Digi Embedded Yocto 1.6

First Steps Guide

Revision history—90001423

Revision	Date	Description
E	July, 2015	Updated to Digi Embedded Yocto 1.6.6
F	October, 2015	Updated to Digi Embedded Yocto 1.6.7
G	November, 2015	Updated to Digi Embedded Yocto 1.6.8
H	May, 2016	Added support for a new CC6 variant, a patch and minor fixes in the Linux kernel.
J	September, 2017	Updated to Digi Embedded Yocto 1.6.9 - rebranded to Madcap Flare.

Trademarks and copyright

Disclaimers

Information in this document is subject to change without notice and does not represent a commitment on the part of Digi International. Digi provides this document “as is,” without warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Digi may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time.

Warranty

To view product warranty information, go to the following website:

www.digi.com/howtobuy/terms

Send comments

Documentation feedback: To provide feedback on this document, send your comments to techcomm@digi.com.

Customer support

Digi Technical Support: Digi offers multiple technical support plans and service packages to help our customers get the most out of their Digi product. For information on Technical Support plans and pricing, contact us at +1 952.912.3444 or visit us at www.digi.com/support.

Support portal login: www.digi.com/support/eservice

Contents

Overview of Digi Embedded Yocto 1.6

Platforms supported 5

Developing with Digi Embedded Yocto

Setting up your workstation 7
Installing Digi Embedded Yocto 7
Updating Digi Embedded Yocto 7
Creating projects 7
Updating existing projects 8
Configuring projects 8
Building images 8
Customizing Digi Embedded Yocto 8
- Configuring default network settings 8
- Adding packages to your target image 9
- Adding features to your target image 9
- Advanced customization 10

Using the Eclipse application development environment

Installing Eclipse 12
Configuring the Eclipse Yocto plugin 12
Configuring to use a standalone toolchain 14
- Cross compiler options 14

Configuring to use a system derived toolchain 15
- Cross compiler options 15

Preparing your DEY project for application development 15
Creating a project 16
Running an example application 16
Debugging an example application 17

Overview of Digi Embedded Yocto 1.6

This guide is intended for embedded systems developers with previous experience with the Yocto Project who want to use Yocto with Digi embedded modules. This document will guide you through the installation and setup of Digi Embedded Yocto, a Linux distribution built upon the Yocto project.

Platforms supported 5

Platforms supported

Digi Embedded Yocto 1.6 supports the following Digi embedded platforms:

- ConnectCore 6 SBC
- ConnectCard for i.MX28

It contains two Yocto layers:

1. **meta-digi-arm**: This layer is based on meta-fsl-arm and contains the BSP customizations for Digi's supported platforms.
2. **meta-digi-dey**: This layer adds two new target images to Yocto:
   - **dey-image-minimal**, a minimal busybox command line based image.
   - **dey-image-graphical**, a graphical SATO (a GNOME mobile graphical environment) based image.

Developing with Digi Embedded Yocto

There are two distinct development workflows to consider on Digi Embedded Yocto.

System development: When the Digi Embedded Yocto installation generates target images, that is the U-Boot bootloader, Linux kernel and root file system.

Application development: When the developer is working on a C/C++ user space application that needs to be run and debugged, which will be integrated in the final images at a later stage.

Setting up your workstation 7
Installing Digi Embedded Yocto 7
Updating Digi Embedded Yocto 7
Creating projects 7
Updating existing projects 8
Configuring projects 8
Building images 8
Customizing Digi Embedded Yocto 8

Setting up your workstation

Your PC workstation must be set up correctly in order to use Digi Embedded Yocto. You will find instructions regarding supported operating systems and specific OS setup in the Yocto online Quick Start Guide. Please follow the instructions corresponding to your operating system before continuing.

Installing Digi Embedded Yocto

The repo tool installs Yocto. Download repo to a directory within your path and add execution permissions.

```
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > /usr/
local/bin/repo
chmod a+x /usr/local/bin/repo
```

It is assumed that you are running repo from /usr/local/dey-1.6 (make sure you have user write permissions).

```
cd /usr/local/dey-1.6
repo init -u https://github.com/digidotcom/dey-manifest.git -b refs/tags/1.6.9
repo sync -j4 --no-repo-verify
Note that the process above will either install Digi Embedded Yocto for the first time, or updat
```

If you are updating from a previous major release like dey-1.4, you must install in a new directory and re-create your projects.

Updating Digi Embedded Yocto

Digi performs extensive validation testing on the released Digi Embedded Yocto tags (dey-1.6.x). However, the Yocto community continually updates its stable branches with fixes. If you want to keep up to date with these changes, sync your installation with the head of the release branch as follows:

```
repo init -u https://github.com/digidotcom/dey-manifest.git -b daisy
repo sync -j4 --no-repo-verify
```

Creating projects

The available platforms in the meta-digi-arm layer are:

- ConnectCore 6 SBC, ccimx6sbc
- ConnectCard for i.MX28, ccardimx28js

To initialize the project and environment, we use the mkproject.sh script. For example, for the ccimx6sbc, you should do the following:

```
mkdir -p $HOME/workspace/ccimx6sbc
cd $HOME/workspace/ccimx6sbc
source /usr/local/dey-1.6/mkproject.sh -p ccimx6sbc
```

This will initialize the project with a conf directory and two configuration files:

- **bblayers.conf:** The available layers are configured here.
- **local.conf:** Local configuration variables affecting only this project are customized here.

The mkproject.sh script sets the environment for the build in the current running terminal. It also creates a dey-setup-environment script in the project's root folder. This script can be safely rerun over existing projects to set up the build environment on a new terminal.

Updating existing projects

When updating your installation of Digi Embedded Yocto, you will need to erase the tmp and sstate-cache directories from existing projects and build them from scratch. Leaving the directories intact may result in problems in the build and the final images.

Configuring projects

Edit the variable MACHINE_VARIANT at file conf/local.conf to configure the project to match your module variant and hardware components. The conf/local.conf file contains a table describing the different module variants.

The following example selects a ConnectCore 6 variant that has Wi-Fi, Bluetooth, a Quad/Dual CPU, and 1024 MiB of RAM:

```
MACHINE_VARIANT = "wbq1023"
```

Note It is important to select the MACHINE_VARIANT that matches your hardware variant. Otherwise, you may experience the following problems: system images may lack support for some hardware, and try to load support for non-existing hardware, or prevent the system from booting.

Building images

To build the images, use one of the following commands depending on whether you want a graphical or command line based file system.

```
  bitbake dey-image-minimal
or
  bitbake dey-image-graphical
```

Note that dey-image-minimal does not support the X11 window system, so in order to build this image you need to add the following line to the project's conf/local.conf:

```
  DISTRO_FEATURES_remove = "x11"
```

Generated images, including a bootable SD card image, are in the <project>/tmp/deploy/images/ccimx6sbc folder.

You can program these images to the module's internal eMMC, a microSD card, or perform a network boot using the U-Boot bootloader that was preloaded on the hardware. For more information about booting the operating system, download the U-Boot Customizations Reference Guide (document number 90001422) from the Documentation section of the Digi Resources page.

Customizing Digi Embedded Yocto

Configuring default network settings

DEY allows you to configure the default network settings for your target's image which will appear in the /etc/network/interfaces file. This is done in your project's local.conf file using the following variables to define your default network settings:

```
  ETHn_STATIC_IP = "<ip address>"
  ETHn_STATIC_NETMASK = "<netmask>"
  ETHn_STATIC_GATEWAY = "<gateway>"
```

Where n is 0 or 1 depending on the number of available Ethernet interfaces in your platform.

```
WLAN0_STATIC_IP = "<ip address>"
WLAN0_STATIC_NETMASK = "<netmask>"
```

To configure dynamic IPs, you can use the following configuration:

```
ETHn_MODE = "dhcp"
WLAN0_MODE = "dhcp"
```

By default, the DEY target images are configured with static IPs.

```
ETH0_STATIC_IP       ?= "192.168.42.30”
ETH1_STATIC_IP       ?= "192.168.44.30" (if applicable)
WLAN0_STATIC_IP      ?= "192.168.43.30" (if applicable)
```

Adding packages to your target image

If you want to add a package to your image, for example strace, add the following to your project's conf/local.conf file:

```
IMAGE_INSTALL_append = " strace"
```

Note Include the space before the word strace.

Adding features to your target image

Apart from the standard Yocto features, the DEY images introduce some new features that allow for a certain degree of customization ease.

```
EXTRA_IMAGE_FEATURES = "<feature-name>"
```

Most of the features are automatically selected based on the selected machine. For example, a machine that supports ALSA will include the dey-audio feature automatically.

A list of new DEY features follows:

- **dey-audio**: Adds audio support to a platform.
- **dey-gstreamer**: Adds the gstreamer framework to a platform.
- **dey-network**: Adds network applications and tools. You can configure some of the network applications in your conf/local.conf by doing:

  ```
  VIRTUAL-RUNTIME_ftp-server = "vsftpd"
  VIRTUAL-RUNTIME_http-server = "cherokee"
  VIRTUAL-RUNTIME_network-utils = "net-tools"
  VIRTUAL-RUNTIME_snmp-manager = "net-snmp-server"
  VIRTUAL-RUNTIME_ssh_server = "dropbear"
  ```

- **dey-wireless:** Includes wireless applications and drivers.
- **dey-bluetooth:** Adds Bluetooth support.
- **dey-debug:** Adds DEY debugging applications asmemwatch and fbtest.
- **dey-examples:** Adds DEY example applications.
- **dey-qt:** Adds QT support for dey-image-graphical. This feature is added by default in dey-image-graphical images.

Advanced customization

When further customization is needed, the suggested way to customize Yocto for your specific need is to create a new layer. An example of this could be called meta-custom. You can use it directly above meta-digi-arm, or you can use it to customize the Digi demo images already provided in meta-digi-dey.

Your meta-custom layer can:

- Include new recipes, for example to add new applications to Yocto that compile from source, or to add new files to the file system.
- Modify existing recipes in any other layer by adding a .bbappend recipe. This is also designed to modify existing files in the file system.
- Create a new target image.
- Create new machine configurations.
- Provide default kernel configurations or configuration fragments for your new platforms.

Using the Eclipse application development environment

Installing Eclipse 12
Configuring the Eclipse Yocto plugin 12
Configuring to use a standalone toolchain 14
Configuring to use a system derived toolchain 15
Preparing your DEY project for application development 15
Creating a project 16
Running an example application 16
Debugging an example application 17

Installing Eclipse

The recommended Eclipse version to use with Digi Embedded Yocto 1.6 is the Kepler 4.3 standard edition. Other versions might work but could lead to unexpected problems.

Download the Kepler tarball and unpack it on a location of your choice.

```
tar -xzvf eclipse-standard-kepler-SR2-linux-gtk-x86_64.tar.gz
```

Start the Eclipse IDE, and check for updates from the Help menu. Then select Install New Software from the Help pull-down menu, selecting **Kepler-download.eclipse.org/releases/kepler**.

Install all the updates and the following Eclipse components:

Linux tools

- LTTng - Linux Tracing Toolkit

Programming languages

- Autotools Support for CDT
- C/C++ Development Tools

Mobile and device development

- C/C++ Remote Launch
- Remote System Explorer End-user Runtime
- Remote System Explorer User Actions
- Target Management Terminal
- TCF Remote System Explorer add-in
- TCF Target Explorer

Install the Eclipse Yocto plugin by adding the following repository downloads.yoctoproject.org/releases/eclipse-plugin/1.6/kepler.

If you experience user interface problems with Eclipse on newer versions of Kubuntu, try changing the GTK theme to something other than Oxygen.

Configuring the Eclipse Yocto plugin

The Eclipse Yocto Plugin can be used with either standalone toolchains or system derived toolchains - that is, the toolchain built as part of your Digi Embedded Yocto project.

You can install a standalone toolchain in one of two ways:

- Install the Application Development Toolkit (ADT) installer.
- Install a pre-built Digi Embedded Yocto toolchain.

The ADT installer can be downloaded from the Yocto project web site. It will install a generic ARM toolchain and sysroot and will allow you to debug using an external soft floating point ARMv5 or compatible target.

The Digi Embedded Yocto toolchain installs target-specific ARM toolchains and a sysroot that matches dey-image-graphical. It can be downloaded from Digi's server and can be used to develop both console or graphical applications. This is the recommended toolchain to install.

To configure the ADT:

1. Go to **Windows > Preferences > Yocto Project ADT**.
2. Download the Digi Embedded Yocto prebuilt toolchain to a directory of your choice and run it.

   For example, for the ConnectCore 6 you would do as follows:

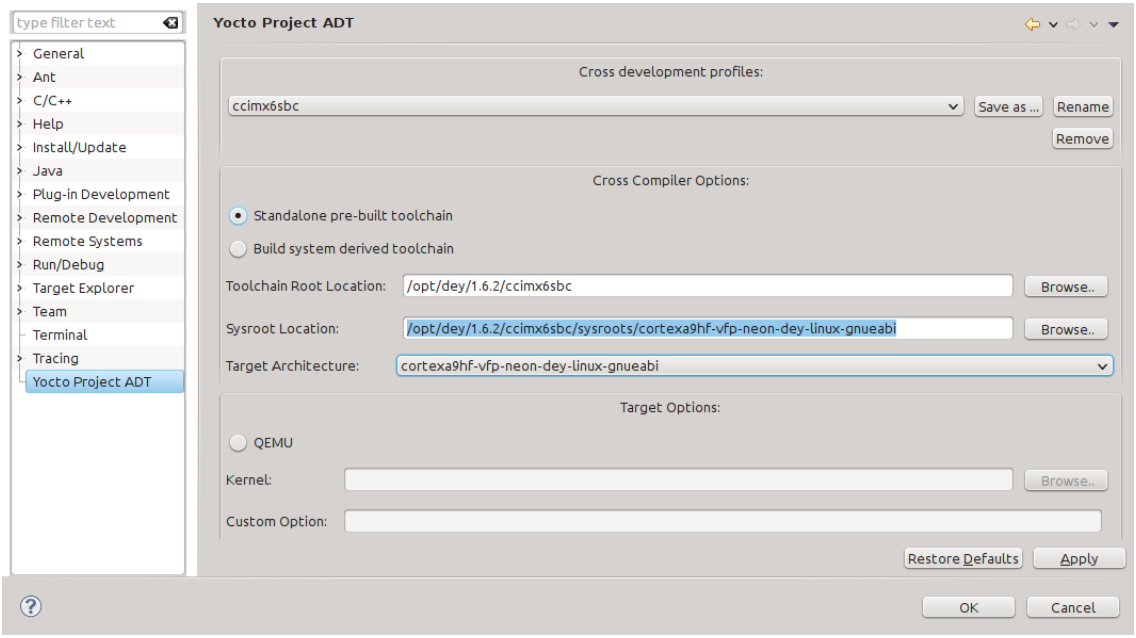
```
./dey-eglibc-x86_64-dey-image-graphical-cortexa9hf-vfp-neon-toolchain-1.6.9.sh
```

It is a good idea to create a platform specific directory if you are going to use toolchains for different architectures.

Configuring to use a standalone toolchain

Cross compiler options

Standalone pre-built toolchain	
Toolchain root location	`/opt/dey/1.6.9/ccimx6sbc`
Sysroot location	`/opt/dey/1.6.9/ccimx6sbc/sysroots/cortexa9hf-vfp-neon-dey-linux-gnueabi`
Target architecture	`cortexa9hf-vfp-neon-dey-linux-gnueabi`

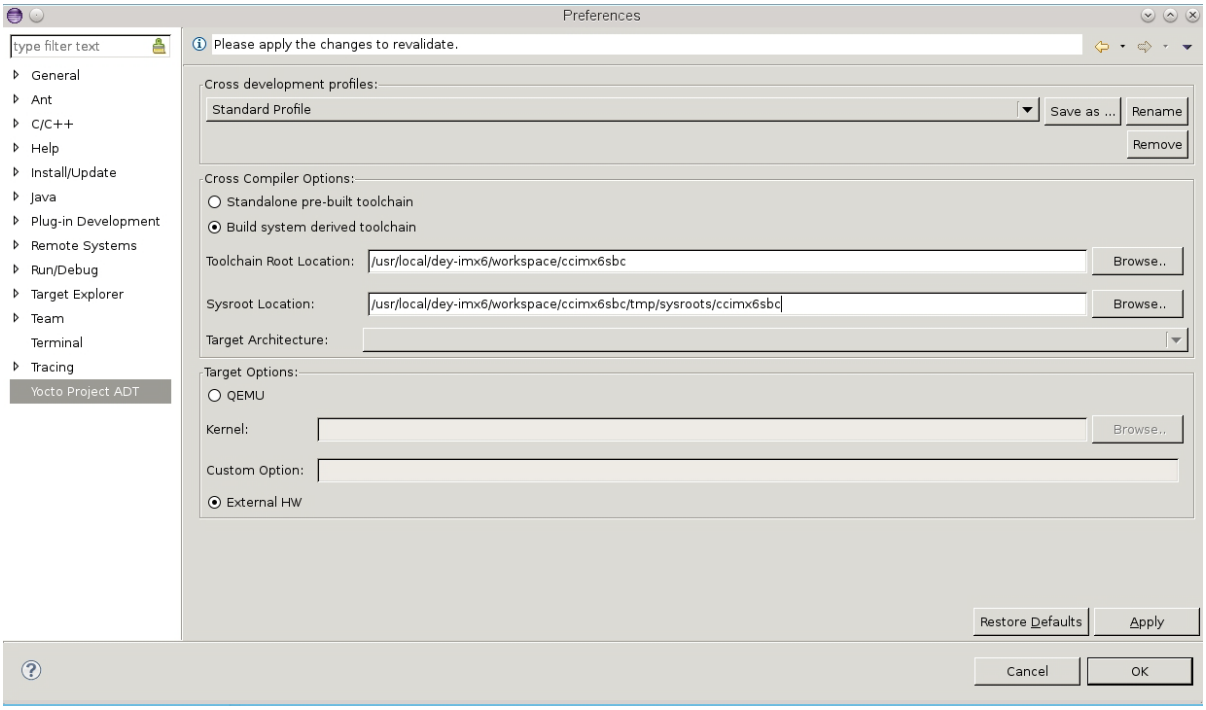


Configuring to use a system derived toolchain

This will use the toolchain compiled in your DEY project.

Cross compiler options

Build system derived toolchain	
Toolchain root location	~/workspace/<project>
Sysroot location	~/workspace/<project>/tmp/sysroots/$MACHINE



Preparing your DEY project for application development

To use the build system derived toolchain run the following in your project:

```
bitbake meta-ide-support
```

To build a toolchain installer for the standalone toolchain configuration run the following in your project:

```
bitbake dey-image-minimal -c populate_sdk
or
bitbake dey-image-graphical -c populate_sdk
```

To be able to perform debugging on a remote target you need to run a root file system with the tcf-agent daemon running. By default, the sdk images (dey-image-minimal-sdk and dey-image-graphical-sdk) include the tcf-agent. You can also add the agent to your DEY project by adding the following features to your local.conf:

```
EXTRA_IMAGE_FEATURES = "tools-debug eclipse-debug"
```

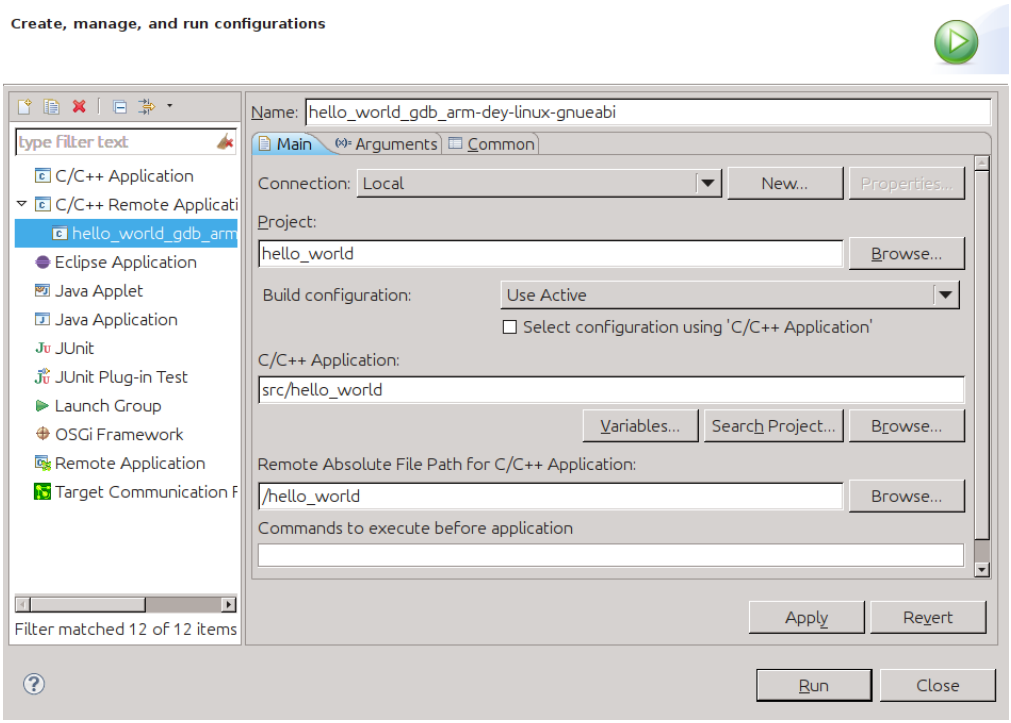
Creating a project

File > New > C project > Yocto Project ADT autotools Project

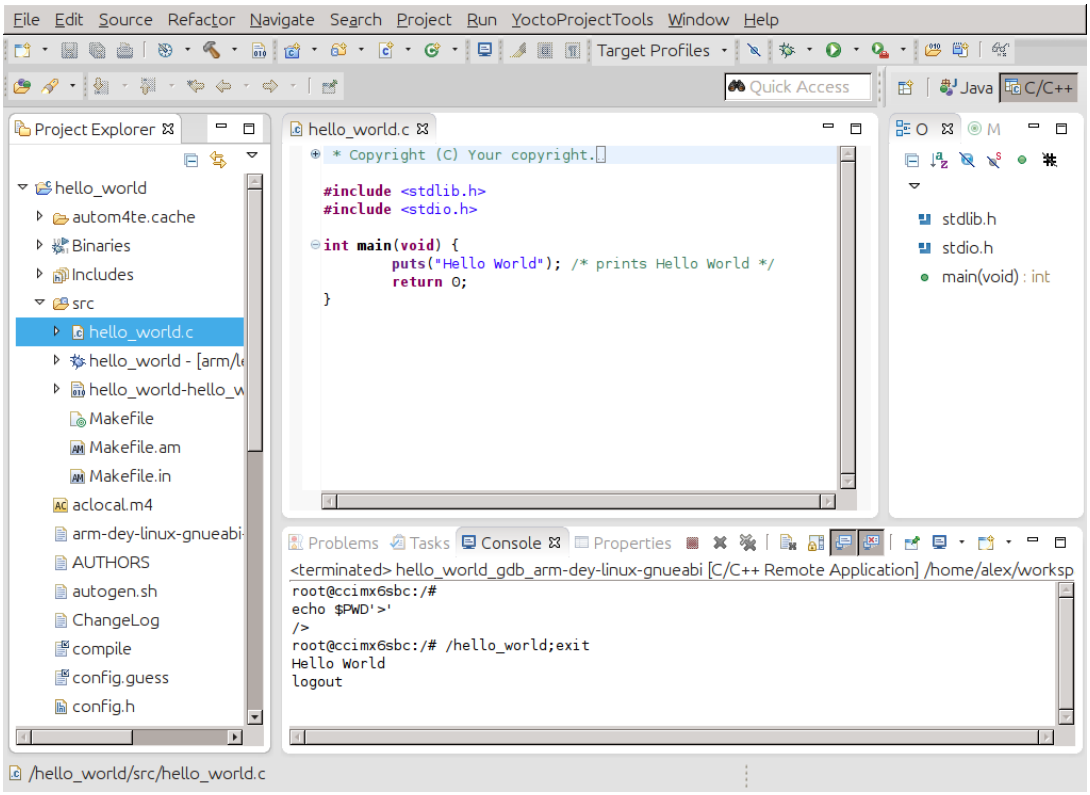
In the following sections, we will use a Hello World ANSI C Autotools sample project, called hello_world.

Running an example application

1. Build the application. **Right click project > Build Project**
2. Configure the remote session as follows:
   - Open a remote terminal and run ifconfig to get its IP address.
   - In **Run Configurations**, extend the C/C++ remote application to show a hello_world_gdb_arm-dey-linux-gnueabi remote debug launcher already created for you.
   - If there is no launcher preconfigured, create a new C/C++ remote application with a TCF (Target Communication Framework).
   - Create a new connection to the target's IP address.
   - Fill in the **Remote Absolute File Path for C/C++ Application** with the absolute path in the remote target to copy the application to, for example /hello_world.



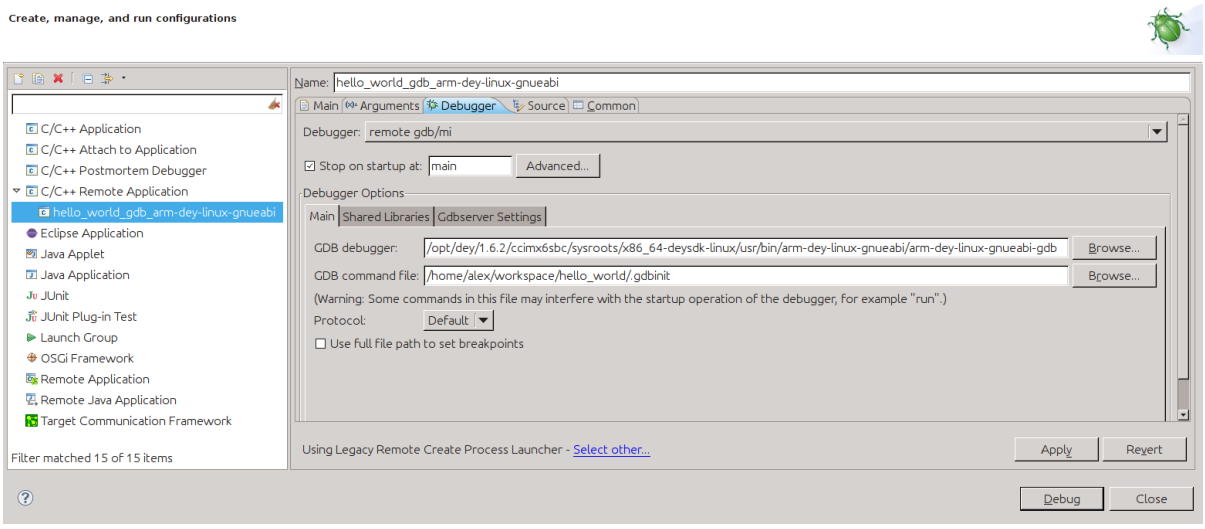
Then run the configuration. When prompted, the username and password is root/root. You will see the program output on the console view.



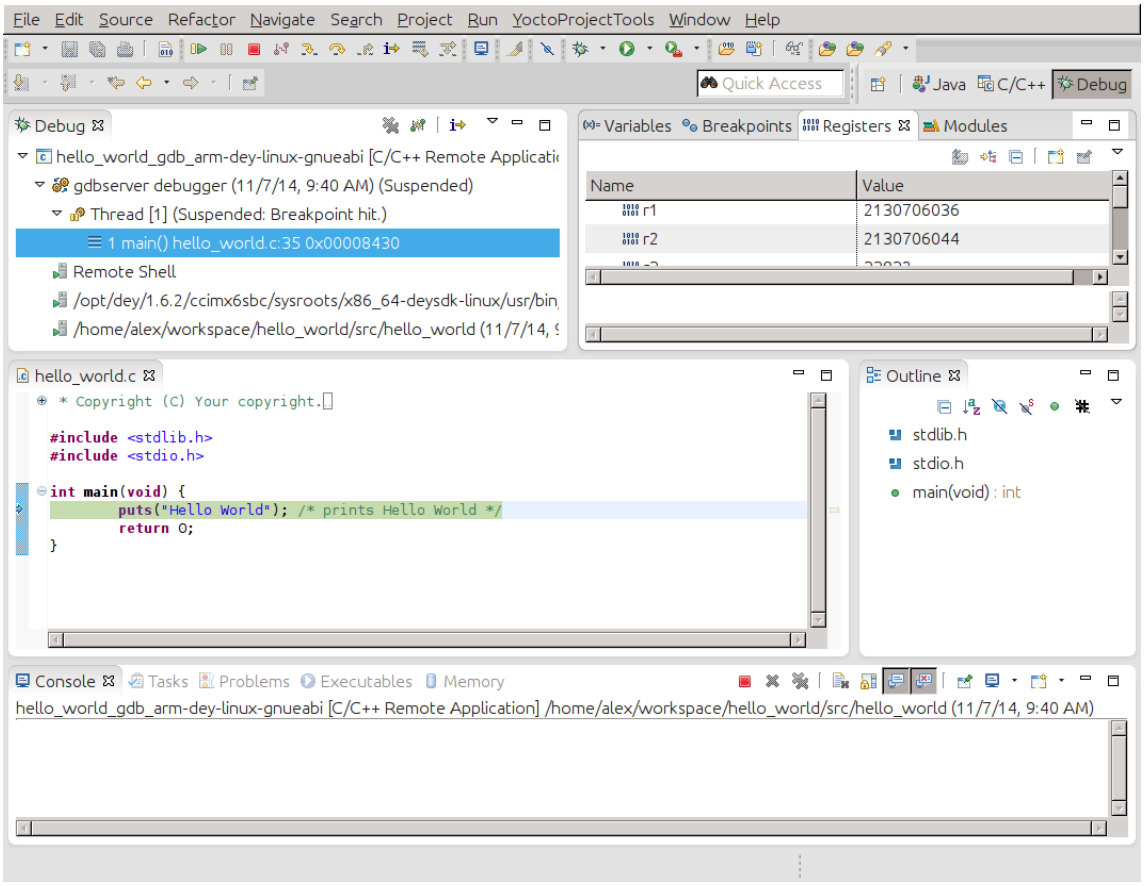
Debugging an example application

Configure the remote session as follows:

1. In **Debug Configuration,** check **C/C++ Remote application configuration**.
2. On the debugger tab, GDB debugger should point to the configured Yocto toolchain.



3. Click the Debug button. The debug perspective appears with the application executed on the external hardware.



You can find further details in the Yocto Project Development Manual.