> reboot
Restarting...
```

Now the kernel has been successfully re-programmed to flash. It is now guaranteed that the *vmlinux* file and *zImage* file will match for kernel debugging purposes.

Debugging the Kernel

Refer to the *Getting Started with MAJIC and Linux* application note for information on configuring the MAJIC for the X-Hyper250 target. In this situation the samples file found in the `/opt/edta/X-Hyper250_linux` directory should be used for connecting to the target. MDI-Server can either be run from the `/opt/edta/X-Hyper250B_linux` directory, or the startup files can be copied to the `/opt/edta/bin` directory before starting.

Keeping the Minicom window open to monitor the target, now open two more console windows. Start MDI-Server in the second console window.

```
[root@mc bin]# mdi-server -l mdi.so
Started listening socket on port 2345.
```

Then start a GDB session in the third console window:

```
[root@mc bin]# ./xscale_le-gdb
GNU gdb 5.2.1
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you
are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for
details.
This GDB was configured as "--host=i686-pc-linux-gnu
--target=armv5tel-hardhat-linux".
```

Now enter the following command in GDB:

```
(gdb) set remoteti 10
(gdb) file /opt/xhyper250/kernel/linux-2.4.18-rmk7-pxa1-xhyper250R1.1/vmlinux
Reading symbols from /tftpboot/vmlinux-rf-061603-xhyper250...done.
(gdb) target remote localhost:2345
Remote debugging using localhost:2345
0x00000000 in ?? ()
```

This is following the *Getting Started with MAJIC and Linux* application note exactly. In the MDI-Server window you should now see:

```
Accepted gdb MDI connection.
Attaching...Capabilities:
   TraceOutput
   TraceCtrl
End of Capabilities List
Devices:
Index    Name
[0]      PXA250 for Hybus Linux
Selecting device [0] of 1

Notification from the target:
Target power detected on VREF
Auto JTAG detection process detected 1 TAP
JTAG connection established
Done.
```

Now, in GDB, let the target run:

```
 (gdb) cont
Continuing.
```

In the Minicom window you should see the Linux kernel begin to boot up:

```
--------------------------------------------------------------------------------
X-HYPER250-R1
Copyright (C) 2002 Hybus Co,. ltd.
X-HYPER250 comes with ABSOLUTELY NO WARRANTY; Read the GNU GPL for
details.     This is free software, and you are welcome to redistribute it
under certain conditions; read the GNU GPL for details.
Support: http://www.hybus.net
--------------------------------------------------------------------------------
Autoboot in progress, press any key to stop ...Autoboot started.
Starting kernel ...
Uncompressing
Linux.....................................................Linux
version 2.4.18-rmk7-pxa1-xhyper250R1.1 (root@mc) (gcc version 3.2.1
200203CPU: Intel XScale-PXA250 revision 4
Machine: X-Hyper250B PXA250 evaluation board
On node 0 totalpages: 16384
zone(0): 16384 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: root=1f03 rw console=ttyS0,115200 init=/linuxrc
Console: colour dummy device 80x30
Calibrating delay loop... 397.31 BogoMIPS
Memory: 64MB = 64MB total
Memory: 62436KB available (1451K code, 292K data, 356K init)
Dentry-cache hash table entries: 8192 (order: 4, 65536 bytes)
Inode-cache hash table entries: 4096 (order: 3, 32768 bytes)
Mount-cache hash table entries: 1024 (order: 1, 8192 bytes)
Buffer-cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 16384 (order: 4, 65536 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
PXA USB Controller Core Initialized
get_random_bytes called before random driver initialization
USB Function Ethernet Driver Interface
Starting kswapd
JFFS2 version 2.1. (C) 2001 Red Hat, Inc., designed by Axis Communications
AB.  Console: switching to colour frame buffer device 80x30
pty: 256 Unix98 ptys configured
Serial driver version 5.05c (2001-07-08) with no serial options enabled
ttyS00 at 0x0000 (irq = 14) is a PXA UART
ttyS01 at 0x0000 (irq = 13) is a PXA UART
ttyS02 at 0x0000 (irq = 12) is a PXA UART
ads7843 touch screen driver initialized
block: 128 slots per queue, batch=32
Cirrus Logic CS8900A driver for Linux (V0.01)
eth0: CS8900A rev D at 0xf0000300 irq=0, no eeprom , addr:
00:12:34:56:78:9A    ac97_codec: AC97 Audio codec, id: 0x4352:0x5914
(Cirrus Logic CS4297A rev B)    Probing X-Hyper250 Flash at physical
address 0x00000000                          Using buffer write method
Using static partition definition
Creating 4 MTD partitions on "X-Hyper250 Flash":
0x00000000-0x00040000 : "Bootloader"
0x00040000-0x000c0000 : "Partition Tables"
0x000c0000-0x001c0000 : "Kernel"
0x001c0000-0x02000000 : "Filesystem"
Linux Kernel Card Services 3.1.22
options:  none
Intel PXA250/210 PCMCIA (CS release 3.1.22)
X-Hyper250 PCMCIA INIT
```

```
MMC subsystem, $Revision: 0.3.1.14 $
MMC block device driver, $Revision: 0.3.1.16 $
PXA250 MMC controller driver, $Revision: 0.3.1.12 $
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 4096 bind 8192)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com
VFS: Mounted root (jffs2 filesystem).
Freeing init memory: 356K
Setting up RAMFS, please wait ...
done and exiting
INIT: version 2.78 booting
INIT: Entering runlevel: 3
Starting syslog Starting system logger:
Starting pcmcia Starting PCMCIA
Starting etwork Starting network
Starting X...
Xhyper250RB login:
```

Log into Embedded Linux as the root user:

```
Xhyper250RB login:  root
```

You can now change directory and view the file structure of the operating system:

```
[root@Xhyper250RB /root]$cd ..
[root@Xhyper250RB /]$ls
bin   dev  home  linuxrc     mnt   rd    sbin  usr
conf  etc  lib   lost+found  proc  root  tmp   var
[root@Xhyper250RB /]$
```

Begin debugging the Linux kernel by setting, and hitting, several breakpoints. Use the Cntrl-C command to gain control of the target, then type the following commands

```
Program received signal SIGTRAP, Trace/breakpoint trap.
(gdb) b sys_sync
Breakpoint 2 at 0xc009074c: file buffer.c, line 366.
(gdb) b get_pid
Breakpoint 3 at 0xc006ca1c: file fork.c, line 89.
(gdb) info breakpoints
Num Type           Disp Enb Address    What
2   breakpoint     keep y   0xc009074c in sys_sync at buffer.c:366
3   breakpoint     keep y   0xc006ca1c in get_pid at fork.c:89
(gdb) cont
Continuing.
```

In Embedded Linux run the `list` (ls) command in order to cause the breakpoint at `get process identification` (get_pid) to be hit:

```
[root@Xhyper250RB /]$ls
```

In GDB you should see:

```
Breakpoint 3, get_pid (flags=17) at fork.c:89
89 {
```

Let the target run again and try executing the sync command:

```
(gdb) cont
Continuing.
```

```
[root@Xhyper250RB /]$sync
```

In GDB you should see the get_pid breakpoint hit first, and then if you continue, the sys_sync breakpoint hit.

```
Breakpoint 3, get_pid (flags=17) at fork.c:89
89 {
(gdb) c
Continuing.

Breakpoint 2, sys_sync () at buffer.c:366
366            fsync_dev(0);
(gdb)
```

You have now successfully demonstrated kernel level debugging on the Hybus X-Hyper250B board.

Flash Programming

To program the Hybus X-Hyper250B you have two options; The EPI flash programming utility can be used, OR, the Microvision flash programming utility can be used.

EPI Flash Programming Utility

To use the EPI flash programming utility refer to the *Getting Started Using the EPI Flash Programming Utility* application note, and use the following files and settings:

1. If using Windows, when creating a flash programming shortcut with the MAJIC Setup Wizard, use the samples files in the `\EDTA\samples\X-Hyper250B` directory. If using MONICE, call MONICE from the `/opt/edta/samples/X-Hyper250B` directory.

2. Choose the flash programming utility compiled for the memory region 0xA0008000. The full path to this file is: `C:\Program Files\EDTA20\samples\le\RAM_0xA0008000\flash.axf`, or `/opt/edta/samples/le/RAM_0xA0008000/flash`.

3. Choose the Intel **`28F128J3A`** flash part.

4. Set the flash base address to **`0x00000000`**.

5. In the Flash Device Menu (9) select **`2 parts in Parallel`**. Be sure to double check that the bus width is 16 bits times 2, to equal a 32 bit bus.

Microvision Flash Programming Utility

To use the Microvision flash programming utility follow the instructions below:

1. Extract the program.zip file provided by Microvision into the C:\ directory.

2. Create a shortcut to MONICE using the MAJIC Setup Wizard, making sure to select the sample files in the `C:\Program Files\EDTA20\samples\X-Hyper250` directory.

3. Run the MONICE debugger, and type on the command line:

Mon>> fr c c:\make-arm\HYBUS-PXA250\flash.cmd

This flash programming utility automatically programs the bootloader into the Hybus X-Hyper250 target without any options. All options are hard-coded into the flash.cmd file

In order to change the default settings of this flash programmer, such as the image address, block sizes, or file size, you must edit the flash.cmd file.

To change the block size and file size, respectively, change the following commands in the flash.cmd file:

ew $blksize = 0x10000

ew $filesize = 0x10000

To change the hardware start address edit the line:

ew $flash_start=0x00000000

.

References

Title	Filename	Source
X-Hyper250B Xscale PXA250 Evaluation Board	X-Hyper250B-1.3_eng.doc	Hybus
Getting Started with MAJIC and Linux	AppNote_MAJIC_Linux.pdf	EPI
Getting Started Using the EPI Flash Programming Utility	AppNote_MAJIC_Flash.pdf	EPI