Introduction to Embedded Systems
Design and Implementation

Development Environment of Embedded System
Outline

- Embedded hardware development
- Embedded software development
FPGA board

Source: http://www.altera.com/  
Source: http://www.hitechglobal.com/Boards/Virtex4LX200.htm
FPGA

• Field Programmable Gate Array (FPGA)
  • a general purpose chip which can be programmed to carry out a specific hardware function

Source:
http://www.xilinx.com/
http://www.altera.com/products/devices/cyclone2/cy2-index.jsp
Development tools
Programmable logic design process

- Programmable logic design process

Source: http://www.netrino.com/Articles/ProgrammableLogic/
Embedded System Development Tools and Platform

- Embedded System Evaluation Board (for example)
Embedded System Development Tools and Platform

• Embedded System Evaluation Board with FPGA add-on (for example)
Embedded System Development Tools and Platform

- ICE/JTAG interface

Source: www.arm.com
Software development
Embedded System Development Tools and Platform

- ICE/JTAG interface

Source: http://www.keil.com/ulink/
Embedded System Development Tools and Platform

- ICE/JTAG interface

Source: www.arm.com
Embedded Software Development Tools

Source: Volker Soffel, “Embedded Programming”
Embedded Software Development Tools

- **Console mode**
  - GNU toolchain
    - gcc: a cross-compiler
    - binutils: a set of tools for manipulating binaries
    - glibc: c-lib
    - gdb: debugger (ICE is required)

- **IDE**
  - ADS
  - GNUPro
  - ...

Embedded Software Development Tools

Source: www.arm.com
Embedded Software Development Tools

Source: www.arm.com
Equipment Requirement

- PC
- WINeZ ARM (ICE)
- JTAG POD
- PreSOC development learning platform

Source: http://www.microtime.com.tw/
Create XScale-PXA270

Creator PreSOCes

Source: http://www.microtime.com.tw/
Create XScale-PXA270 (4)

Source: http://www.microtime.com.tw/
Functions in Domingo

- Program window
- Register window
- Memory window
- Breakpoints window
- Source Editor window
- Watch List window
Remote Debug by Domingo

Source: http://www.microtime.com.tw/
Simple development environment
PCM-7230 board

PCM-7230 connections

Circuit Reference:

- CRT out (CN8)
- 3 Push buttons (CN3, Pin1~Pin6)
- COM1 (CN9, Pin33~Pin42)
- USB Port (CN9, Pin47~Pin58)
- Power switch (JP4)
- power in cable (CN29)

JP2
First pin is right
(see the next page)

GNU Debugger (GDB)

- Allows you to see what is going on `inside' another program while it executes
  - Start your program, specifying anything that might affect its behavior
  - Make your program stop on specified conditions
  - Examine what has happened, when your program has stopped
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another
Graphic User Interface to GDB

- It makes GDB easier to use
  - DDD
  - Insight

![Diagram showing Graphic User Interface to GDB]
Insight

- A graphic user interface to GDB
- Written by Tcl/Tk, since 1994
- From Red Hat and Cygnus Solutions
- [http://sources.redhat.com/insight/](http://sources.redhat.com/insight/)
- GPL license
Prepare to Build Cross Debugger

- Cross compiler, binutils and newlib are ready
- Download gdb source package
  - ftp://sources.redhat.com:/pub/gdb/releases/gdb-6.4.tar.gz
  - ftp://sources.redhat.com:/pub/insight/releases/insight-6.4.tar.bz2
- Uncompress source package
  - tar –zxvf gdb-6.4.tar.gz
  - tar –jxvf insight-6.4.tar.bz2
Build Cross Debugger (1)

- Build cross binutils (--prefix=/foo)
- **Add /foo/bin to PATH**
- Build cross compiler (--prefix=/foo)
- Build cross newlib (--prefix=/foo) (optional)
- Configure GDB
  - 
    ```
    ./configure --prefix=/foo --target=arm-elf \
    --enable-sim
    ```
  - make
  - make install
Practice on Source-Level Debug (1)

- Use cross compiler to compile test.c
  - "-g" option: add debug information which makes possible source-level debug

```
% /foo/bin/arm-elf-gcc -g test.c -o test.exe
```
Practice on Source-Level Debug (2)

- **Local side**: Run X server in cygwin
- **Local side**: Allow the X window program on remote side to display on local machine
  
  
  ```
  % xhost + target.yy.yy.yy
  ```

- **Remote side**: Set IP address of X server which we want to display on
  
  ```
  % export DISPLAY=myip.xx.xx.xx:0.0 (bash)
  
  % setenv DISPLAY myip.xx.xx.xx:0.0 (csh or tcsh)
  ```
Practice on Source-Level Debug (3)

- **Local side**: Run **Ingisht** (GUI of GDB)

  `% /foo/bin/arm-elf-insight`
Practise on Source-Level Debug (4)

- **Local side**: Load file: test.exe
  - Select menu: File -> Open
  - Load the program we want to debug
Practice on Source-Level Debug (5)

- **Local side**: Set target machine to simulator
  - Select menu: File -> Target Settings
  - Set **Target** to **ARM simulator**

Set breakpoint at “main” and “exit”
Practice on Source-Level Debug (6)

- **Local side**: Begin to do source-level cross debug on ARM simulator
  - Select menu: RUN-> Run
  - Begin to debug
Kernel debugger

- **Hardware setup**
  - Use a Null modem serial cable to connect across the Target and Development machines
  - On the Development machine
    - `stty ispeed 115200 ospeed 115200 -F /dev/ttyS0`
    - `cat testfile.txt > /dev/ttyS0`
  - On the Target machine:
    - `stty ispeed 115200 ospeed 115200 -F /dev/ttyS0`
    - `cat /dev/ttyS0`

- **Software setup**
  - Downloaded Linux kernel source: `linux-2.6.6.tar.gz`
  - Downloaded the 2.6.7 patch: `patch-2.6.7.bz2`
  - Downloaded the Kgdb patch: `linux-2.6.7-kgdb-2.2.tar.bz2`
  - Apply patch and kgdb patch
  - Compiling the kernel on the development machine
Kernel debugger (Cont.)

- Starting the debug session
  - After booting the target machine will wait for the host development machine to connect, by displaying the message:
  - Waiting for connection from remote gdb...
  - For setting a debug session with baud rate of 115200 on /dev/ttyS0, run as "root" user:
    - <root#> gdb ./vmlinux

http://kgdb.linsyssoft.com/quickstart.htm
Any question?