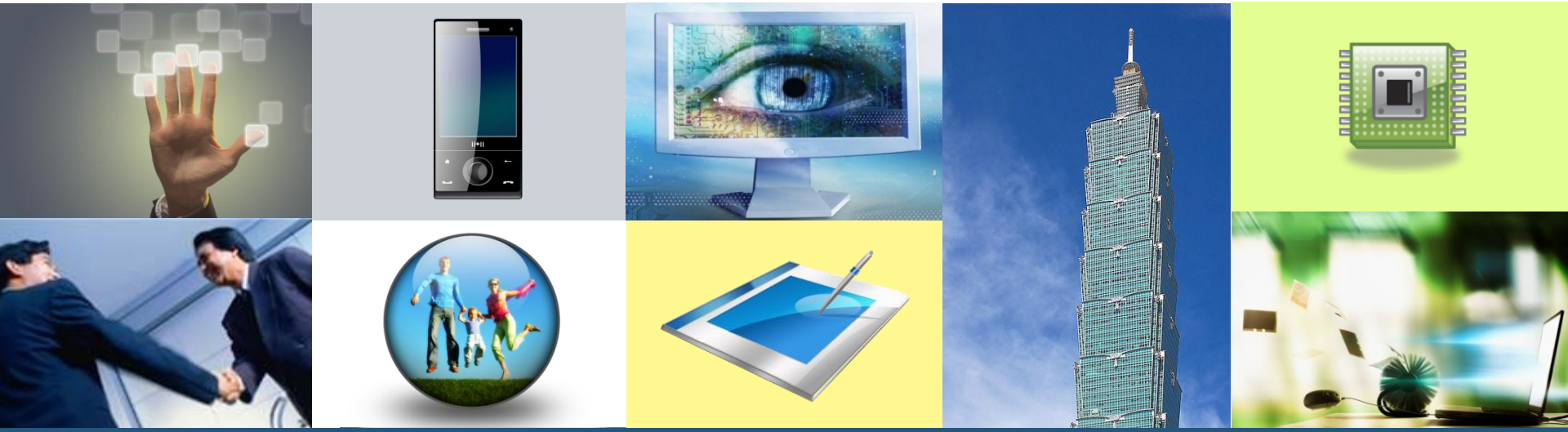
Dual Core(D1088) AICE Debug with Demo Project

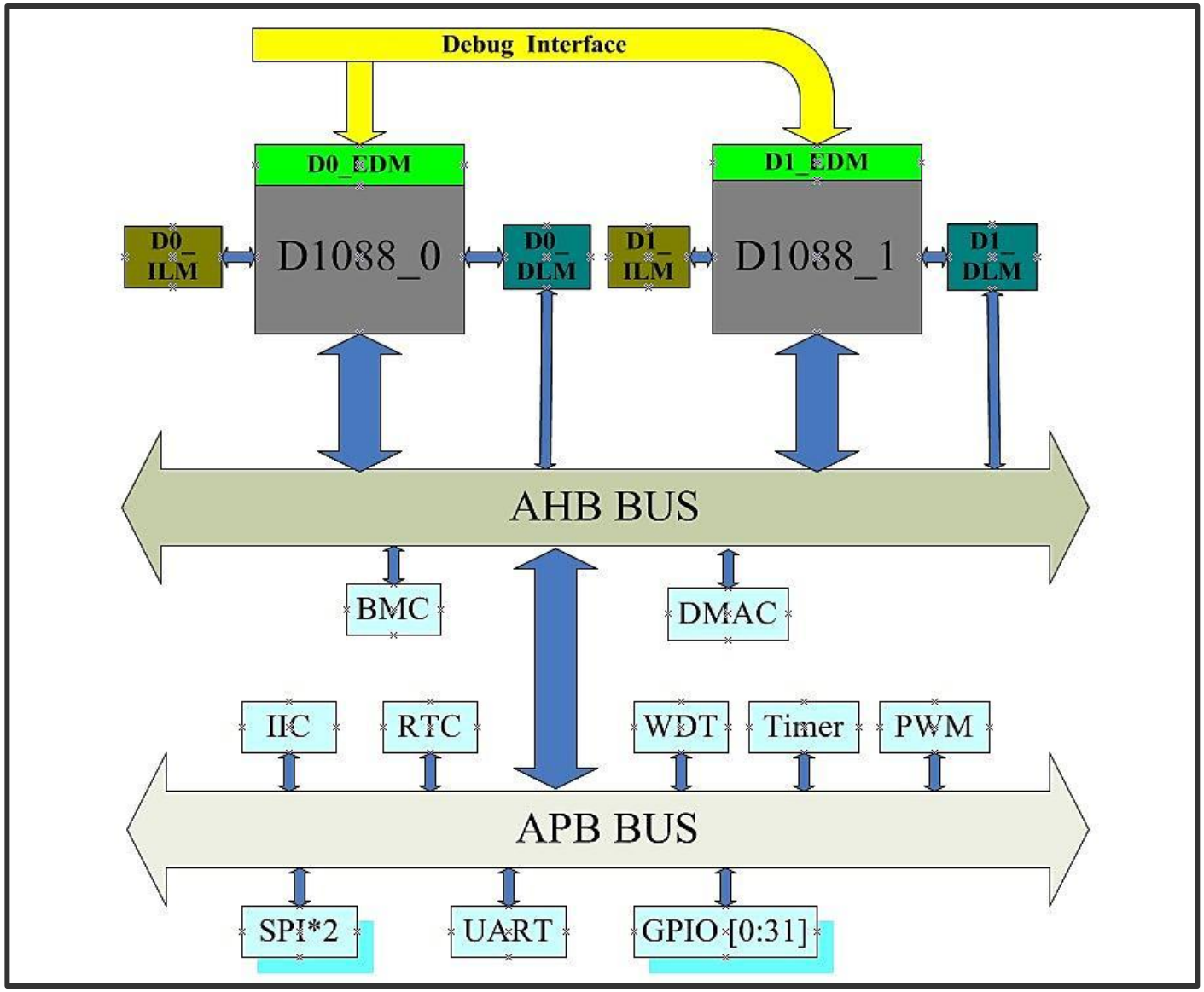
Driving Innovations™



Agenda

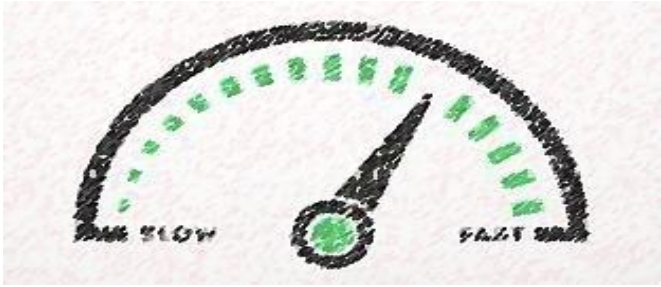
- Block Diagram of Dual core(D1088)
- Need for Multi Core Debugging
- Andes Multi Core Debug Connection
- Demo Environment Setting
  - Demo Environment
  - Demo Project Introduction
  - Flow chart
- Demonstration (For Execution in AndeSight™ v3.0.0)
  - Check Diagnostic Report (For Confirmation of cores)
  - Procedure to follow
- Demonstration (For Execution in Cygwin Window)
  - Issue ICEman.exe
  - Open Cygwin
  - Execute GDB
  - Load and Run Project

Block Diagram of Dual Core(D1088)

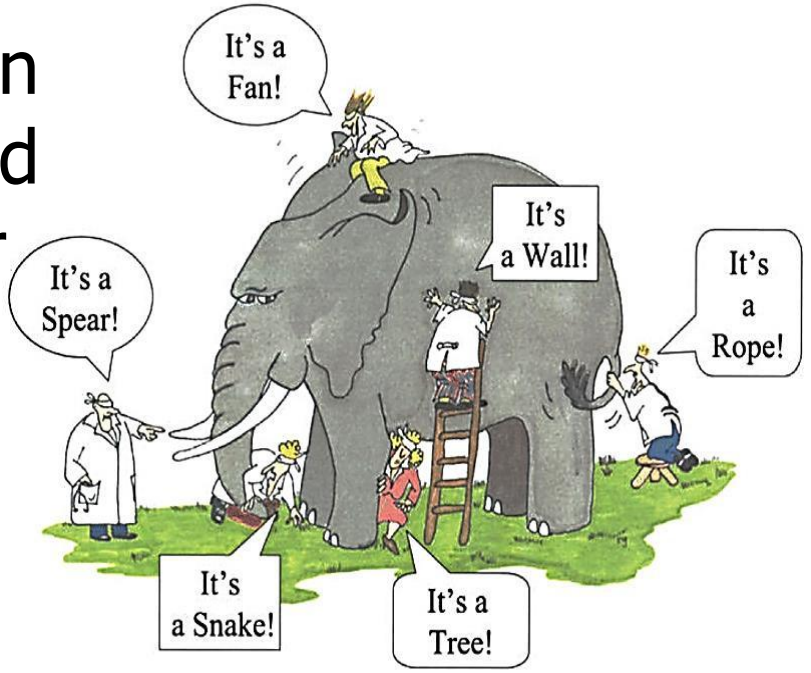


Need for Multi Core Debugging

- In complex SoCs, just to observe and control a single core is insufficient.

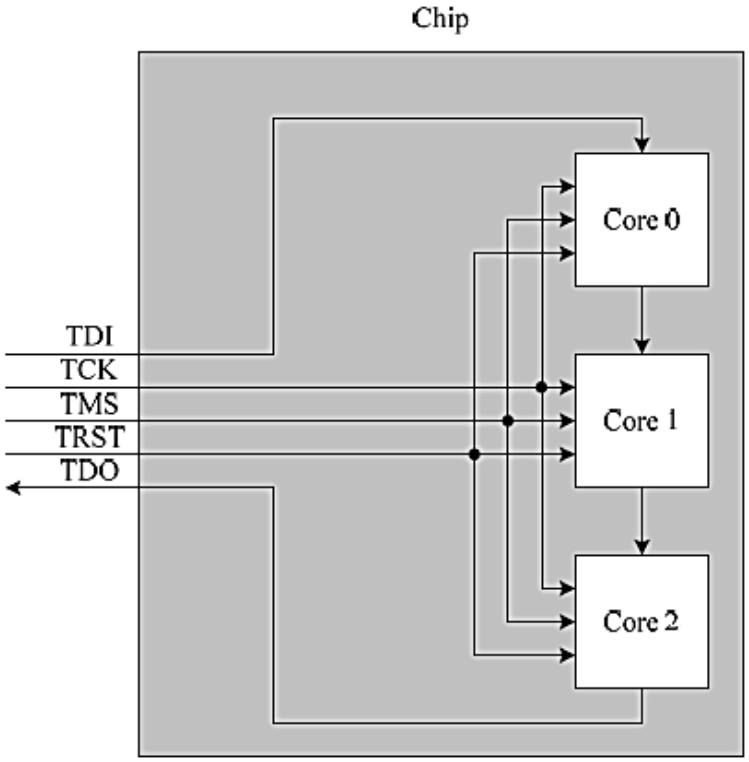


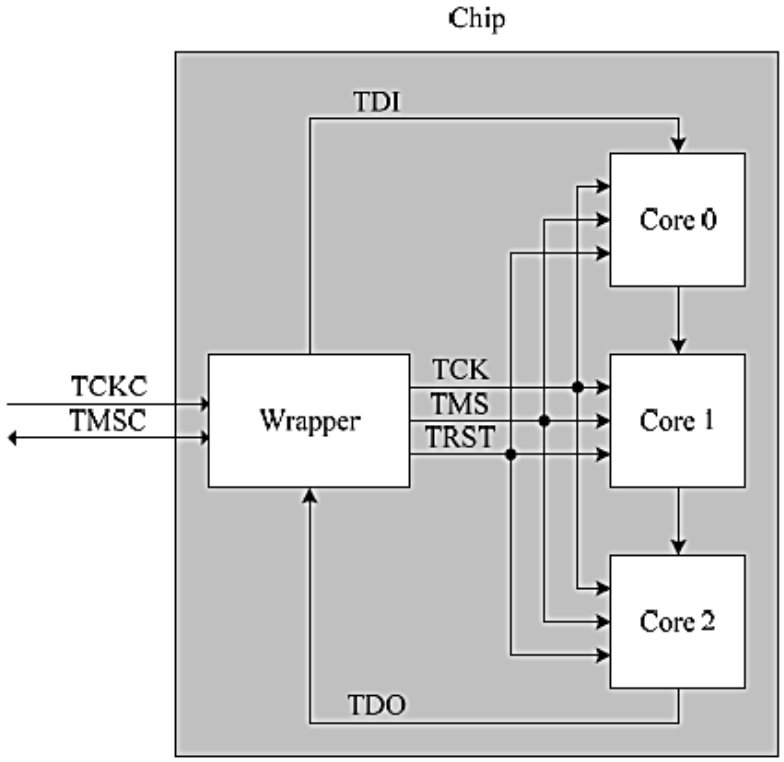
- Rather the interactions of multiple cores are needed when user wants to detect, trace, and eliminate software problems or to profile system behavior for performance optimization.



Andes Multi Core Debug Connection

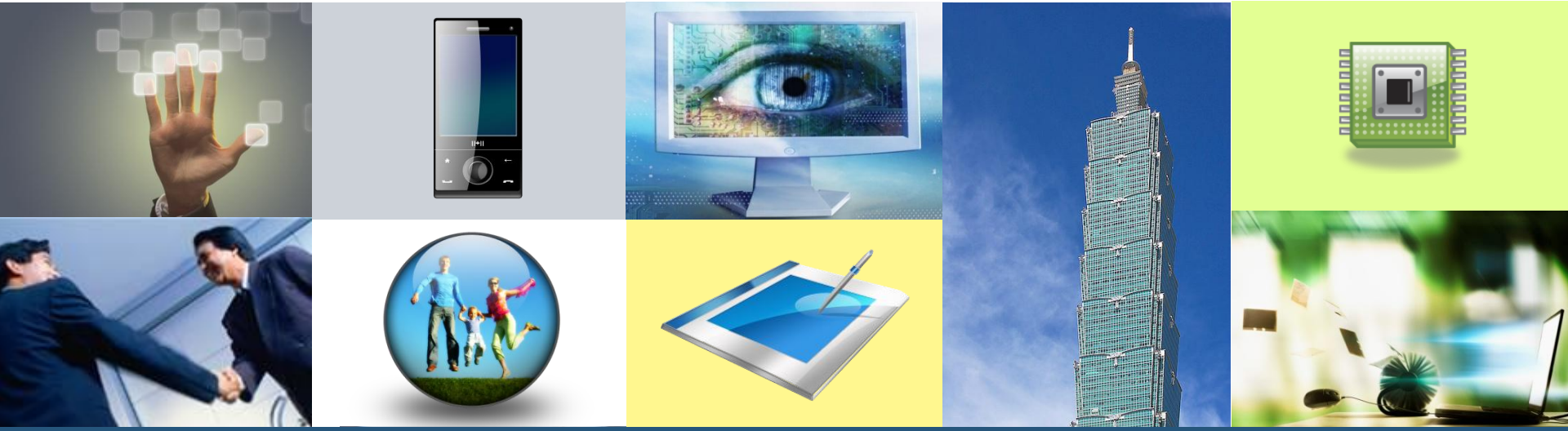
- Single-chip Multi-Core System
  - 5-wire interface (AICE-MCU)
  - 2-wire interface (AICE-MINI)





Demo Environment Setting

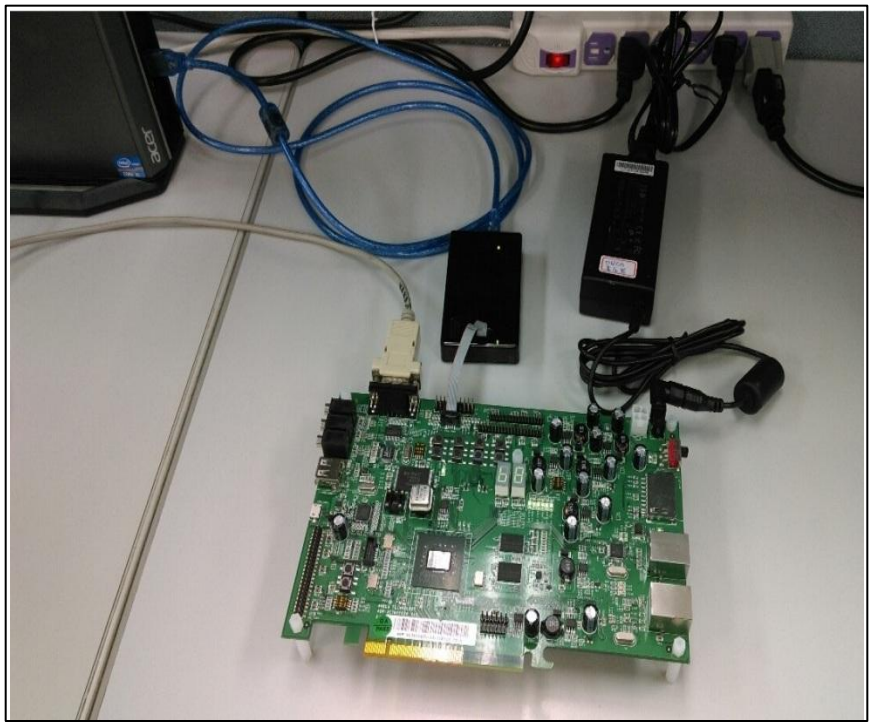
Driving Innovations™



Demo Environment

- Netlist: Dual D1088+AE210P
- Demo project:
  - Core 0: C0_7SEG
  - Core 1: C1_int
- Components:

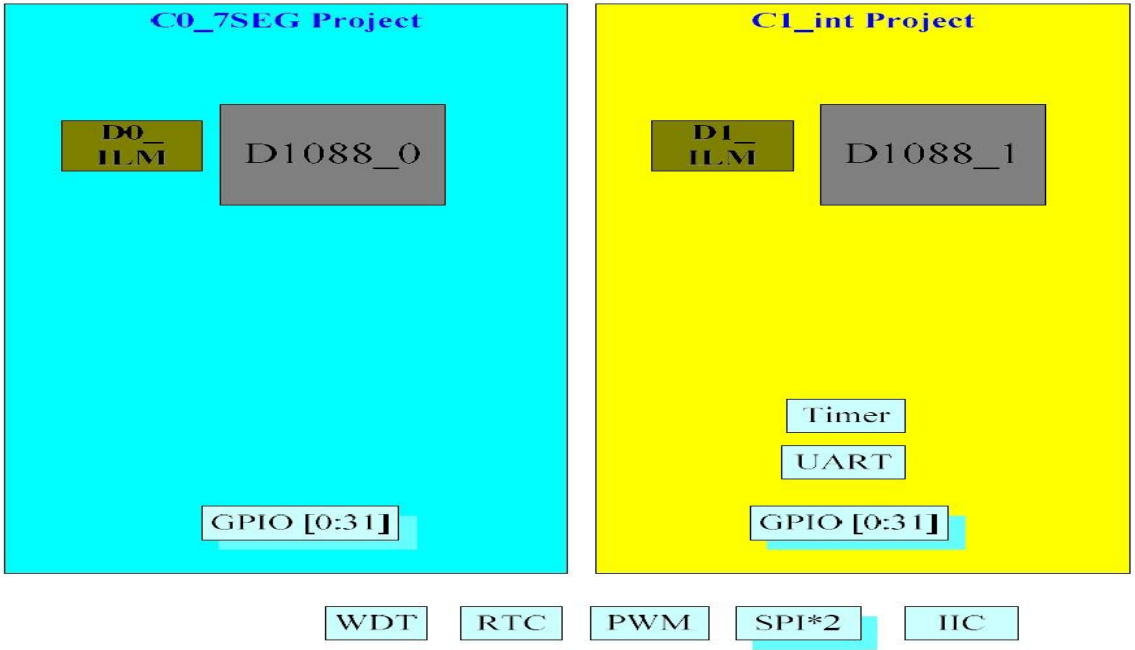
XC7_FPGA Board , AICE- MCU & UART cable.



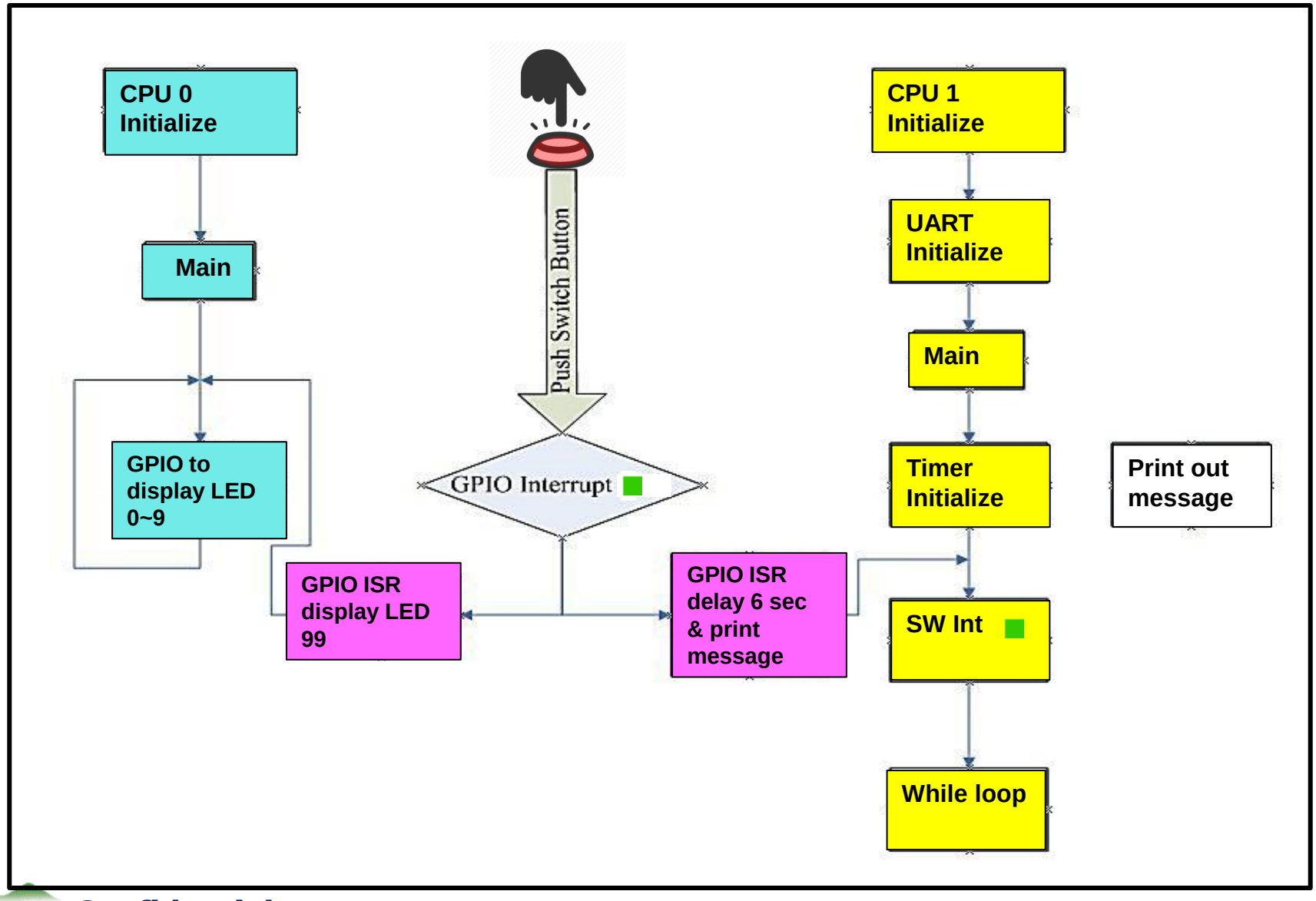
- NOTE: Though ICEman supports the assignment of AICE configuration file to multi-cores at a time, BSP v4.0 accepts only one coprocessor for a CPU core.

Demo Project Introduction

- The demo scenario is:
  - Let both CPU(D1088) do "Reset and Hold"
  - Load individual project(C0_7SEG & C1_int) to ILM in each CPU Core.
- Both C0_7SEG & C1_int program execute at separate core(D1088)
- A custom script can be used to perform AICE RESET-HOLD in a multi-core system if the ICEman default reset timing is not suitable.

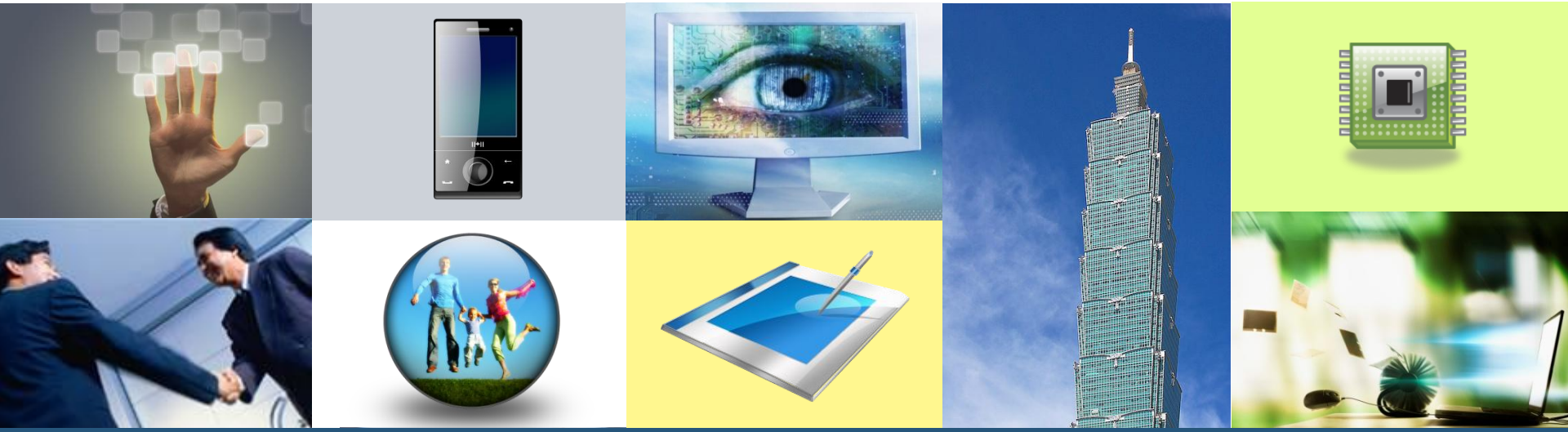


Flow Chart


CPU 0 Initialize
Main
GPIO to display LED 0~9
Push Switch Button
GPIO Interrupt
GPIO ISR display LED 99
GPIO ISR delay 6 sec & print message
CPU 1 Initialize
UART Initialize
Main
Timer Initialize
Print out message
SW Int
While loop

Demonstration
(For Execution in AndeSight™ v3.0.0)

Driving Innovations™

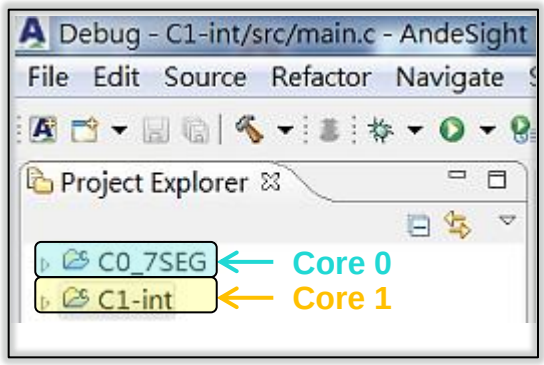


Check Diagnostic Report (For Confirmation of cores)

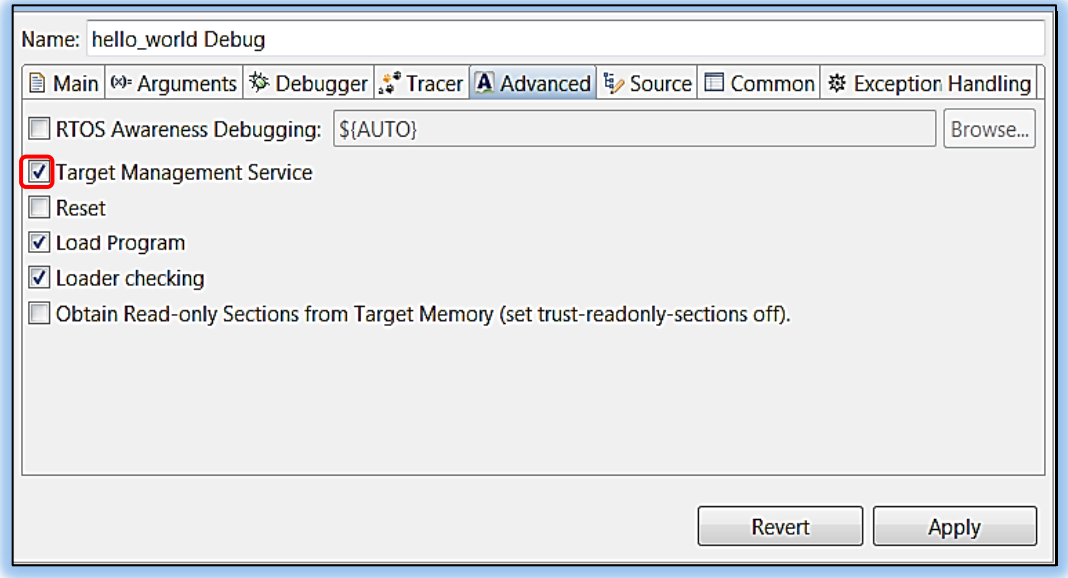
```
Andes ICEman v3.4.0 (OpenOCD) BUILD_ID: 2017022117
Burner listens on 2354
Telnet port: 4444
TCL port: 6666
Andes AICE-MCU v1.8.8
There are 2 cores in target
JTAG frequency 24 MHz
**********
Diagnostic Report
**********
JTAG frequency 24 MHz, 9
set JTAG frequency 12 MHz
set JTAG frequency 24 MHz
ice_state = 00000009
There are 2 cores in target
Core #0: EDM version 0x1010
hardware reset-and-hold success
**********
[PASS] check USB connectivity
[PASS] check AICE versions
[PASS] check the detected JTAG frequency
[PASS] check changing the JTAG frequency
[PASS] check JTAG connectivity
[PASS] check that JTAG domain is operational
[PASS] check that TRST resets the JTAG domain
[PASS] check that SRST does not resets JTAG domain
[PASS] check selecting core
[PASS] check reset-and-debug
[PASS] check that DIM and CPU domain are operational
**********
Core #1: EDM version 0x1010
hardware reset-and-hold success
**********
[PASS] check USB connectivity
[PASS] check AICE versions
[PASS] check the detected JTAG frequency
[PASS] check changing the JTAG frequency
[PASS] check JTAG connectivity
[PASS] check that JTAG domain is operational
[PASS] check that TRST resets the JTAG domain
[PASS] check that SRST does not resets JTAG domain
[PASS] check selecting core
```

Procedure to follow

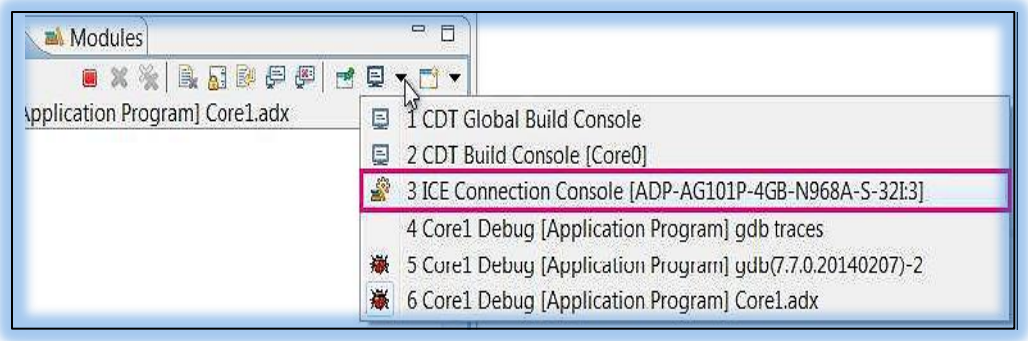
1) Create and build project for each core



2) Launch a debug session for the Core 0 project.



3) Check the Console view and switch to “ICE Connection Console” (For Core 0)



- Note: Target Management Service” in the Debug Configurations dialog must be selected.

Procedure to follow (1)

- ✓ Take note of the port number that each core listens to in the ICE Connection Console. For example:
  - The **Core #0** listens on 9902.
  - The **Core #1** listens on 9903.

3) Run the debug session and note the CPU core and Port number

```
11 }
12 int main()
13
14 {
15 while (1){
16     outw(GPIO_BASE + GPIO_DIR_OFFSET, 0x00BF0000); /*SEG'0'*/
17     delay(1000000);
18
```

Console | Problems | Properties | Terminal | GDB Command | Disassembly

C0_7SEG.adx [Application Program] gdb(7.7.0.20140207)-4

```
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin --target=nds32le-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
```

Breakpoints

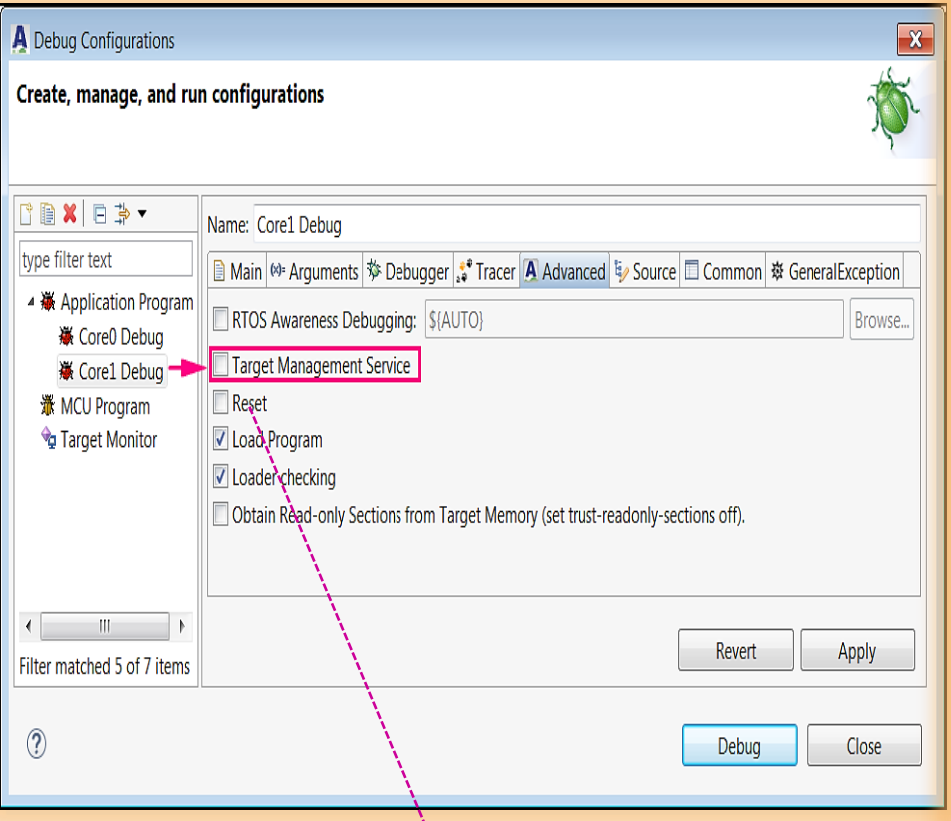
djpeg.c [line: 554]

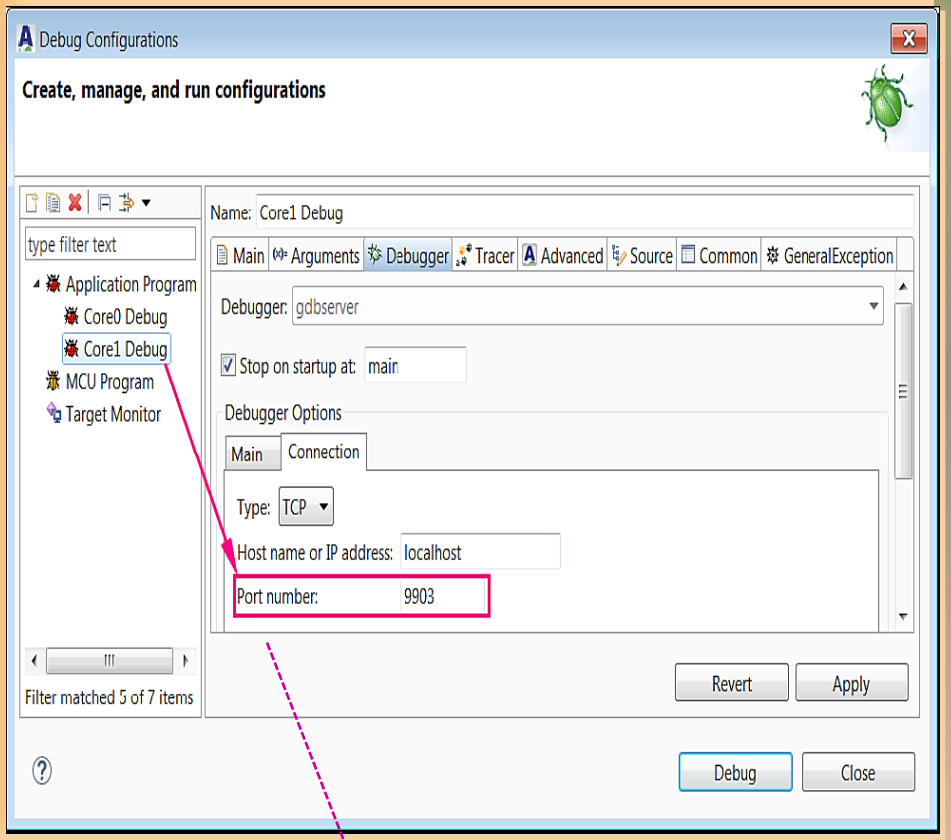
Select process

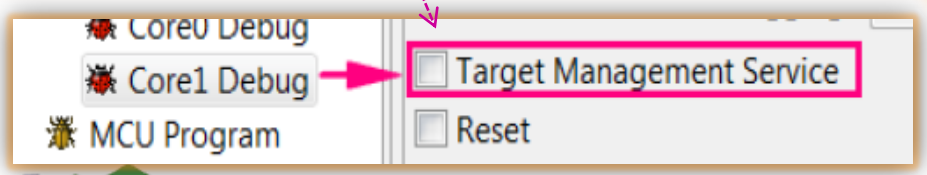
Target: ADP-AE210P-D1088
CPU0 GDBServer: 9902
CPU1 GDBServer: 9903
Burner Port: 9900

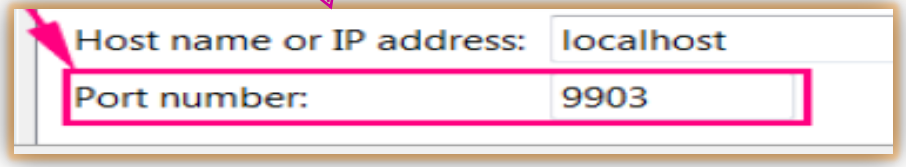
Procedure to follow (2)

4) Uncheck “Target Management Service” and go for “Debugger > Connection” tab to specify the port no. (For Core 1)







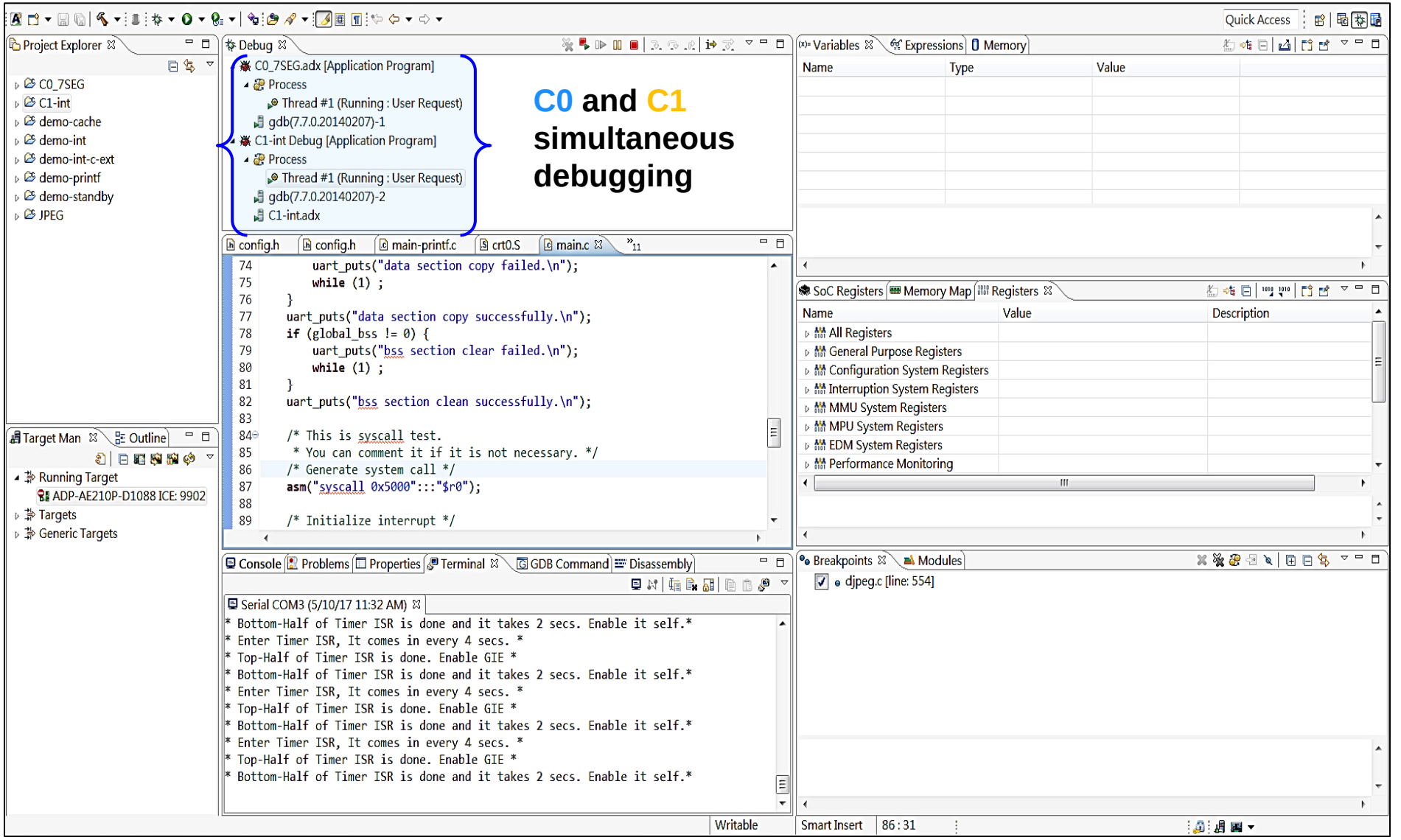


NOTE:

Multiple projects may be used simultaneously for multi-core debugging. As breakpoints in AndeSight are global across projects in the workspace, a breakpoint that user sets for a project at a source line or on an address may be hit in other projects having the same filename or address.

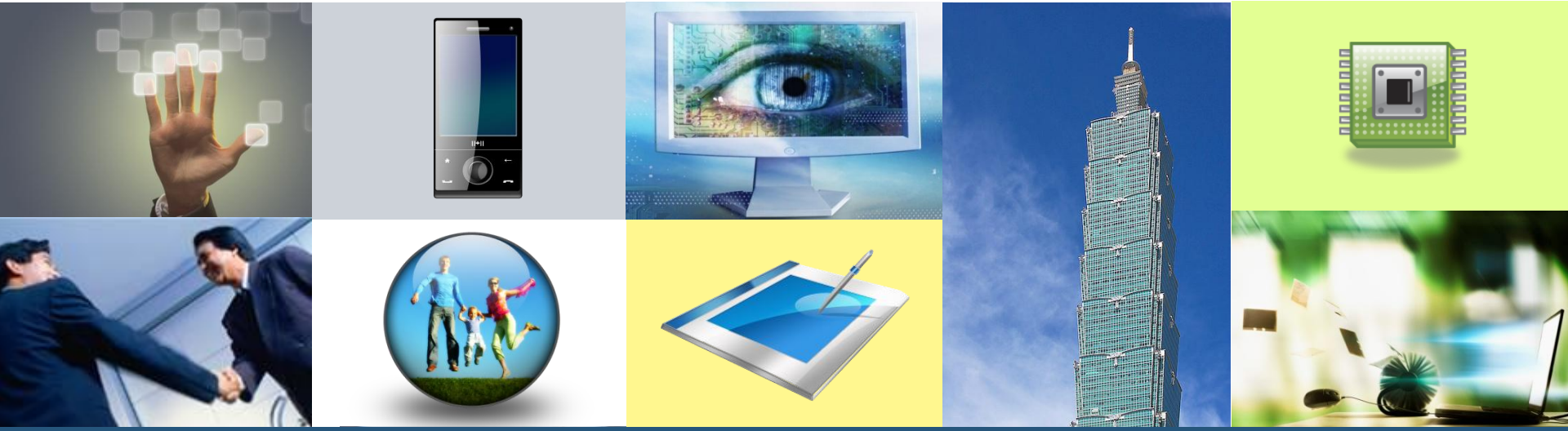
Please close unused debug sessions to avoid such a case – or just resume the debug session if an unexpected breakpoint is hit.

Procedure to follow (3)



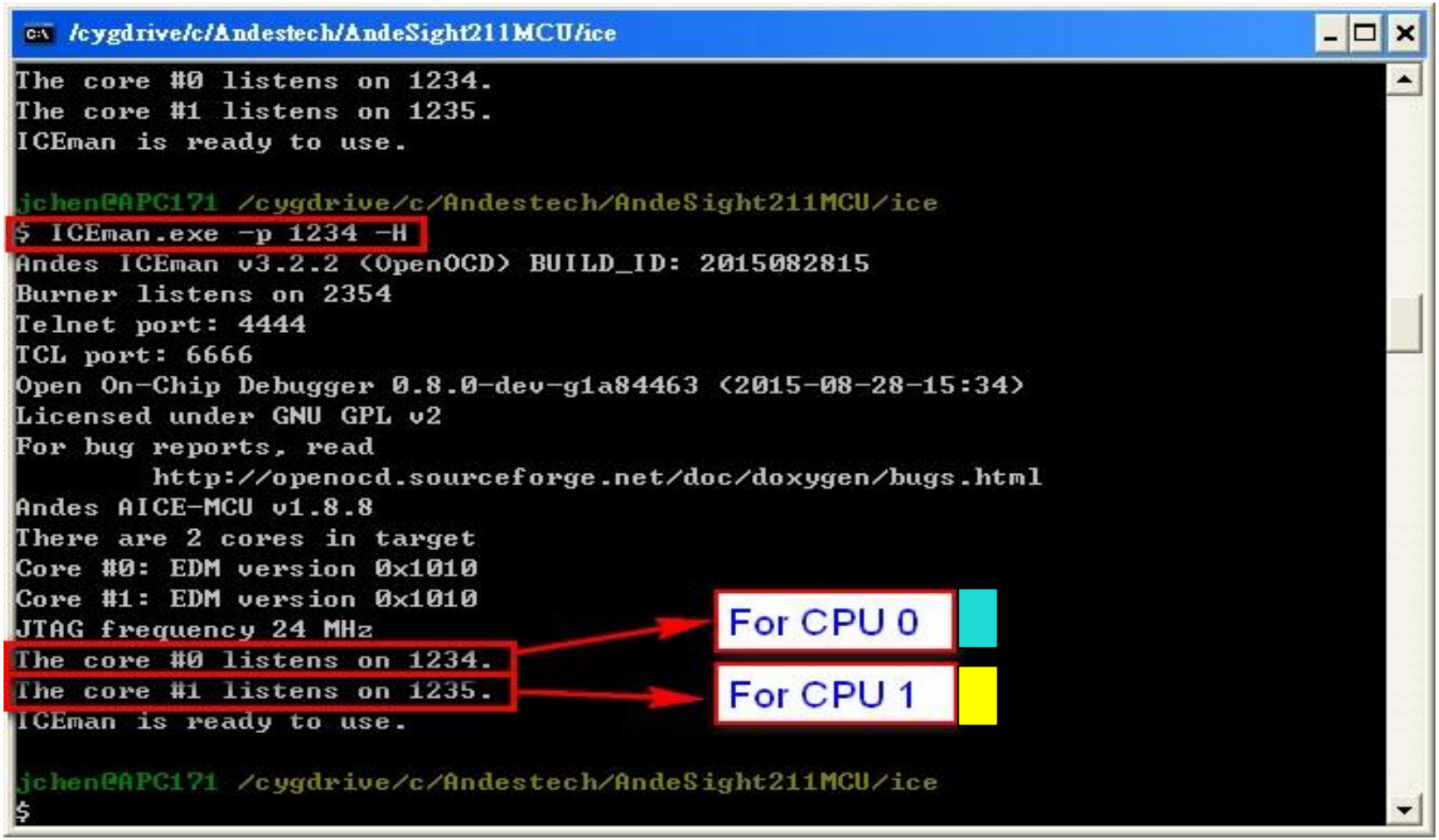
Demonstration
(For Execution in Cygwin Window)

Driving Innovations™



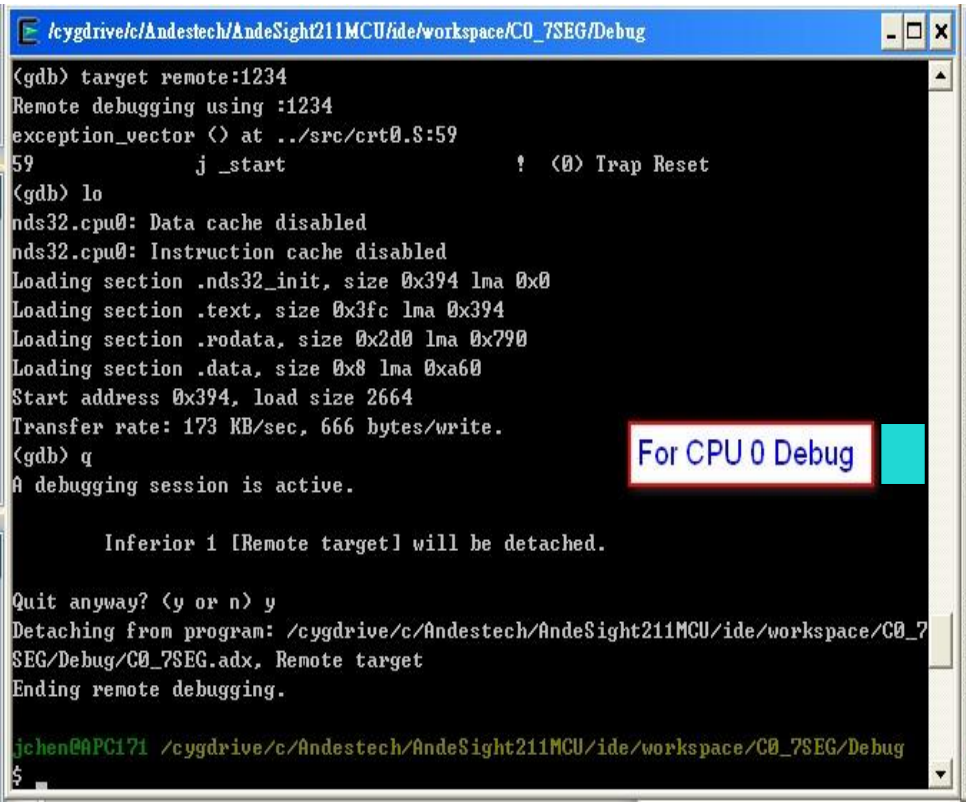
Issue ICEman.exe

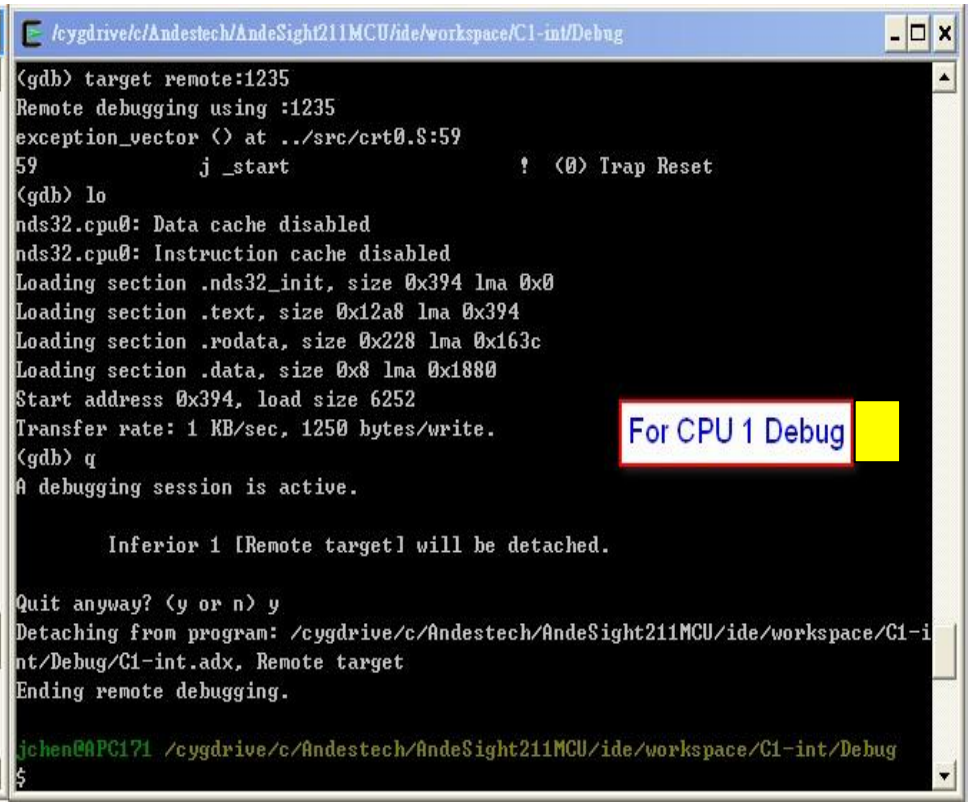
- Open **Cygwin** window
- Issue ICEman.exe –p 1234 –H to get the GDB Port ID.



Open Cygwin

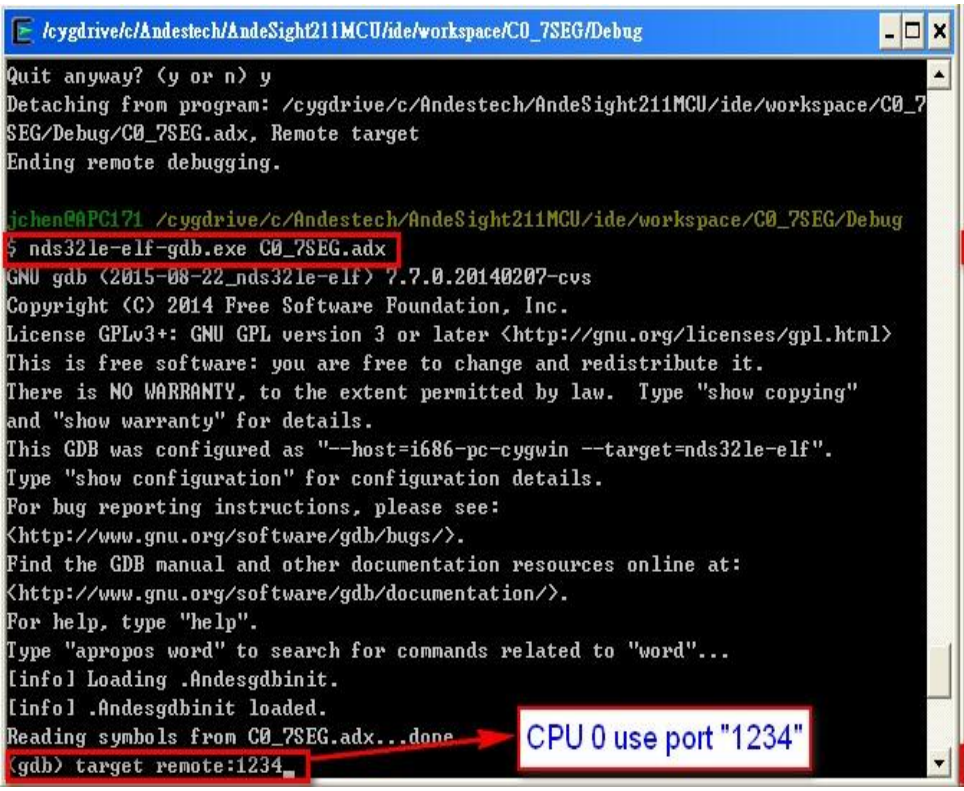
- Open two Cygwin windows for separate CPU .

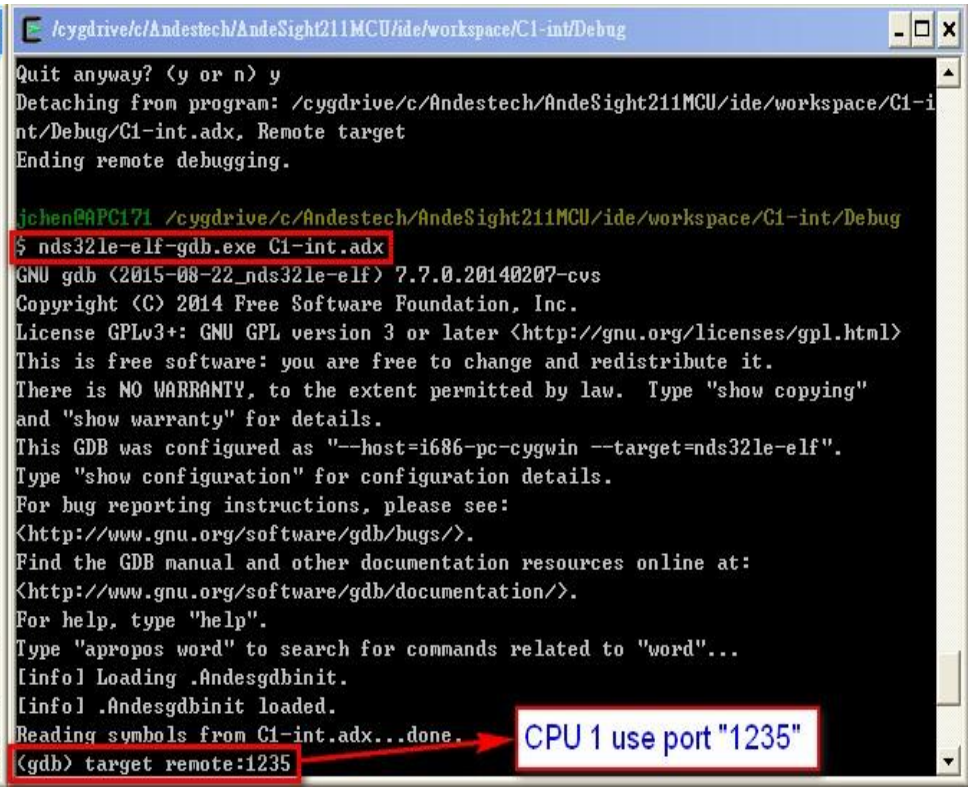




Execute GDB

- Go through individual GDB port to connect separate CPU





Load and Run Project

- Load into different projects to the implementation of the ILM on each CPU, and start Debug.