
# Linux for DesignWare® ARC Getting Started

Version 5977-009 November 2012

## Linux for DesignWare® ARC Getting Started

Synopsys, Inc.



Synopsys, Inc.
700 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

# Contents

*Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center for ARC Processor licensees with an active/valid support agreement.*

## Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services, which include downloading software, viewing Documentation on the Web, and entering a call to the Support Center.

To access SolvNet:

1. Go to the SolvNet Web page at http://solvnet.synopsys.com/.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)

If you need help using SolvNet, click SolvNet Help in the Support Resources section.

## Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

Open a call to your local support center from the Web by going to http://solvnet.synopsys.com/ (Synopsys user name and password required), then clicking “Enter a Call to the Support Center.”

Send an e-mail message to your local support center.

- E-mail support_center@synopsys.com from within North America.
- Find other local support center e-mail addresses at http://www.synopsys.com/support/support_ctr.

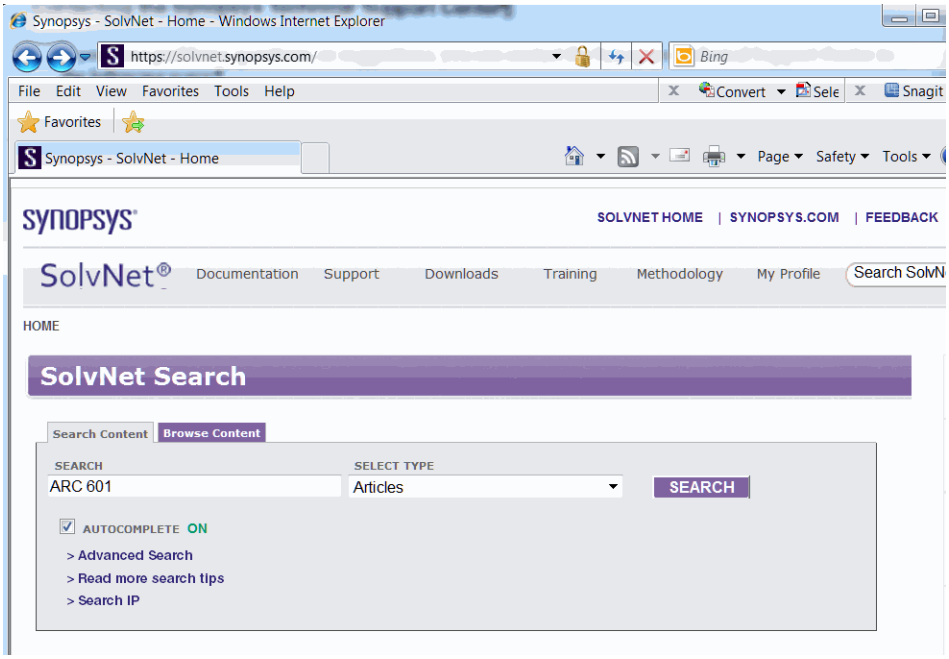
Telephone your local support center.

- Call (800) 245-8005 from within the continental United States.
- Call (650) 584-4200 from Canada.
- Find other local support center telephone numbers at http://www.synopsys.com/support/support_ctr.

# Accessing ARC Application Notes

SolvNet articles include useful application notes to help you do your job using DesignWare ARC products. To find application notes for your DesignWare ARC product, do the following:

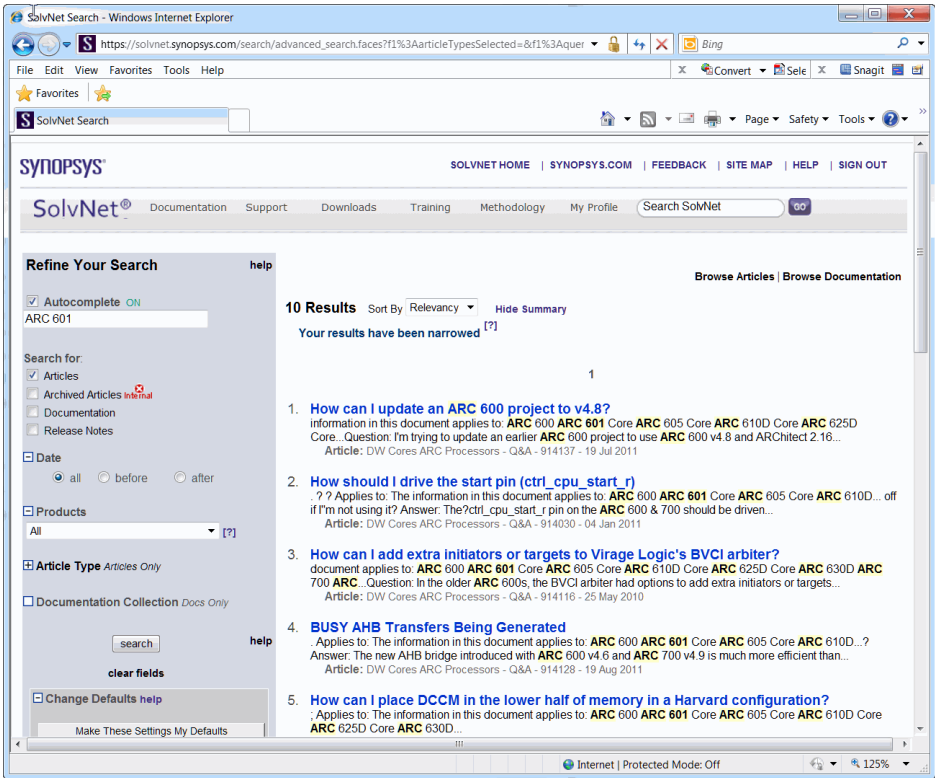
1. Go to the SolvNet Web page at http://solvnet.synopsys.com/.




2. Enter the name of your product in the **Search** field.
3. Select **Articles** from the **Select Type** drop-down list.
4. Click **Search**.

The figure below shows example results.

**Note** Application notes are updated frequently, so results can vary.



# 1 Introduction

*This document describes how to get up and running quickly with Linux.*

## Typographical Conventions

This document uses the following typographical conventions:

| Convention | Meaning | Examples |
|---|---|---|
| **bold** | • functions<br>• predefined classes<br>• values to be entered literally<br>• GUI elements | • **get_callback()**<br>• the **simple_initiator_socket** class<br>• **template**<int init, int target><br>• Select **Save All**. |
| italics | • text to be replaced with your own values<br>• Titles of documents | • *InstallationDir*/MetaWare<br>• **sc_time** *p* //period<br>• *ARC xCAM Release Notes* |
| monospace | code in general | ls.push_back("SHMEM"); |

# 2
# Building Linux for DesignWare ARC

*This section describes how to build Linux for DesignWare ARC.*

## Installation and Build Overview

A minimal Linux environment requires a kernel with a root file system containing the init program, shell, inetd and utilities such as `mount`, `ls`, etc.

The ARC Linux distribution consists of the following components (sources):

- U-Boot bootloader
- Linux kernel,
- `strace`
- LMBench

It also includes a pre-built binary initramfs with enough userland apps (Busybox based) to quickly get a kernel up and running.

All the above components, including the ARC GNU Toolchain are available from github at:

https://github.com/foss-for-synopsys-dwc-arc-processors/

## Prerequisites

The ARC GNU tools must be built from sources and installed before you attempt kernel builds. Build and Installation instructions can be found in the readme files that accompany ARC GNU. To do so, perform the next command:

```
$ git clone git://github.com/foss-for-synopsys-dwc-arc-processors/toolchain.git
```

Ensure the `PATH` environment variable points to both the `bin` folder of the installed toolchain.

**Note**

Installation organization of older versions of the ARC GNU Toolchain (such as 4.2.1) was slightly different and required addition of two entries to `PATH`: `INSTALL/<efl32_install_dir>/bin` and `INSTALL/<uclibc_install_dir>/bin`

For version 4.4 of the ARC GNU toolchain as it is required for ARC Linux 3.2, you must add only `INSTALL/bin` to the `PATH` environment variable.

# Unpack the Initramfs

Download the initramfs tarball from github and extract it:

```
$wget https://raw.github.com/foss-for-synopsys-dwc-arc-processors/arc_initramfs_archives/master/arc_initramfs_10_2012_dyn_dev.tgz
$ tar -zxvf arc_initramfs_10_2012_dyn_dev.tgz
```

The pre-built initramfs comes preloaded with the following:

- Startup and configuration scripts: `/etc/init.d/rcS`, `/etc/inittab`, `/etc/inetd.conf`
- Target system libraries: `libc.so`, `libm.so`, `ld.so` …
- Cross-compiled BusyBox binary, which provides commonly used Unix utilities such as `shell`, `mount`, `cat`, `ls`, etc., including the `init` program. The initramfs includes both statically and dynamically linked versions with the dynamically linked version as the default (using symbolic links).
- Other cross-compiled utilities such as `strace`

**Note**

Unlike prior versions of ARC initramfs, the current version no longer requires you to create static device nodes, avoiding the requirement of having root or sudo privileges on the host.

The `/dev` folder is now dynamically created as a virtual file system, using the kernel's `CONFIG_DEVTMPFS` (CONFIG_HOTPLUG dependent) option. This folder is mounted in the root fileystem at boot-up by the `/init` script

# Building the Kernel

1. Configure the kernel:

```
$ git clone git://github.com/foss-for-synopsys-dwc-arc-processors/linux.git
$ cd linux
$ git checkout stable-arc-3.2

For an ISS Simulator, do:
$ make ARCH=arc fpga_defconfig
$ make ARCH=arc menuconfig
        Select:    CONFIG_ARC_CACHE_LINE_SHIFT=5
```

```
          CONFIG_ARC_ISS_VIRT_PERIP
          Unselect:  CONFIG_ARC_BOARD_ANGEL4
For an Angel4 board, do:
$ make ARCH=arc fpga_defconfig

For a ML509 board, do:
$ make ARCH=arc fpga_defconfig
$ make ARCH=arc menuconfig
           Select:    CONFIG_ARC_CACHE_LINE_SHIFT=5
                       CONFIG_ARC_PLAT_CLK=70000000
                      CONFIG_ARC_BOARD_ML509
           Unselect:  CONFIG_ARC_BOARD_ANGEL4

For a HAPS51 board, do:
$ make ARCH=arc haps51_defconfig
```

The `defconfig` make target generates a kernel build configuration file `.config`, based on the `defconfig` file residing in `arch/arc/configs/`

You can further modify the default configuration using `menuconfig` or `nconfig` targets.

2. Adjust the `initramfs` folder path in `.config`. (Optional)

   Kernel build needs the exact path (absolute or relative) of the untarred `initramfs` folder. This information is specified in kernel `.config` file with option `CONFIG_INITRAMFS_SOURCE`. By default its value is `../arc_initramfs`. This option works out-of-the-box if both the kernel and `initramfs` folders are at the same level in the directory hierarchy. If not, change it to reflect the path on your system.

**Note**

If this `initramfs` path is a symbolic link, make sure a trailing '/' is appended:

`CONFIG_INITRAMFS_SOURCE="../arc_initramfs/"`

3. Build the kernel with the following command:

   ```
   $ make ARCH=arc
   ```

   It is no longer necessary to specify `bootpImage` target as in previous ARC Linux releases.

This procedure generates the vmlinux binary, which is ready to run on the specific platform it is built for. See also Running ARC Linux on a HAPS51 Development System.

# 3
# Running ARC Linux

*This section provides instructions on how to run Linux for DesignWare ARC on the MetaWare instruction-set simulator or one of the supported ARC development systems.*

## Running ARC Linux on the MetaWare Instruction-Set Simulator

ARC Linux runs on the MetaWare debugger instruction-set simulator (ISS). The ISS not only emulates the ARC ISA (and CPU micro-architecture such as caches and MMU); it can also emulate some of the essential peripherals such as UART, frame buffer, and Ethernet (Windows only). This ability makes the simulated Linux experience very realistic, allowing you to type shell commands on the console, start new programs, and see your output in real time. It also provides a very convenient environment to debug the kernel itself with features such as instruction tracing and profiling.

The kernel binary built for real FPGA peripherals also works in the ISS, as corresponding device drivers detect where they are executing and abort the device initialization if executing on the ISS instead of real hardware.

To emulate an ARC 700 version 4.10 or later system, launch the debugger with the following command line:

```
mdb -toggle=include_local_symbols=1 -toggle=deadbeef=1 \
    -profile -sim -a7 -on=enable_exceptions –mmuv3 -Xmpy \
    -icache=16384,32,2,o -dcache=16384,32,4,o \
    -core4 -prop=dcache_version=3 -prop=icache_version=3 \
    -simextp=SCDIR/bin/termsim,term_base=0xc0fc1000,\
    term_port=1,term_int_vector=5,term_tcpport=+0 ./vmlinux
```

## Running ARC Linux on an ML509 Development System

### Overview

The ML509 system is an FPGA prototyping platform from Xilinx which is supported as a platform for ARC Linux.

The ML509 system is fully supported through the ARC FPGA peripherals IP library and FPGA RDFs.

The platform is also supported in ARC Linux with selected device drivers.

For more information, see the following website:

http://www.xilinx.com/univ/xupv5-lx110t.htm

## Connecting to the ML509

An ML509 system needs two different connections to the host PC.

- Programming the FPGA bit file: A Xilinx platform Cable or Digilent USB-to-JTAG cable (connected to the board's J1 connector) is needed together with Xilinx iMPACT installed on the host.

  The Windows driver for the Xilinx platform cable is compatible with Digilent, so they can be used interchangibly. iMPACT can be installed with Xilink ISE (~5GB) or LabTools (~1GB)

- Debugging the target: Ashling Opella-XD Probe (connected to Board's J51 connector via flying leads and to the host with USB connector). The connection of the six leads is shown below.

*Figure 1: ML509Connectors*

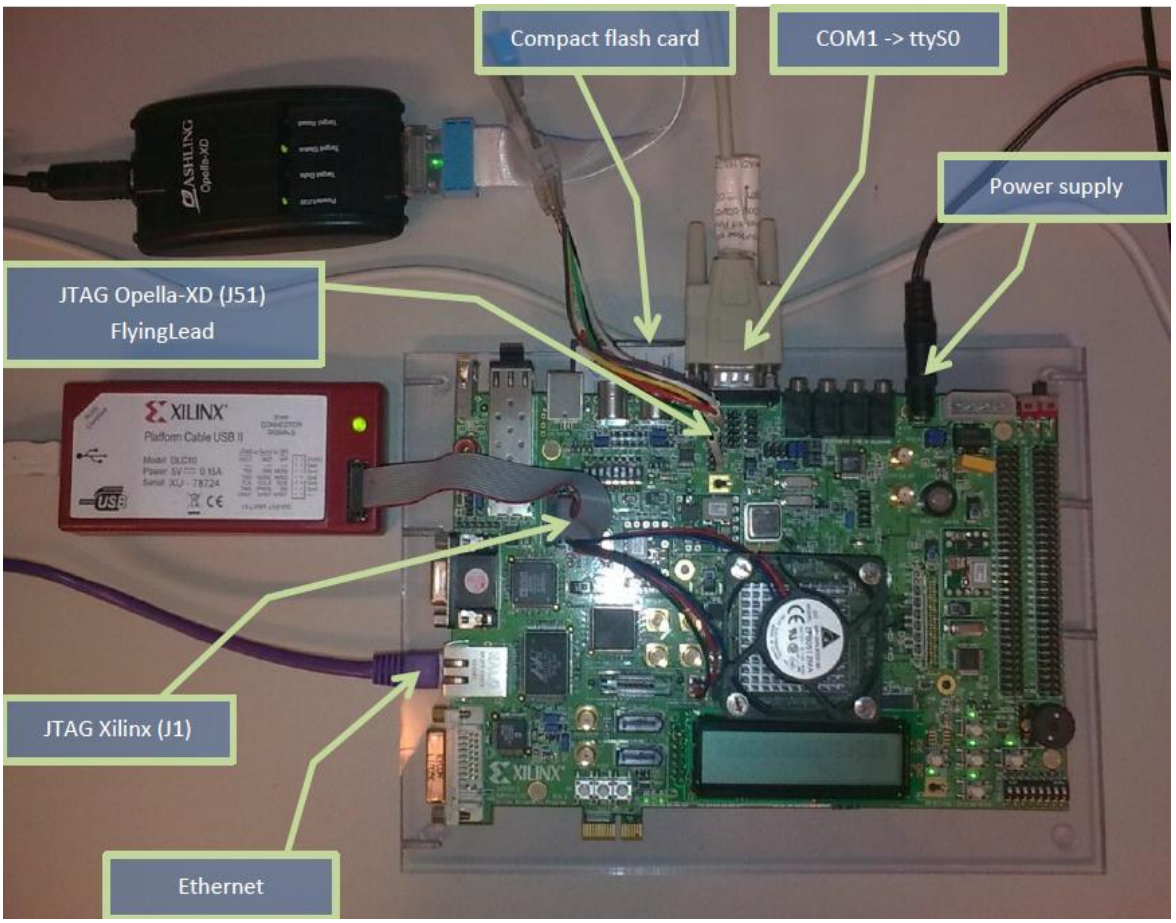
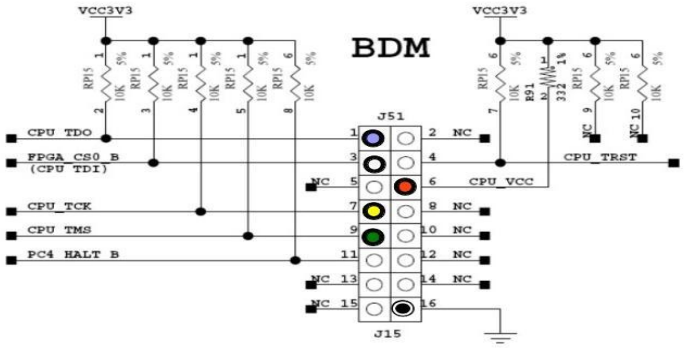


*Figure 2: Opella-XD FlyingLead Connections*




## Downloading the Image and Running the Executable on ML509

You can use the MetaWare debugger in combination with the Opella-XD Probe to download the binary vmlinux kernel to the target and run it.

After you connect the cables and program the FPGA bitfile using iMPACT, issue the following command (from the Windows host command prompt)

```
mdb -gui -OKN -toggle=include_local_symbols=1 -profile -hard \
  -dll=c:/AshlingOpellaXDforARC/opxdarc -prop=jtag_frequency=8MHz
./vmlinux
```

Click **Run** when the GUI loads.

# Running ARC Linux on a HAPS51 Development System

## Overview

The HAPS51 Development System from Synopsys is also supported by ARC Linux.

The integrated system for use with ARC Linux comes complete with integrated daughter cards to provide the following interfaces with corresponding DW IP controllers:

- Ethernet interface (requires add-on Ethernet PHY HAPS module)
- DVI interface (requires add-on DVI PHY HAPS module)
- SD card (requires add-on multi-Interface HAPS module)
- Buttons and LEDs (requires add-on multi-Interface HAPS module)

Support is provided in ARC Linux as a platform called plat-haps51 with a corresponding `haps51_defconfig` kernel configuration file.

The HAPS51 Development System must be purchased separately. Contact your Synopsys sales representative for more information.

## Connecting to HAPS51

The HAPS51 system comes with an extensive documentation set on how to configure and use the system, including extensions.

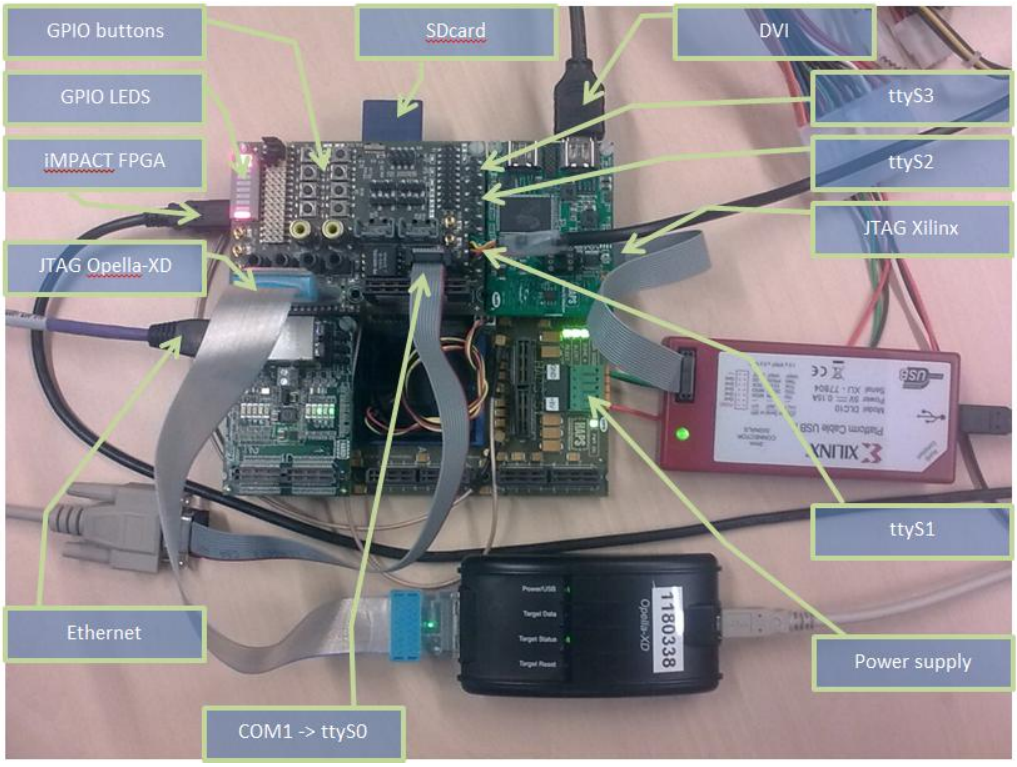



*Figure 3: HAPS-51 Development System with ARC Linux Interface Modules*

## Downloading image and running on HAPS51

Use the MetaWare debugger with the OpellaXD Probe to download the compiled kernel image (vmlinux) to the target system. Use the debugger to load the image into memory, then click the debugger **Run** button. Linux boot to a Busybox prompt (Console = ttyS0, 115200n8).

The haps51_defconfig module includes the majority of peripherals found on the board. This includes support for dw_uart, dw_intc (external Interrupt controller), dw_gpio (for accessing LEDS and buttons on the board), STMAC ethernet controller, and SDCard.

Note that with `CONFIG_HOTPLUG` enabled in the kernel and `mdev` set in the `init` scripts, the SD card is autodetected on boot or when you plug it in the slot. However, it needs to be mounted manually:

```
[ARCLinux]$ ls -l /dev/mm*
[ARCLinux]$ mount /dev/mmcblk0p1 /mnt/sdcardP1
[ARCLinux]$ ls -l /mnt/sdcardP1
```

# 4
# Debugging ARC Linux

*This section describes debugging options in ARC Linux.*

## ARC Linux Kernel Debugging using KGDB

While user-space programs can be debugged with regular GDB (in combination with gdbserver), this is not the case for debugging the kernel. gdbserver is a user-space program itself and cannot control the kernel. KGDB, Kernel GDB, solves this by acting as a gdbserver that is inside the kernel.

### Configuring the Kernel for KGDB

Your kernel configuration needs to have the following options set:

```
CONFIG_EXPERIMENTAL
CONFIG_KGDB
CONFIG_KGDB_SERIAL_CONSOLE
```

#### Kernel command line

Use the kgdboc option on the kernel command line to tell KGDB which serial port to use.

Examples:

One serial port:

```
console=ttyS0,115200n8 kgdboc=ttyS0,115200
```

Two serial ports:

```
console=ttyS0,115200n8 kgdboc=ttyS1,115200
```

These examples assume you want to attach gdb to the kernel at a later stage. Alternatively, you can add the `kgdbwait` option to the command line. With `kgdbwait`, the kernel waits for a debugger to attach at boot time. In the case of two serial ports, the kernel command line looks like the following:

```
console=ttyS0,115200n8 kgdboc=ttyS1,115200 kgdbwait
```

### Connect from GDB

After the kernel is set up, you can start the debugging session. To connect to your target using a serial connection, you need to have a development PC with UART that runs GDB and a terminal program.

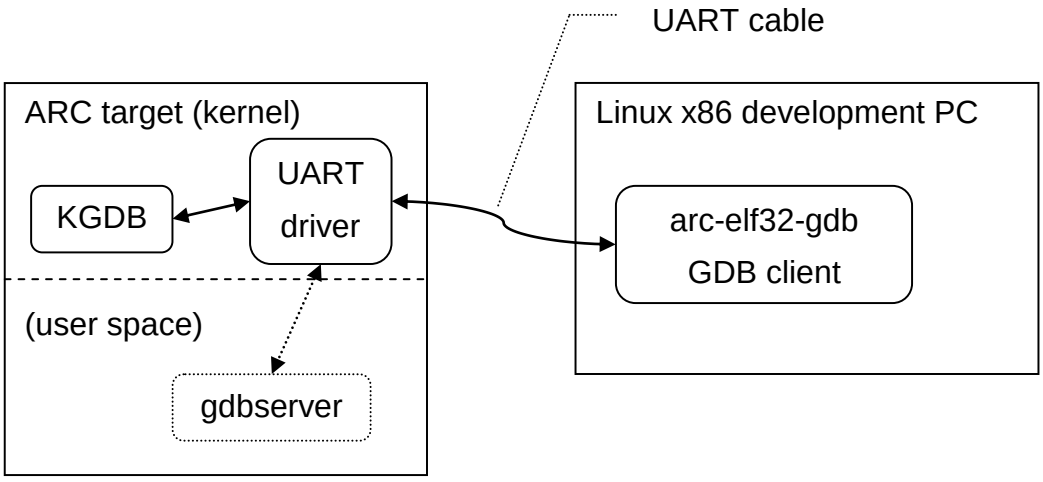



Figure 4: Connections for Cross-Debugging with KGDB

### Stop the Kernel

First, stop the kernel on the target using a SysRq trigger. To do so, send a 'remote break command using your terminal program, followed by the character 'g'.

Example using minicom: Ctrl-a, f, g

Example using Tera Term: Alt-b, g

You must also stop the kernel if you have two UARTs, even though one of the two UARTs is dedicated to KGDB.

### Connect GDB

After stopping the kernel, connect GDB:

```
$ arc-elf32-gdb vmlinux
(gdb) set remotebaud 115200
(gdb) target remote /dev/ttyUSB0
```

You are then connected to the target and can use GDB like any other program. For instance, you can set a breakpoint now using `b` *identifier* and then continue kernel execution again using `c`.

# ARC Linux Kernel Debugging Using the MetaWare Debugger

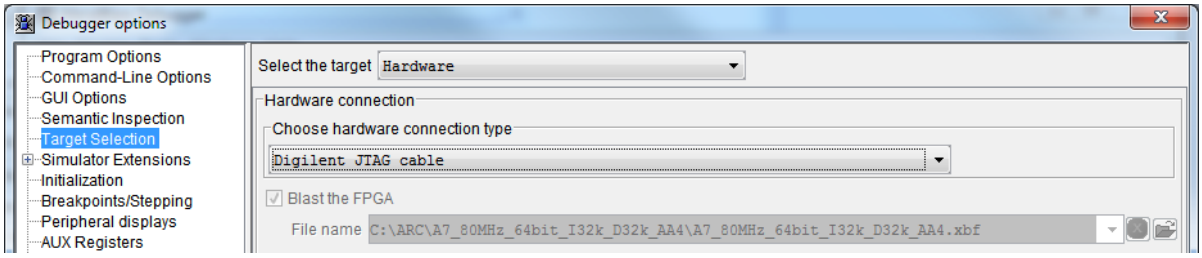
You can debug the kernel with the MetaWare Debugger (mdb), using either a simulation or a JTAG connection to hardware.

## Connect to Your Target

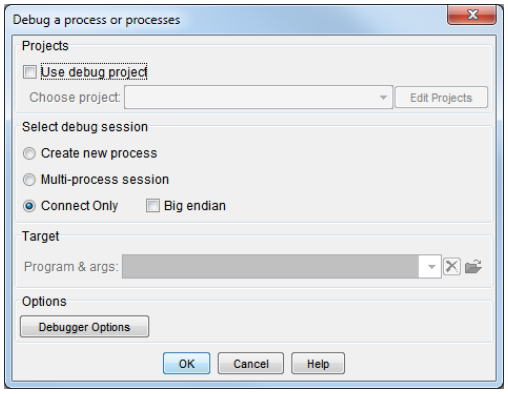
If you are loading the kernel using the MetaWare debugger, connect by following the steps described in Chapter 3: Running ARC Linux.

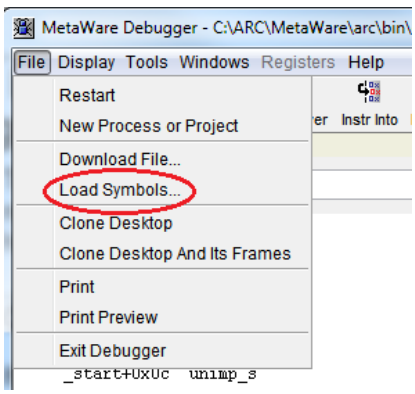
If you are using U-Boot to start your kernel, follow this additional step:

- Start the MetaWare debugger.
- In **Debugger Options > Target Selection** configure your target and JTAG probe.



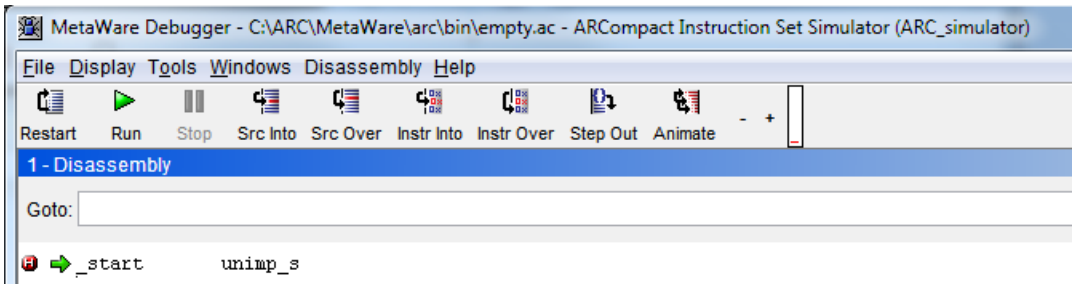

- Select **Connect Only** and click **OK**.
- Select **File > Load Symbols** menu and point to your kernel image so that the debugger can resolve the symbols when you step through the code.



### Debugging

After connecting, you can start debugging. The GUI provides buttons for start, stop and various options for stepping. Breakpoints can be set or cleared by clicking in front of the instruction.




For more details about using the MetaWare Debugger, see the *DesignWare MetaWare Debugger User's Guide for ARC*.

# Debugging User Applications with Gdbserver and GDB

The gdbserver/gdb combination provides the ability to debug applications running on ARC Linux. gdbserver runs on the target just as another application, taking the to-be-debugged application as an argument (it can also attach to an existing process). gdb runs on the host, connecting to target gdbserver for actual debugging.

By default gdbserver is not available in the initramfs. So it needs to be either copied over to target or accessed from NFS-mounted host.

Target:

```
# /mnt/host/GNU_INSTALL/target-bin/gdbserver 10.0.0.2:567  app_to_debug
```

Host

```
$ arc-linux-uclibc-gdb app_to_debug
(gdb) target remote 10.0.0.6:567
(gdb) break main
(gdb) continue
```

See the following SolvNet article for more details:
https://solvnet.synopsys.com/retrieve/034472.html

# 5
# Using U-Boot with ARC Linux

*U-Boot is a universal boot loader written and maintained by Denx Software. This section describes how to install and start using U-Boot with Linux for DesignWare® ARC®.*

*More information on U-Boot can be found at*http://www.denx.de/wiki/U-Boot/WebHome

## Building U-Boot for ARC

1. Download U-Boot from github:

```
$ git clone git://github.com/foss-for-synopsys-dwc-arc-processors/u-
boot.git
$ cd u-boot
$ git checkout arc-v2012.07-stable
```

2. Make target configuration

```
For HAPS51 board, do:
$ make ARCH=arc snps_haps51_config
```

3. Build u-boot with the following command:

```
$ make ARCH=arc
```

This procedure generates the U-Boot" binary to run on ARC.

## Supported Boot Methods

This section provides examples of each of the boot methods supported by U-Boot for ARC Linux.

### Downloading and Running the Image on HAPS51

To download the compiled U-Boot image to the target system, use the MetaWare debugger with the OpellaXD Probe. Click the **Run** button in the debugger GUI when the image is loaded into memory by the debugger. U-Boot boots to the prompt (Console = ttyS0, 115200n8).

## Flash U-Boot into HAPS51

**Note**

Use caution when performing this procedure. If the procedure is not performed as listed below, the run-time system can be damaged.

To flash an image into the HAPS51 onboard flash memory, perform the following steps in the order listed below:

1. Download U-Boot into memory using the MetaWare debugger.
2. Click **Run** in the debugger and verify U-Boot is running as expected.
3. Click **Stop** in the debugger.
4. Use the debugger to load `u-boot.bin` at memory location 0x84000000.
5. Click **Run** again.
6. In the U-Boot console, type the following command:

```
haps51# protect off all
```

7. In the U-Boot console type the following command:

```
haps51# erase 0x10000000 +0x40000
```

8. In the U-Boot console type the following command:

```
haps51# cp.b 0x84000000 0x10000000 0x40000
```

# 6
# Appendix

## Troubleshooting

Following are some of the common pitfalls you might encounter when trying to start the system for the first time.

**Problem #1: Linux Panics During Early Boot**
The CPU configuration (real or simulated) doesn't match the configuration the kernel is built for. You see message like

```
MMU pg size != PAGE_SIZE
```
*or*
```
Cache H/W doesn't match kernel Config
```

**Causes**

- Mismatch in cache-line size, MMU page size, MMU version, or other parameters. Kernel configuration includes a driver, which tries to initialize the non-existent hardware.
- Clock frequency of the board as configured with the MetaWare debugger using `-prop=gclk=80` does not match kernel configuration option `CONFIG_ARC700_CLK`.

**Problem #2: Garbled Output in Serial Console**

**Causes**

- The baud rate of the UART does not match kernel configuration option (based on the platform you are building for)

**Problem #3: Linux Fails to Boot to the Shell Prompt (Userland Coming up Failure)**

**Scenarios**

- `Warning: unable to open an initial console.`

  This message indicates the absence of device nodes in your root filesystem (initramfs-based system)

- `Kernel panic - not syncing: Attempted to kill init!`

  The init task (the first user task to start) is crashing, possibly because the BusyBox binary is faulty.

- `ABI mismatch - you need newer toolchain`

  The 3.2 kernel has a new syscall ABI, which requires an updated toolchain (patched GNU 4.4 tools). The kernel can detect if the user code is built using the correct tools or not.

**Causes**

- The init task (first user task to start) is crashing, possibly because the BusyBox binary is faulty.

# Customizing the Initramfs

## Building Upstream BusyBox for ARC

The ARC *initramfs* includes a pre-built, binary-only BusyBox. You can also build your own version from upstream sources. Any recent upstream version of BusyBox builds out-of-the-box for ARC Linux.

To build a custom BusyBox:

1. Download the upstream sources and cross compile:

```
$ git clone git://busybox.net/busybox.git
$ make CROSS_COMPILE=arc-linux-uclibc- && make install
```

2. Copy the resulting binary into initramfs:

```
$ cp busybox_unstripped ../arc_initramfs/bin/busybox
```

3. Rebuild the kernel with the updated initramfs.

## Updating the System Libraries in Initramfs

```
$ rm -rf arc_initramfs/lib/*
$ cp -rvfd UR_GNU_INSTALL/arc-linux-uclibc/lib/*.so* arc_initramfs/lib/.
$ rm -rf arc_initramfs/lib/*stdc*
```

**Note**

It is important to remove the libstdc++ shared library, as it is too big for an initramfs-based system. If you need libstdc++ for your application, consider alternate root file systems such as NFS ROOT or an in-target persistent root file system, depending on availability of a flash or IDE disk. See Development Using NFS-Based Workflow.

# Development Using NFS-Based Workflow

While initramfs serves as a good starting point for a working ARC Linux system, it quickly becomes a major inconvenience in an iterative develop-test-debug workflow, as it requires a development, copy-to-ramfs, kernel-rebuild, and kernel-reboot cycle for every change.

Application development must not require a kernel rebuild or especially a reboot. Ideally, users want to develop on the target itself, but given the embedded nature of ARC Linux

deployment and the lack of target resources (processing power, applications), the next-best option is to develop on the host and be able to access that development environment on target in real time.

This method is supported through NFS protocol-based remote access (by default, the NFS client is already enabled in the ARC Linux kernel).

Details might vary by distribution, but if you have NFS-shared one or more host folders, you can mount the same folders on the target as follows:

```
# mount -o nolock -t nfs 10.0.0.2:/home/host_folder /target
```

As a next step, the initramfs can be eliminated altogether by having an NFS-based root file system. To do so, enable `CONFIG_ROOT_NFS` and `CONFIG_IP_PNP` in kernel `config`.

Configure `cmdline` with the details of host and directory that will serve the file system:

```
CONFIG_CMDLINE="root=/dev/nfs nfsroot=10.0.0.2:/arc_rootfs,nolock ip=dhcp"
```