

Workshop Embedded systems

Linux Debug a New Paradigm

André De Ceuninck a.d.ceuninck@logic.nl

Sept. 30 2005



Embedded Linux

Hardware-Assisted Debugging Tools for Kernel, Process and Application Debug - ARM Processors -



Why Hardware-Assist ?

- Stable platform is NOT required
- Non-Intrusive, Real-Time Code Execution
- Does not Lose Context after System Reset
- Complete System Visibility to all ASIC or ASSP devices / registers

A new model for Linux debug

Logic
Solutions
are available for:

ALTERA
AMD
ARC
ARM*
REESCALE
HITACHI
INFINEON
INTEL
MIPS*
NEC
NIOS
PHILIPS
RENESAS
ST MICRO
TEXAS INST.
TRANSMETA
VIA

- **Virtual communication port via DCC**
 - Including Linux Console redirection
- **Source-level kernel debug**
- **Process Resource Lists**
- **Application Debugging**
- **All from within the same debugger**

*And licensees

May. 15, 2005



There's always a Logic Solution



- ALTERA
- AMD
- ARC
- ARM*
- REESCALE
- HITACHI
- INFINEON
- INTEL
- MIPS*
- NEC
- NIOS
- PHILIPS
- RENESAS
- ST MICRO
- TEXAS INST.
- TRANSMETA
- VIA

Virtual Communication Port

- **Debug Communications Channel (DCC)**
 - Contained in the EmbeddedICE logic (CP14 register)
 - Allows target software to communicate with emulator / debugger
 - 32-bit data read register, 32-bit data write register
 - 6-bit communication control register for handshaking
 - XScale also uses this for debug message handling

- **Linux DCC driver**
 - Implements multiple tty devices in Linux OS
 - Typically built in with kernel code
 - Eliminates need for other communications ports (e.g. serial, TCP/IP, USB)

```
Target Console
BusyBox v0.60.5 (2004.03.26-05:08+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

/dev # cd /etc
/etc # ls
default          hosts.equiv      issue.net        passwd          samba
dhcpcd           inetd.conf       motd             profile         services
fstab            init.d           mtab             rc              shells
group            inittab          network          rc.d            resolv.conf
host.conf        inputrc          nsswitch.conf   rpc
hosts            issue            pam.d

/etc # more inittab
::sysinit:/etc/rc
#::askfirst:-/bin/sh
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/ttyS0
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/tts/0
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/ttyS1
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/tts/1
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/ttyDCC0
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/ttyDCC1
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/ttyDCC2
#::respawn:/sbin/getty -L -n -l /bin/sh 38400 /dev/ttyDCC3
```



- ALTERA
- AMD
- ARC
- ARM*
- REESCALE
- HITACHI
- INFINEON
- INTEL
- MIPS*
- NEC
- NIOS
- PHILIPS
- RENESAS
- ST MICRO
- TEXAS INST.
- TRANSMETA
- VIA

Linux Console output via JTAG

- Output starts at boot
- No need to wait for initialization of peripheral communication
- All Linux command line messages and functionality available



```
Target Console
Linux version 2.4.20_mvl131-brh (john@AAIP061) (gcc version 3.3.1 (MontaVista
3.1-3.0.10.0300532 2003-12-23)) #14 Tue May 18 17:04:04 PDT 2004
CPU: XScale-80200 [69052001] revision 1 (ARMv5TE)
CPU: D undefined 5 cache
CPU: I cache: 32768 bytes, associativity 32, 32 byte lines, 32 sets
CPU: D cache: 32768 bytes, associativity 32, 32 byte lines, 32 sets
Machine: ADI BRH
On node 0 totalpages: 32768
zone(0): 32768 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: root=/dev/ram initrd=0xc2000000,4000K
Calibrating delay loop... 104.65 BogoMIPS
Memory: 128MB = 128MB total
Memory: 123836KB available (1242K code, 315K data, 228K init)
XScale Cache/TLB Locking Copyright(c) 2001 MontaVista Software, Inc.
XScale cache_lock_init called
  Calling consistent alloc
  low_level_page initialized
  low_level_page @ 0xc8800000
    icache_lock_fn @ 0xc8800080
    dcache_lock_fn @ 0xc88000a0
    icache_unlock_fn @ 0xc8800098
    dcache_unlock_fn @ 0xc88000f0
Initializing TLB locking
TLB locking initialized
Dentry cache hash table entries: 16384 (order: 5, 131072 bytes)
Con0/Con1/
```



- ALTERA
- AMD
- ARC
- ARM*
- REESCALE
- HITACHI
- INFINEON
- INTEL
- MIPS*
- NEC
- NIOS
- PHILIPS
- RENESAS
- ST MICRO
- TEXAS INST.
- TRANSMETA
- VIA

Source-Level Kernel Debug

- Use JTAG connection for communications
- Direct connect to processor core
- “Halt Mode” debug
- Static view of processor and code
- Tool facility to program flash with kernel boot code and root file system

The screenshot displays a source-level kernel debugger interface. On the left, a code window shows the source code for `default_idle(void)` in `kernel/process.c`. Line 82, `local_irq_enable();`, is highlighted in yellow. On the right, a window titled "CP15 Registers" shows a list of registers and their values. The registers are grouped into categories: Current, User/System, Fast Interrupt, Interrupt, Supervisor, Abort, Undefined, CP0, CP14, and CP15. The CP15 registers are expanded to show various system registers like `ID_CODE`, `CACHE_TYPE`, `CONTROL`, `AUX_CONTROL`, `TRANSLATION_TABLE_BASE`, `DOMAIN_ACCESS_CONTROL`, `PREFETCH_FAULT_STATUS`, `FAULT_ADDRESS`, `INVAL_I_D_CACHE`, `INVAL_ICACHE`, `INVAL_ICACHE_SGL_ENTRY`, `INVAL_DCACHE`, `INVAL_DCACHE_SGL_ENT_MVA`, `CLEAN_DCACHE_SGL_ENT_MVA`, and `DRAIN_WRITE_BUFF`. The values for several registers are highlighted in yellow, including `CONTROL` (000039FF), `TRANSLATION_TABLE_BASE` (C22BC000), `DOMAIN_ACCESS_CONTROL` (0000001D), and `PREFETCH_FAULT_STATUS` (0000000F).

Application Debug

- **Linux applications are all mapped to the same address space**
- **Breakpoints can be set on dynamically located tasks**
- **“Run Mode” or “Task Mode” Debug**
 - Breakpoints stops the task, not the processor
- **Drill down from application level through the kernel to the board support package**
 - Combine Run mode with Halt mode debug

- ALTERA
- AMD
- ARC
- ARM*
- REESCALE
- HITACHI
- INFINEON
- INTEL
- MIPS*
- NEC
- NIOS
- PHILIPS
- RENESAS
- ST MICRO
- TEXAS INST.
- TRANSMETA
- VIA

Linux Process/Application Representation

- Tasks are shown in Operating systems Resources
- Identical to “ps” output
- Attach / debug a running process
- Start and debug a new process
- Debug multiple applications simultaneously

The image shows two windows from a debugger. The 'Target Console' window displays the output of the 'ps' command, showing a list of processes with columns for PID, TTY, Uid, Size, State, and Command. The 'Operating System Resources' window shows a table with columns for PID, PPID, TTY, Uid, Size, State, and Command, which is a more detailed view of the same process information.

PID	TTY	Uid	Size	State	Command
1		root	1828	S	init
2		root	0	S	[keventd]
3		root	0	R	[ksoftirqd_CPU0]
4		root	0	S	[kswapd]
5		root	0	S	[bdflush]
6		root	0	S	[kupdated]
7		root	0	S	[mtdblockd]
38	ttyS0	root	1900	S	/bin/sh --
39	ttyS1	root	1900	S	/bin/sh --
40	ttyS8	root	1908	S	/bin/sh --
41	ttyS9	root	1908	S	/bin/sh --
42	ttyS10	root	1900	S	/bin/sh --
43	ttyS11	root	1900	S	/bin/sh --
57	ttyS10	root	1392	S	/home/gdbserver /dev/ttyDCC5 /home/jbasic
58	ttyS10	root	1348	T	/home/jbasic
59	ttyS10	root	0	Z	[sh]
62	ttyS9	root	1888	R	ps

*And licensees

May. 15, 2005



There's always a Logic Solution



Linux Process Debugging

- Step through / set breakpoints in application code, kernel keeps running
- Examine / modify memory and process(or) registers
- Examine / modify application variables (global/local/stack)
- Full control of application code
- Highlighted Task List as the Focus Context
- Seamless transition in and out of kernel and process debug

The screenshot shows a debugger window with the following components:

- Code Editor:** Displays C code for a program starting at line 298. Line 302 is highlighted with a yellow arrow. The code includes initialization of variables like 'head', 'PC', 'pPC', 'SP', and 'trace_flag', followed by a 'while' loop for processing input lines.
- Operating System Resources:** A table listing system processes. The entry for PID 58 is highlighted in cyan, showing it is running the command '/home/jbasic'.

PID	PPID	TTY	Uid	Size	State	Command
1	0	?	root	1828k	Sleep	init
2	1	?	root	0k	Sleep	[keventd]
3	1	?	root	0k	Running	[ksftirqd_CPU0]
4	1	?	root	0k	Sleep	[kswapd]
5	1	?	root	0k	Sleep	[bdflush]
6	1	?	root	0k	Sleep	[kupdated]
7	1	?	root	0k	Sleep	[atdblockd]
38	1	tts/0	root	1900k	Sleep	/bin/sh
39	1	tts/1	root	1900k	Sleep	/bin/sh
40	1	ttyDCC0	root	1908k	Sleep	/bin/sh
41	1	ttyDCC1	root	1908k	Sleep	/bin/sh
42	1	ttyDCC2	root	1900k	Sleep	/bin/sh
43	1	ttyDCC3	root	1900k	Sleep	/bin/sh
57	1	ttyDCC2	root	1392k	Sleep	/home/gdbserver
58	57	ttyDCC2	root	120k	Trace	/home/jbasic
59	42	ttyDCC2	root	1816k	Sleep	sh
60	59	ttyDCC2	root	1444k	Running	/home/dccwrap

Practical use cases

- **“Headless” embedded systems**
 - no CRT, serial, ethernet or other console port available
 - DCC redirects console port over JTAG
- **No-stop debugging**
 - It is technically impossible or not desired to STOP the processor
 - Stop only the process you are debugging, keep your system running
- **Concurrent debug of kernel, processes and applications**
 - While debugging the application, you encounter a problem at the kernel
 - step down to kernel using the same debugger and session
- **Linux as development system**
 - Develop and debug your application from a linux host
 - Linux native version of the debugger available
- **Basically:**
 - Any Linux project on ARM / Xscale

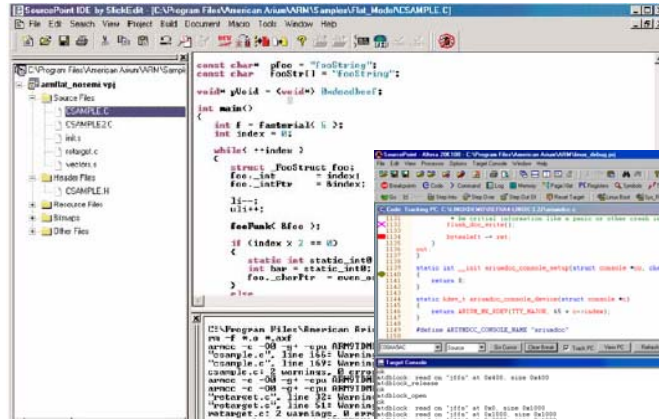
Logic Solutions
are available for:

- ALTERA
- AMD
- ARC
- ARM*
- REESCALE
- HITACHI
- INFINEON
- INTEL
- MIPS*
- NEC
- NIOS
- PHILIPS
- RENESAS
- ST MICRO
- TEXAS INST.
- TRANSMETA
- VIA

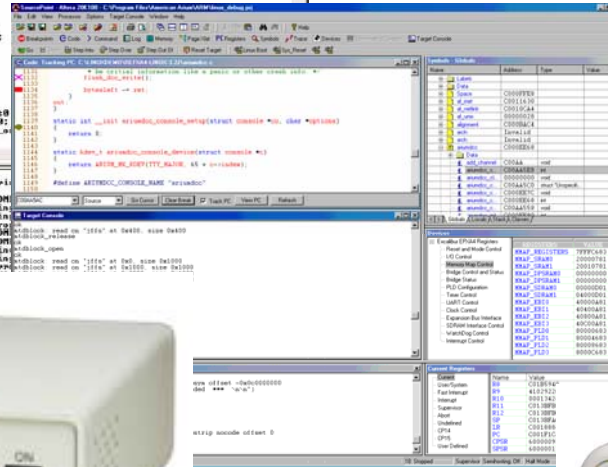
*And licensees
May. 15, 2005

Products

IDE



Multi Core Source Level Debugger



Emulator with JTAG Run Control Features



Emulators with JTAG and ETM Trace features



Development Boards and Kits



There's always a Logic Solution



ALTERA

AMD

ARC

ARM*

ARMADA
REESCALE

HITACHI

INFINEON

INTEL

MIPS*

NEC

NIOS

PHILIPS

RENESAS

ROHM
ST MICRO

TEXAS INST.

TRANSMETA

VIA

*And licensees

May. 15, 2005

More information

www.logic.nl

Let's go live.....



There's always a Logic Solution

