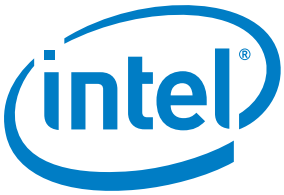


Using OpenOCD and Source Level Debug on Intel® Quark SoC X1000

Application Note

November 2013

Document Number: 528591, Revision: 0.8.0

Contents

Tables

Revision History

Date	Revision	Description
November 2013	0.8.0	Updated text to replace code name with official product name.
July 2013	0.6	Initial release.

1 Introduction

This document explains briefly how to use OpenOCD with Eclipse* or GDB for source level debugging of the Intel® Quark SoC X1000.

You may see references in the code to product codenames:

- Intel® Quark SoC X1000 (formerly codenamed Clanton)
- Intel® Quark Core (formerly codenamed Lakemont Core)

Note: This document is not a complete guide to source level debugging. Its purpose is to enable you to begin debugging the Linux* kernel on the Intel® Quark SoC X1000 at source level using OpenOCD with GDB or Eclipse.

For a complete set of supporting documentation, please visit the website for your specific JTAG hardware. The board has been tested with the following:

- Olimex* ARM-USB-OCD-H

  https://www.olimex.com/Products/ARM/JTAG/ARM-USB-OCD-H/
- TinCanTools* FLYSWATTER2

  http://www.tincantools.com/wiki/Compiling_OpenOCD

1.1 Terminology

Table 1. Terminology

Term	Description
Eclipse	An integrated development environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment.
GDB	GNU* Debugger is the standard debugger for the GNU operating system.
JTAG	Joint Test Action Group (JTAG) is the common name for the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture. Debuggers communicate on chips with JTAG to perform operations like single stepping and breakpointing.
OpenOCD	Free and Open On-Chip Debugger.
vmlinux	A statically linked executable file that contains the Linux kernel in one of the object file formats supported by Linux (such as ELF, COFF and a.out).

2 Prerequisites

Please refer to the OpenOCD section of the Intel® Quark SoC X1000 Board Support Package (BSP) Build Guide and complete the instructions before attempting the steps outlined in this document.

Required software:

- Linux* host system (running Eclipse/GDB/OpenOCD)
- Quark-patched OpenOCD
- GDB
- Eclipse (Juno tested) with CDT Plugin Installed (Main + Optional Features)
- Quark Kernel compiled with debug symbols

Required hardware:

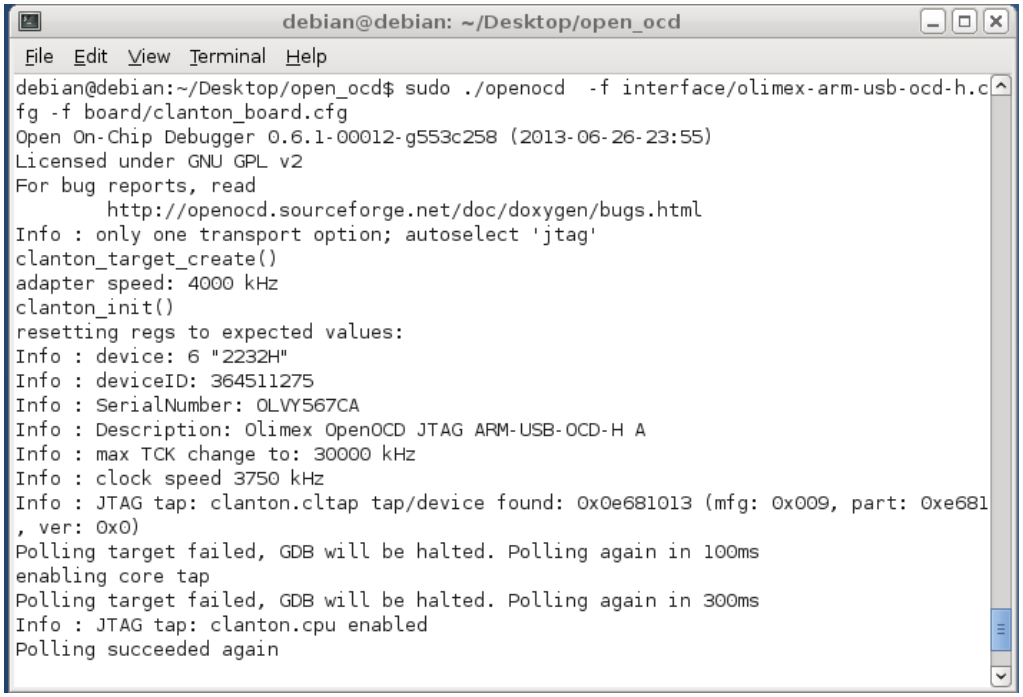
- OpenOCD supported JTAG debugger.
  For example:
  - Olimex* ARM-USB-OCD-H
  - TinCanTools* FLYSWATTER2

3 Debugging

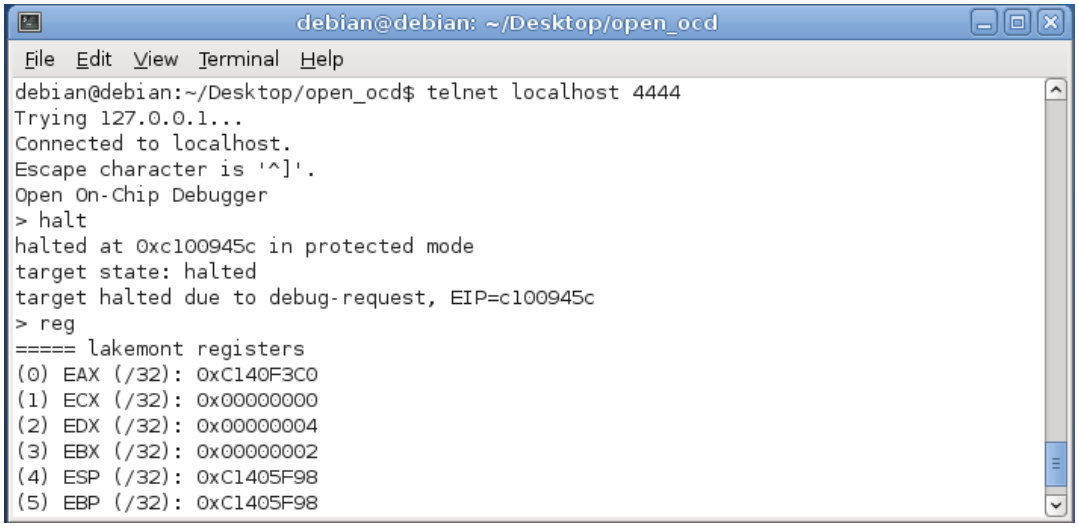
3.1 OpenOCD

The first step to enable source level debug is to connect your JTAG debugger to the board and run OpenOCD with the correct interface configuration file for your JTAG debugger. The example below uses an "olimex-arm-usb-ocd-h" JTAG debugger.

```
sudo ./openocd -f interface/olimex-arm-usb-ocd-h.cfg -f board/clanton_board.cfg
```



It is possible to use OpenOCD as a standalone tool for basic debugging. You can connect to the OpenOCD session using telnet and issue commands (this step is not required for source level debug). This can be seen in the following screenshot.



3.2 GDB

It is possible to perform source level debug using GDB by connecting to OpenOCD's internal GDB server. OpenOCD must be running as shown in the previous section.

Run GDB pointing to a debug symbol compiled Quark Kernel vmlinux file:

```
gdb /path/to/vmlinux
```

Connect to the OpenOCD internal GDB server and halt the board:

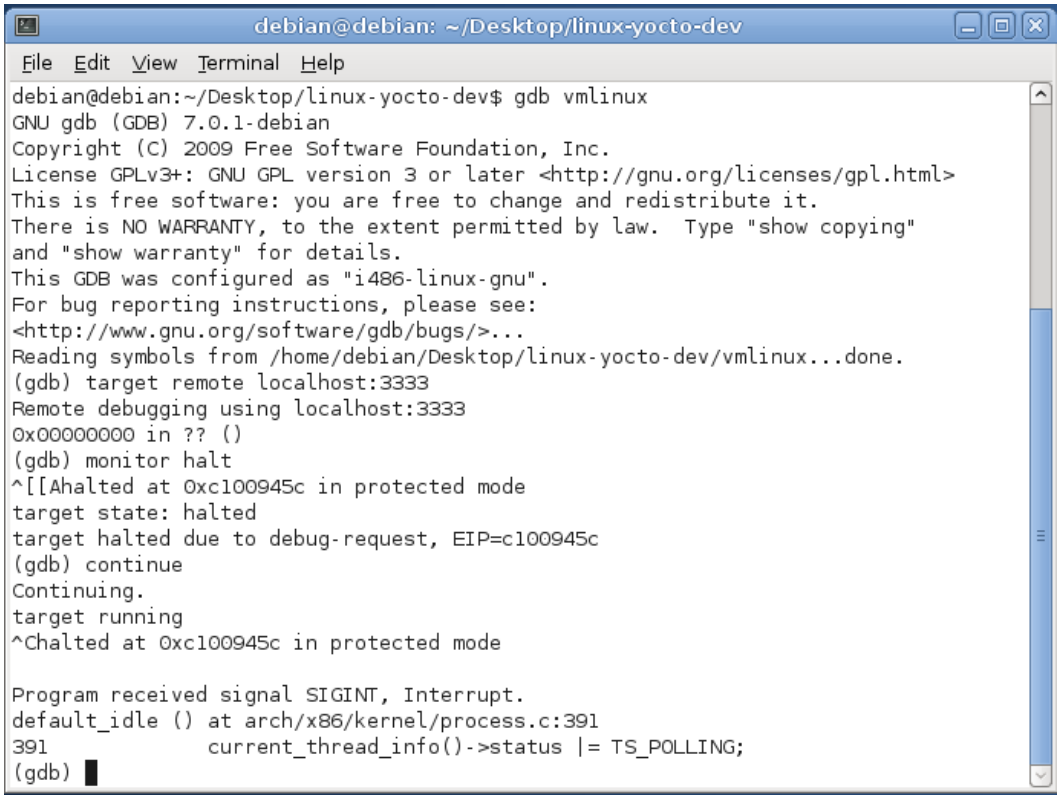
```
(gdb) target remote localhost:3333

monitor halt

continue

ctrl + c
```

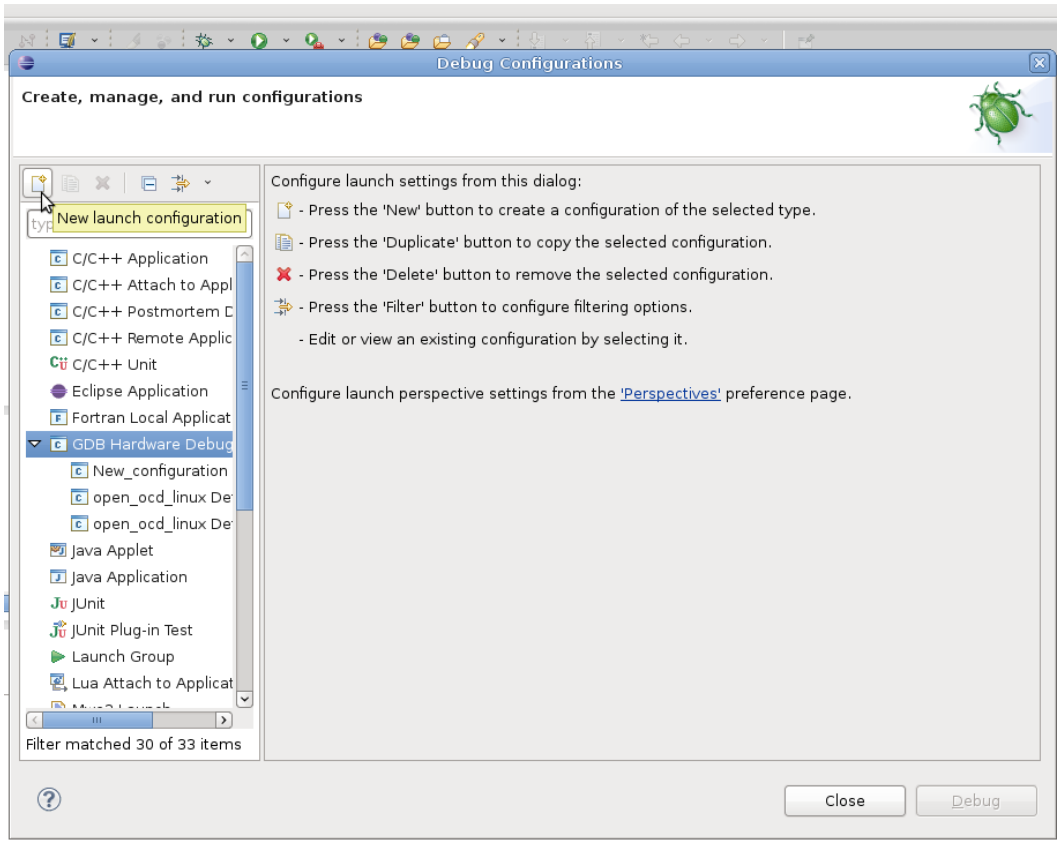
The screenshot below shows these steps in operation. After they are completed, the board is ready to be source level debugged using GDB.



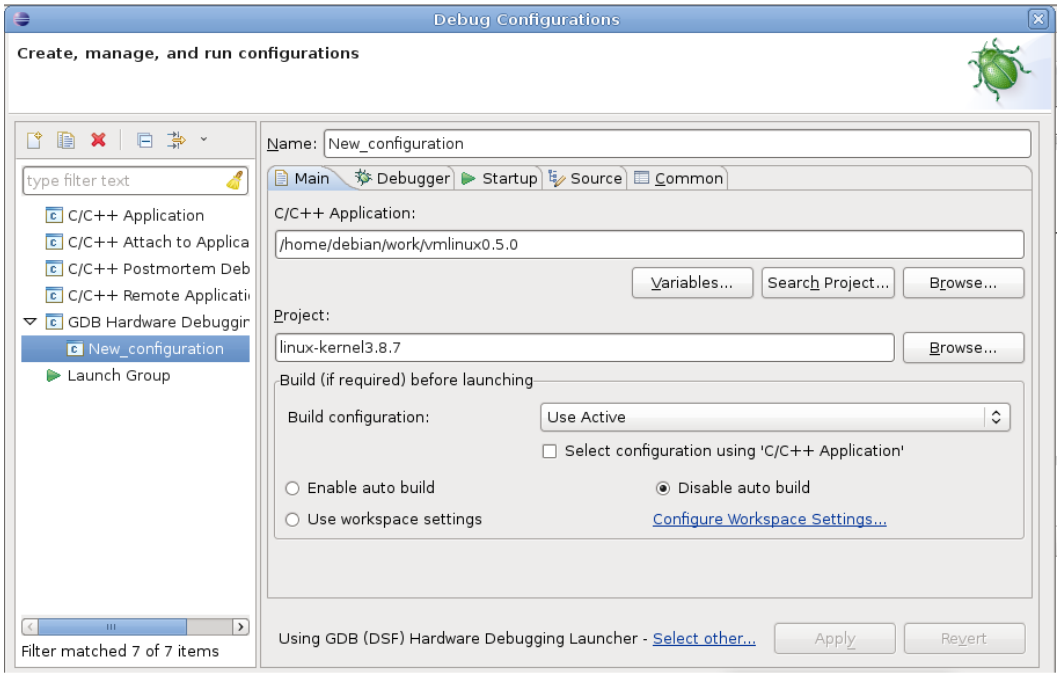
3.3 Eclipse

It is also possible to perform source level debug using Eclipse with the CDT GDB Hardware Debugger plug-in. The following configuration is required to enable source level debugging of the board in the Eclipse environment.

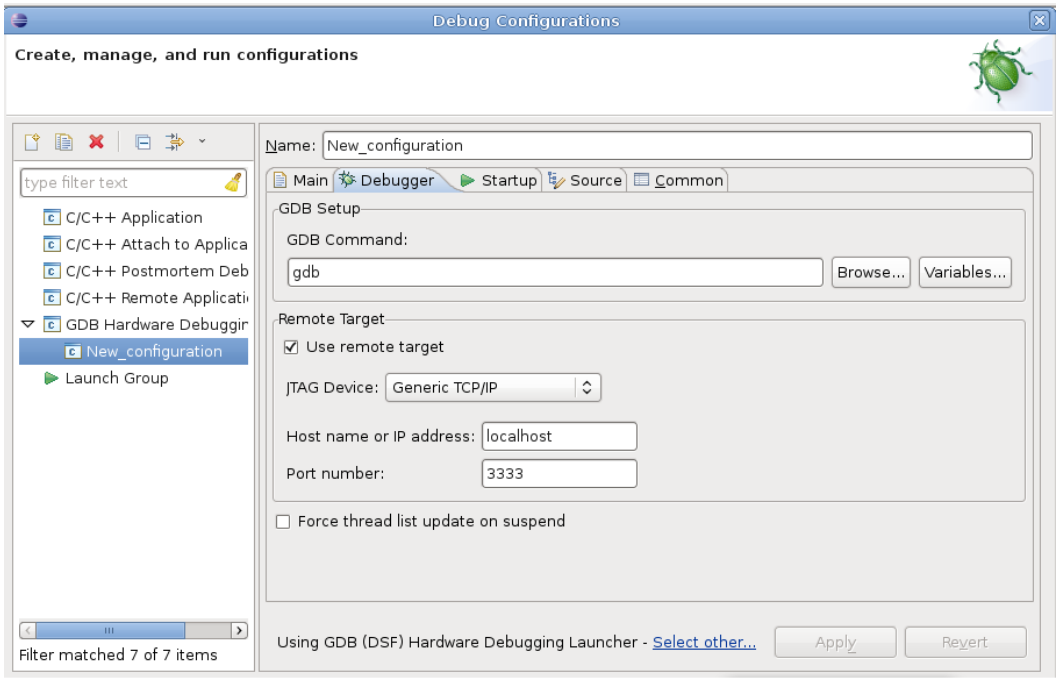
Go to the debug configurations menu, and add a new launch configuration under GDB hardware debugging, as shown below.



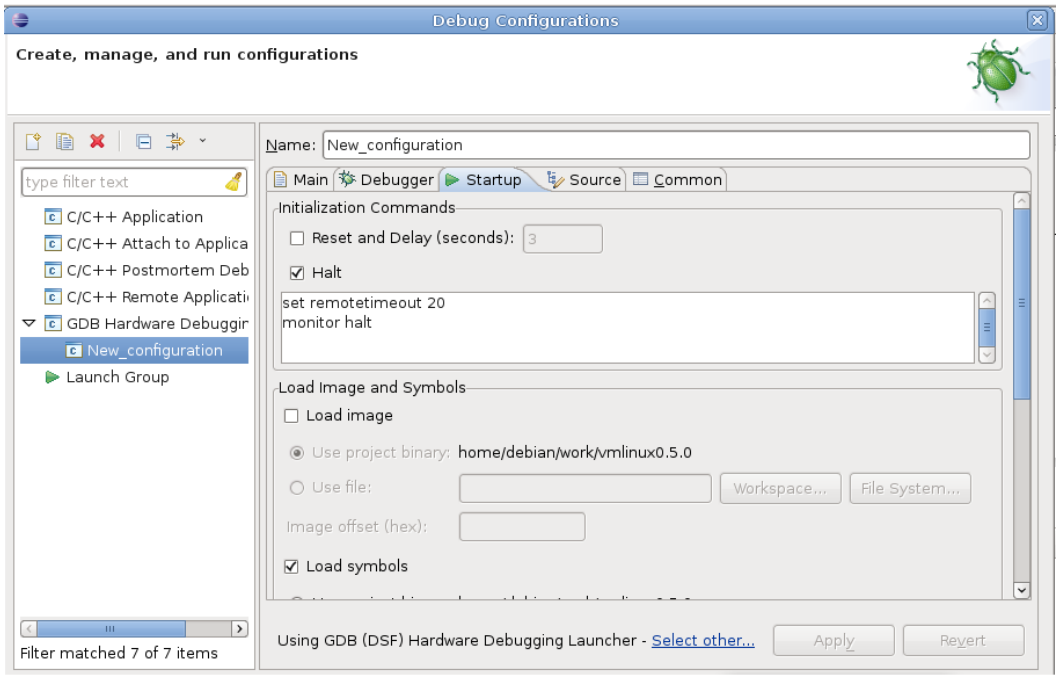
Set application to the debug symbol enabled vmlinux kernel file.



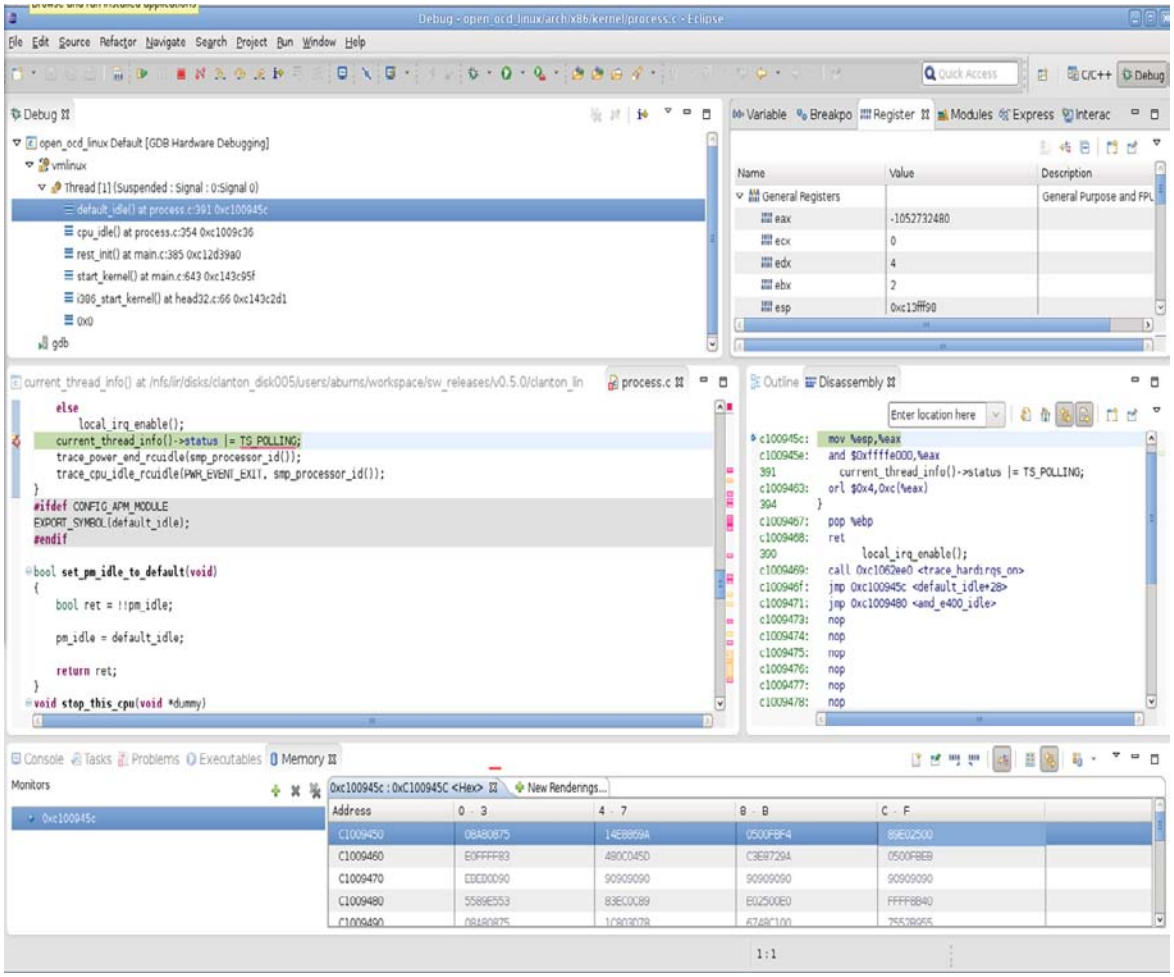
Enable **Use remote target** and set the host name and port number.



Select **Halt** and add the commands: **set remotetimeout 20** and **monitor halt**.



Eclipse is now set up to perform source level debug on the board as shown below.


Debug - open_ocd_linux/arch/x86/kernel/process.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
open_ocd_linux Default [GDB Hardware Debugging]
vmlinux
Thread [1] (Suspended : Signal : 0:Signal 0)
default_idle() at process.c:391 0xc100945c
cpu_idle() at process.c:354 0xc1009c36
rest_init() at main.c:385 0xc12d39a0
start_kernel() at main.c:643 0xc143c95f
i386_start_kernel() at head32.c:66 0xc143c2d1
0x0
gdb
Variable Breakpo Register Modules Express Interac
Name Value Description
General Registers General Purpose and FPU
eax -1052732480
ecx 0
edx 4
ebx 2
esp 0xc13fff90
process.c
else
local_irq_enable();
current_thread_info()->status |= TS_POLLING;
trace_power_end_rcuidle(smp_processor_id());
trace_cpu_idle_rcuidle(PWR_EVENT_EXIT, smp_processor_id());
}
#ifdef CONFIG_APM_MODULE
EXPORT_SYMBOL(default_idle);
#endif
bool set_pm_idle_to_default(void)
{
bool ret = !!pm_idle;
pm_idle = default_idle;
return ret;
}
void stop_this_cpu(void *dummy)
Outline Disassembly
Enter location here
c100945c: mov %esp,%eax
c100945e: and $0xffffe000,%eax
391 current_thread_info()->status |= TS_POLLING;
c1009463: orl $0x4,0xc(%eax)
394 }
c1009467: pop %ebp
c1009468: ret
390 local_irq_enable();
c1009469: call 0xc1062ee0 <trace_hardirqs_on>
c100946f: jmp 0xc100945c <default_idle+28>
c1009471: jmp 0xc1009480 <amd_e400_idle>
c1009473: nop
c1009474: nop
c1009475: nop
c1009476: nop
c1009477: nop
c1009478: nop
Console Tasks Problems Executables Memory
Monitors
0xc100945c
0xc100945c : 0xC100945C <Hex>
New Renderings...
Address 0 - 3 4 - 7 8 - B C - F
C1009450 08A80875 14E8869A 0500FBF4 89E02500
C1009460 E0FFFF83 480C045D C3E8729A 0500FBEB
C1009470 CDCD0090 90909090 90909090 90909090
C1009480 5589E553 83EC0C89 E02500E0 FFFF8B40
1:1

§