

PulpEniX Programming and Debug Guide

This document describes Pullpenix basic programming and debug setup and capabilities, including:

- Basic C/C++ Bare-Metal Programming
- Standard terminal print/scan and file interface
- Dynamic memory allocation (limited to 4Kn in total)
- Terminal based debug
- Eclipse debug environment setup and usage.
- Time stamping and Simulation control.

1. Getting Started

If you are new to Pullpenix first perform and exercise the steps as described in text document:

pulpenix_getting_started.txt

Accomplish a successful “HELLO WORLD”

2. Basic bare-metal programming

Essentially any C/C++ vanilla programming is supported, since the current environment is bare metal, below listed functions are available provide you include the following “.h” files in your sources.

```
#include <stdio.h>
#include <stdlib.h>
#include <gpio.h>
#include <iosim.h>
#include <bm_printf.h>
```

2.1. File Access:

The returned value of below file-open functions is a unique integer which is the file index, to be used in further file access functions.

```
unsigned int bm_fopen_r(char * file_name);    // Open file for Read
unsigned int bm_fopen_w(char * file_name);    // Open file for Write
```

2.2. Formatted Print:

Following act similar to printf, fprintf formatted printing
Use the file index as returned from bm_fopen_* functions.

```
int bm_printf(const char *fmt, ...);           // Print to terminal
int bm_fprintf(int file_idx , const char *fmt, ...); // Print to file
```

2.3. Primitive standard I/O functions

Following functions provide primitive access to Standard-I/O,
In most cases you may alternatively use the bm_printf, bm_fprintf functions

Indicate the active standard-IO file index, use 0 for non-file terminal access.
void bm_access_file(unsigned int file_idx); // will effect below primitive functions

```
int bm_getc ();           // Get a Character from the active file
void bm_putc (char c);    // Put a character into the active file.
void bm_puts(char * s);   // Put a string into the active file
void bm_puth(unsigned int v); // Put an hex value as a string into the active file.
```

Explicit file destination I/O functions,
notice that the active I/O will also be modified accordingly.

```
void bm_fputh(unsigned int file_idx ,unsigned int v); // Put hex value to explicit file
void bm_fputc (unsigned int file_idx , char c);       // Put Character to explicit file
void bm_fputd(unsigned int file_idx ,int v);          // Put decimal value to explicit file
void bm_fputs(unsigned int file_idx , char * s);      // Put string to explicit file
```

```
unsigned int bm_fgetc (unsigned int file_idx);        // Get Char from explicit file
char bm_fgets (unsigned int file_idx , char * str_buf); // Get String from explicit file
char bm_fget_line (unsigned int file_idx , char * str_buf); // Get line from explicit file.
```

// Number string to Value conversion

```
unsigned int decimal_str_to_uint (char * s); // Decimal String to value
unsigned int hex_str_to_uint (char * s);     // Hex String to value.
```

You may refer to apps/bubblesort.c as some example for file interface.

3. Dynamic Memory Allocation (up to 4K)

```
void *bm_malloc(int size); // Similar to malloc
```

Simulation Control

```
unsigned int bm_time_stamp(char * time_stam_str);  
void sim_finish ();
```

4. Terminal based debug

Terminal based debug is a non-GUI debug interface (Oppose to Eclipse GUI) it is basic but, in many cases quite useful for simple debugging with quick turn-around, without the need to setup and invoke an Eclipse session.

4.1. Setup:

You need to open two active terminals on same machine, one will run the Simulation and the other will run the debugger, the two sessions tasks communicate on with the other over TCP/IP in a manner which is mostly seamless to the user. Though the terminals may communicate across the network we currently require running both sessions on the same machine to simplify network setup.

Note: To enter a specific machine (in this example to machine 01) write:

```
ssh enicsw01
```

Then enter your server password.

4.2. Simulation terminal Session

```
cd pulpenics/apps/bubblesort      # move to *.c file directory  
pulp_comp_app_noopt bubblesort    # compile bubblesort  
cd $MY_PULP_IRUN                  # move to the simulation directory  
pulp_get_app bubblesort           # configure simulation to run the hello-world application  
pulp_irun_gdb                     # start simulation in debug mode
```

4.3. Debug Terminal Session

open a second terminal on same server

```
source ... /misc/ pulpenix_setup.sh # (Not needed in your ~/.cshrc (recommended)  
cd $MY_PULP_IRUN # debug session can actually run from anywhere but we recommend  
here
```

pulp_terminal_gdb \$MY_PULP_APPS/bubblesort/bubblesort.elf # bubblesort for example

(gdb) use gdb commands here

c # continue to "main" predefined breakpoint

n # step over

s # step into

See ... *misc/gdb-refcard.pdf* for all GDB commands

4.4. Validation

To check that the program run as it should, in the end:

cd \$MY_PULP_IRUN

cat sort_out.txt

Check that the names are sorted

5. Eclipse debug environment setup and usage

5.1. Introduction

Eclipse is a GUI based open-source most popular IDE (Integrated Development Environment) it can be used as the entire environment for developing, maintaining, building, debugging and testing a project. However, we currently use Eclipse only for the purpose of **remote debug** of applications developed in the PulpEniX environment, where all the development and build flow takes place out of the Eclipse environment, as described above.

It should be noticed that Eclipse essentially is a GUI wrapper for the GDB utility (GNU debug) which is the same as used in the terminal debug mode described in earlier section, therefor successfully testing the terminal debug mode is a prerequisite to setting up the Eclipse Environment.

Side note: Eclipse is a very powerful utility but though supporting a wide range of programming language including C/C++ the Eclipse itself is a JAVA based utility running on a JVM (Java Virtual Machine) this fact has many advantages and is generally seamless to you but on the other hand introduces some challenges in making the environment easily personalized with regards to the Pullpenix setup mainly due to the fact that the JVM does not recognize the hosting environment variables. We made a significant attempt to automate the setup with not much success so far, therefor you are requested just once per project to carefully follow all following steps in order to setup an eclipse remote debug environment.

5.2. Use Desired Eclipse Version

There are plenty of Eclipse versions around, use only the following, should be downloaded once and shared per site and adjust path at misc/pulpenix_setup.sh accordingly.

(For Enics Bar-Ilan users, this is already taken care, no need to download)

Related Eclipse versions placed here:

<https://github.com/gnu-mcu-eclipse/org.eclipse.epp.packages/releases>





Just download and extract, ignore all other instructions at that web page

To extract use:

```
tar -zxvf <file name.tar.gz>
```

20180930-0922-gnumcueclipse-4.4.2-2018-09-linux.gtk.x86_64.tar.gz

▼ Assets 12

	20180930-0922-gnumcueclipse-4.4.2-2018-09-linux.gtk.x86.tar.gz
	20180930-0922-gnumcueclipse-4.4.2-2018-09-linux.gtk.x86.tar.gz.sha
	20180930-0922-gnumcueclipse-4.4.2-2018-09-linux.gtk.x86_64.tar.gz
	20180930-0922-gnumcueclipse-4.4.2-2018-09-linux.gtk.x86_64.tar.gz.sha

5.3. Eclipse remote debug environment setup

Along following guidelines, the bubblesort application is used as an example, you should obviously adjust to your desired application. Setting the environment can take place from any directory but we recommend doing it from the specific application directory.

```
cd $MY_PULP_ENV/apps/bubblesort # Go to your application directory
```

You MUST invoke Eclipse with only this command running a specific Eclipse version, don't use other versions available around.

```
pulp_eclipse
```

If asked so answer "go to workbench, never ask again"

Now follow precisely ALL next steps.

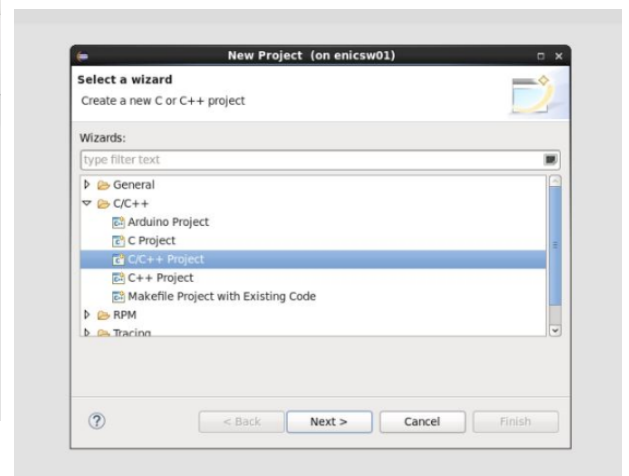
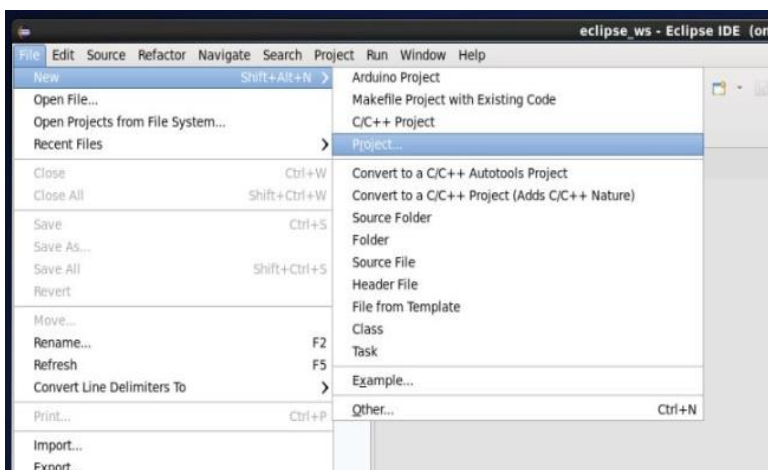
Important: at end of each step don't forget to hit

APPLY/NEXT/OK/FINISH

Notice FINISH does not necessarily mean we finished the entire process but only the specific step, you must proceed through ALL steps below.

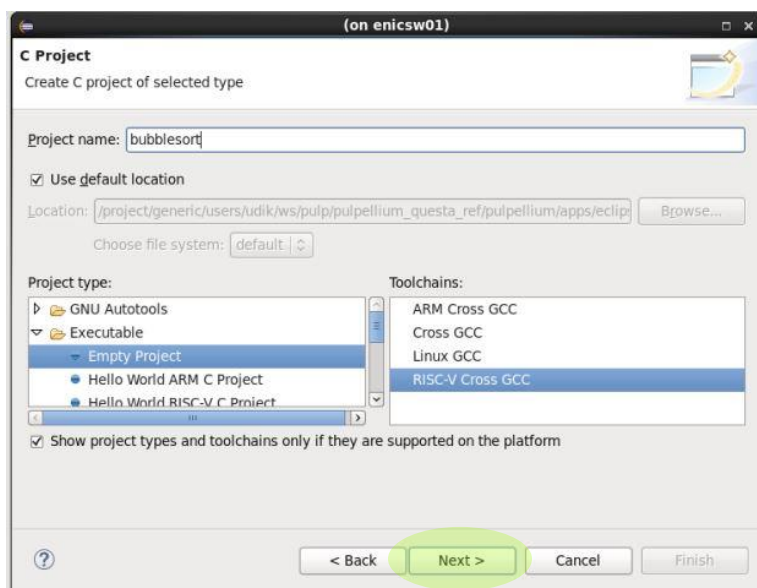
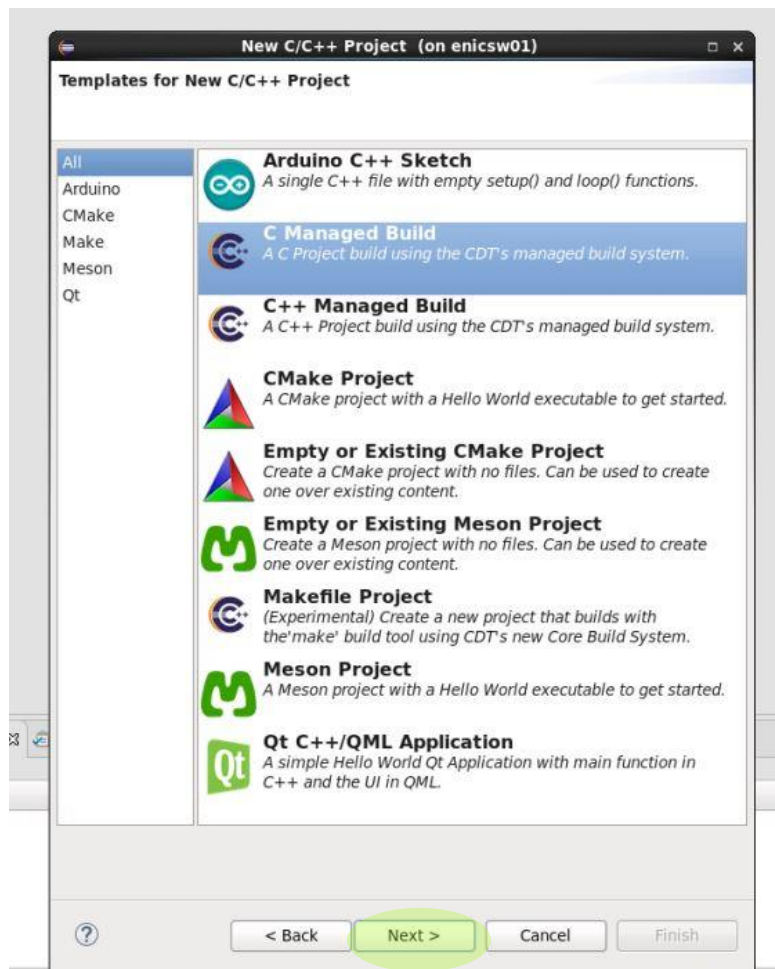
5.4. Open a new c/c++ project.

```
new -> project -> c/c++ -> c/c++ project NEXT
```

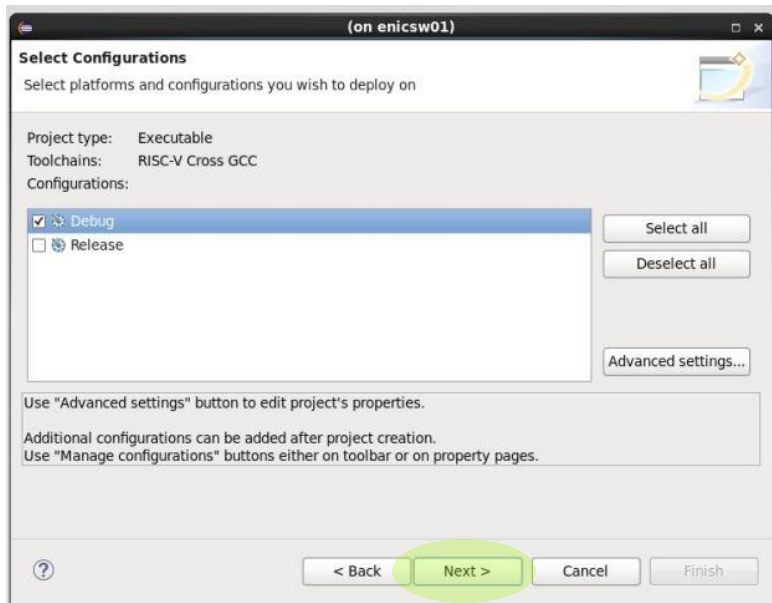


5.5. Make it a cross-debug project

C Managed Build -> Empty Project RISC-V Cross GCC , project-name bubblesort NEXT



Check debug, Uncheck release, NEXT

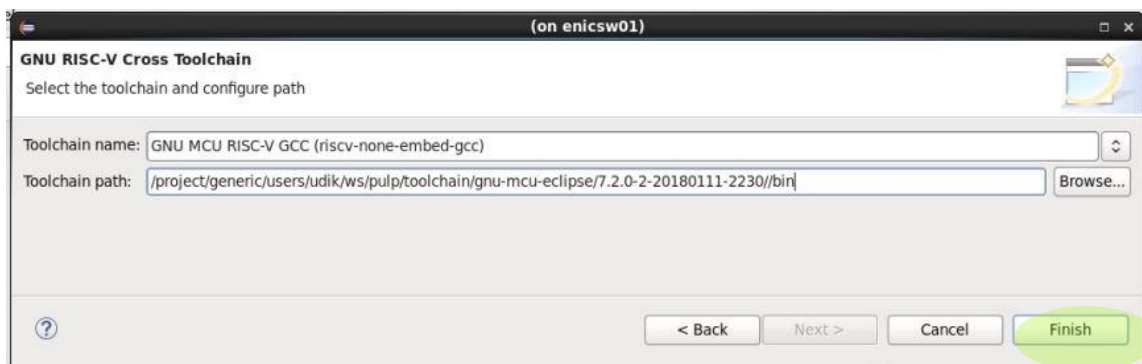


5.6. Setting the toolchain path

From any terminal that ran ... misc/pullpenix_setup.sh

echo \$RISCV_GCC_BIN

capture the returned path and copy-paste into the Eclipse requesting window

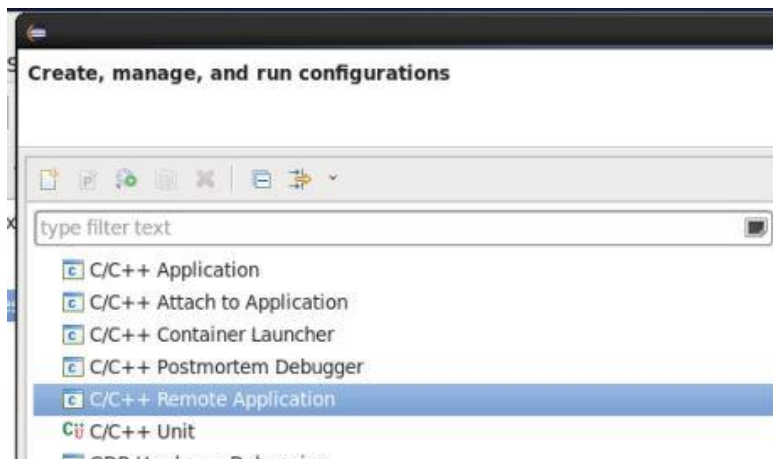
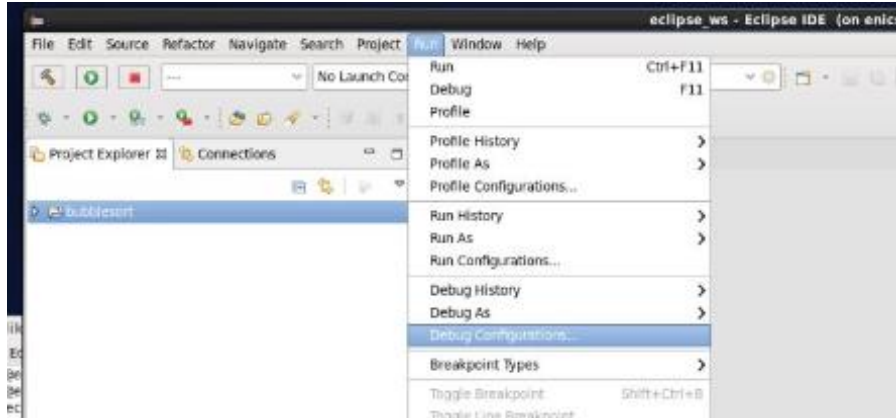


Hit **FINISH**

Sorry! Unfortunately, this is needed as Eclipse does not recognize environment variables.

5.7. Setting up the debug Configuration

run -> debug configuration -> remote application(double click or right button) -> new Configuration



Main tab:

At **name** provide a configuration name such as bubblesort_remote_debug

In **project** enter the project name in this case bubblesort

From any Terminal

```
echo $MY_PULP_APPS/bubblesort/bubblesort.elf
```

capture the result path (including the .elf file name)

```
[udik@enicsw01 bubblesort]$ echo $MY_PULP_APPS/bubblesort/bubblesort.elf  
/project/generic/users/udik/ws/pulp/pulpenix/apps/bubblesort/bubblesort.elf  
[udik@enicsw01 bubblesort]$
```

Copy-paste into **C/C++ application** and to **Remote absolute path fields**

Name: New_configuration

Project: bubblesort

C/C++ Application: /project/generic/users/udik/ws/pulp/pulpenix/apps/bubblesort/bubblesort.elf

Build (if required) before launching

Build Configuration: Use Active

☐ Enable auto build ☐ Disable auto build

☒ Use workspace settings [Configure Workspace Settings...](#)

Connection: Local

Remote Absolute File Path for C/C++ Application: /project/generic/users/udik/ws/pulp/pulpenix/apps/bubblesort/bubblesort.elf

Commands to execute before application

☐ Skip download to target path.

Using GDB (DSF) Automatic Remote Debugging Launcher - [Select other...](#)

Revert Apply

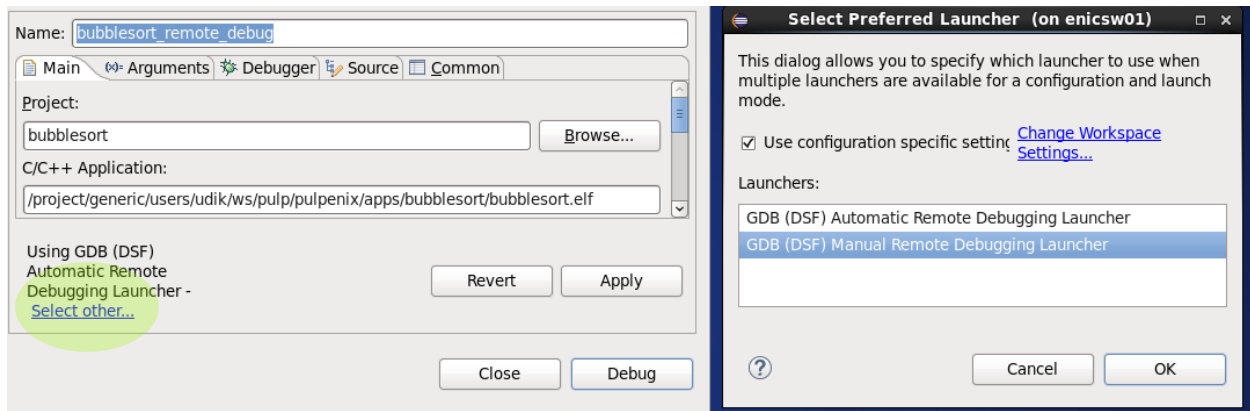
Close Debug

At the bottom hit Using GDB ... **select other**

Check “use configuration specific”

Select GDB (DSF) Manual Remote Debugging Launcher

Hit **OK**



Debugger tab:

At any terminal:

echo \$RISCV_GCC_BIN/\$RISCV_XX-gdb

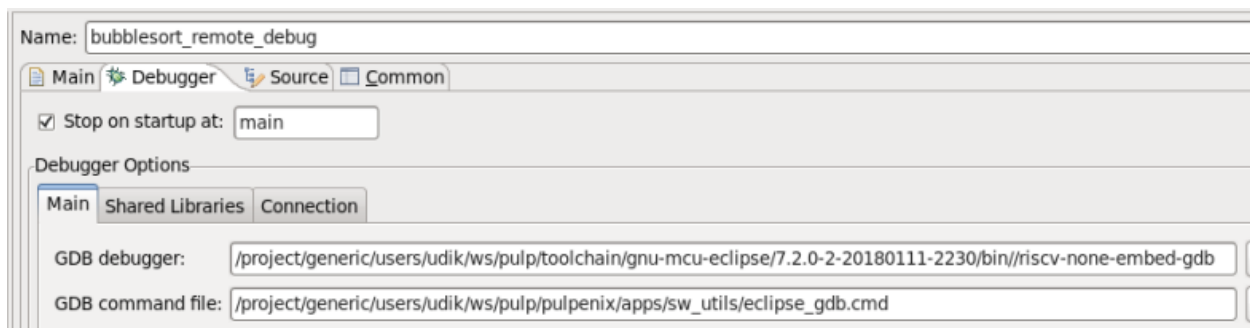
```
/project/generic/users/udik/ws/pulp/pulpenix/sim/irun
[udik@enicsw01 irun]$ echo $RISCV_GCC_BIN/$RISCV_XX-gdb
/project/generic/users/udik/ws/pulp/toolchain/gnu-mcu-eclipse/7.2.0-2-20180111-2230/bin//riscv-none-embed-gdb
```

Copy-paste path to **GDB Debugger**

echo \$MY_PULP_APPS/sw_utils/eclipse_gdb.cmd

Copy-paste to **GDB command file**

```
[udik@enicsw01 irun]$ echo $MY_PULP_APPS/sw_utils/eclipse_gdb.cmd
/project/generic/users/udik/ws/pulp/pulpenix/apps/sw_utils/eclipse_gdb.cmd
```

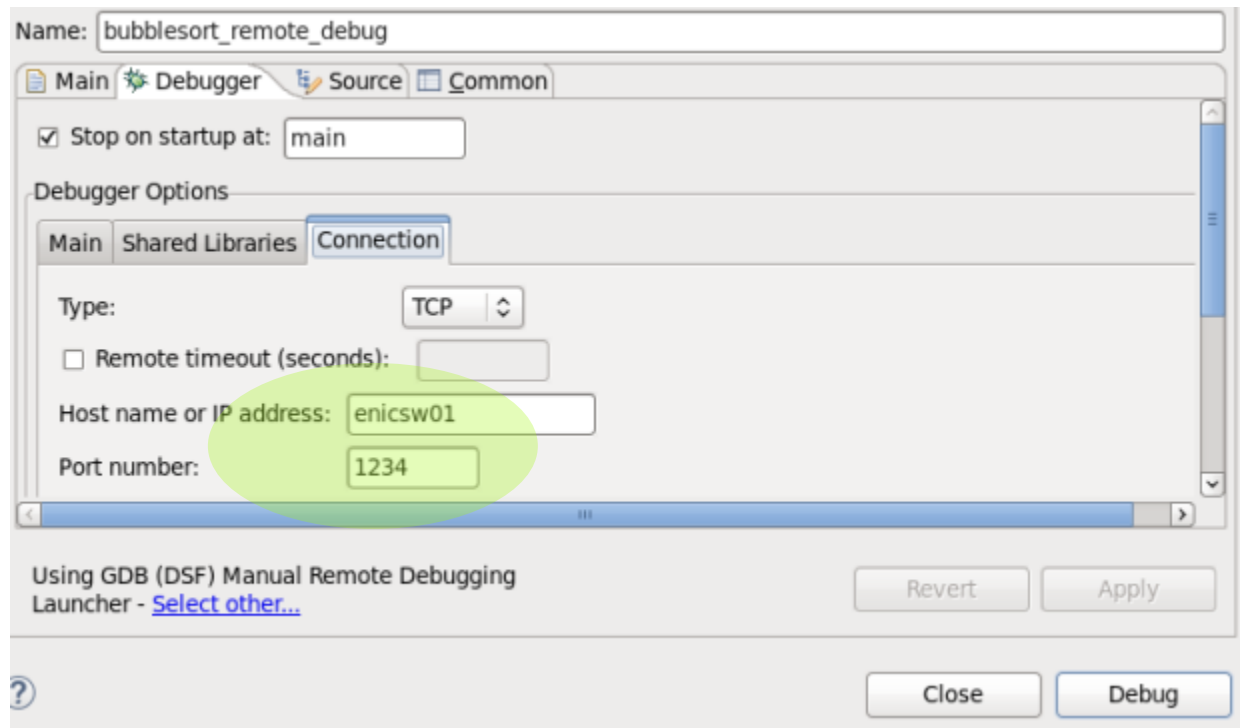


Hit **APPLY**

Now go to **Connection** sub-tab

Enter the host name your simulation session will be running on

It may theoretically be any remote host TCP available on your network but we recommend that both eclipse debug and the simulation session run on the same host.



For the port number you must use **1234** only, as this is the port number the simulation-debug bridge is configured to.

Hit **APPLY**

Now we should be done with the configuration.

if the **DEBUG** Key in the bottom right corner is available and not grayed-out than you are in good shape, otherwise something went wrong, and you should re-check all your steps above.

Obviously, your configuration is stored per project and you don't have to repeat the process for each debug session.

Now, let's proceed to an actual debug session as described in the next section.

6. Eclipse Debug co-simulation Session

Similar to the terminal debug setup, you need to open two different active terminals on the same machine, one will run the Simulation and the other will launch the eclipse debugger, the two sessions tasks communicate with the other over TCP/IP in a manner which is mostly seamless to the user. Though the terminals may communicate across the network we currently require running both sessions on the same machine to simplify network setup.

6.1. Simulation terminal Session

```
cd pulpenics/apps/bubblesort      # move to *.c file directory
pulp_comp_app_noopt bubblesort    # compile bubblesort
cd $MY_PULP_IRUN                  # move to the simulation directory
pulp_get_app bubblesort           # configure simulation to run the hello-world
application
pulp_irun_gdb                     # start simulation in debug mode
```

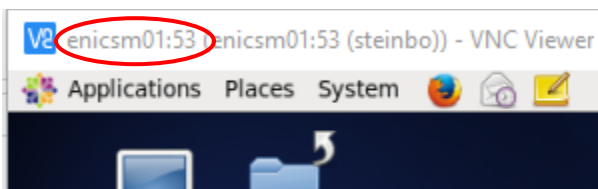
Wait till the simulation session comes up and only then start the eclipse session (to avoid timeout communication issues)

6.2. Eclipse Debug Terminal Session

First to allow Eclipse display the GUI we need to define environment variable:

```
Setenv DISPLAY <your VNC machine>:<your VNC number>
```

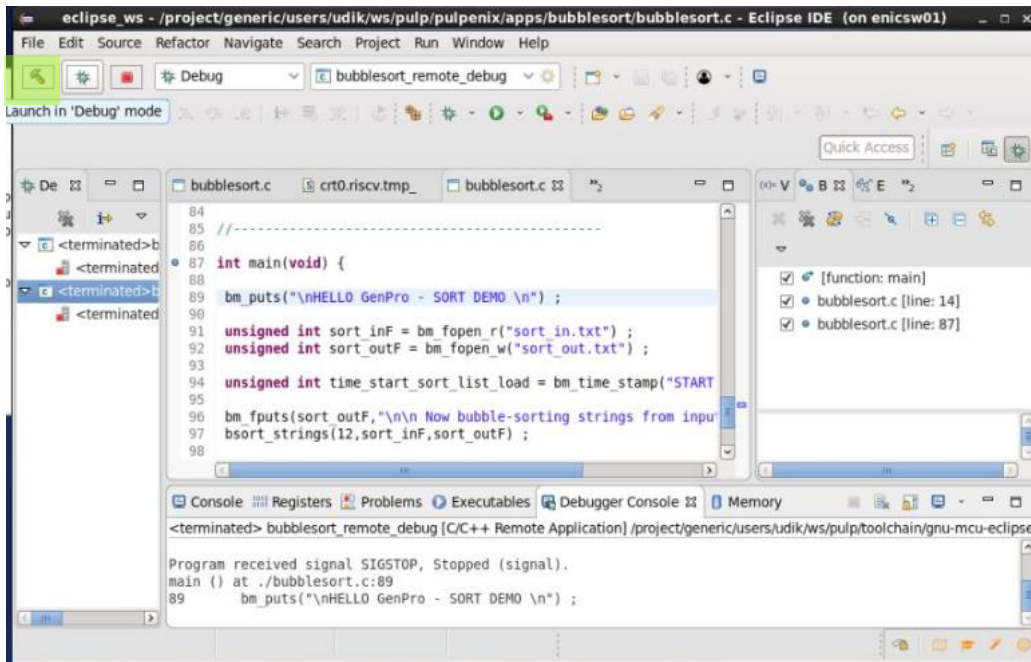
For example in the next case we will write “setenv DISPLAY enicsm01:53 ”



```
pulp_eclipse                      # wait a few seconds Eclipse IDE should come-up
```

To launch the debug you can do one of the two:

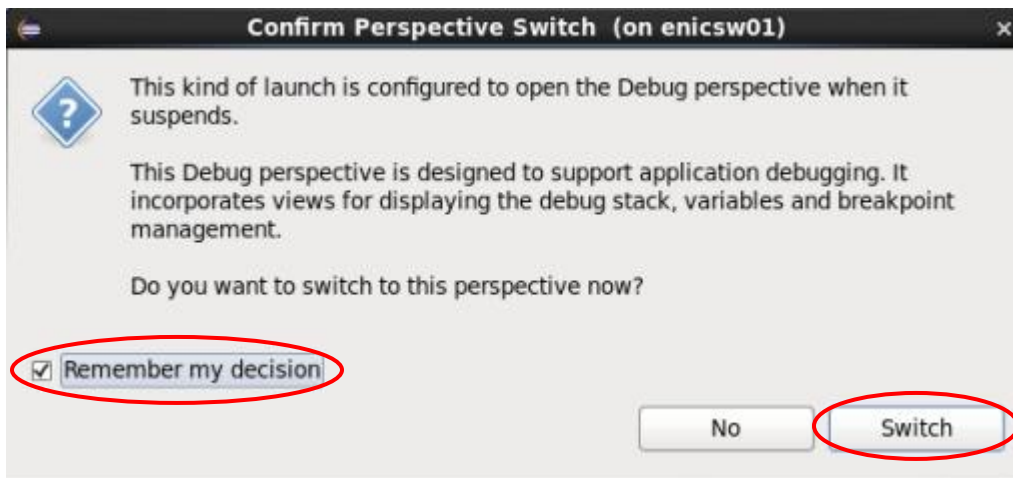
- Launch it from the bug icon (top left)->debug Configuration



- Launch it from pull down menu: Run -> debug-Configuration

After that debug window will appear ->DEBUG (in bottom right corner)

In this stage it possible that Eclipse will drop you the next message



Pick “Remember my decision” and **Switch**.

Now you should be in an active Eclipse software debug session hooked up with the simulation environment where all Eclipse standard debug goodies are available, setting breakpoints etc.

General Eclipse debug features tutorials are all over and it is out of the scope of this document, be aware some tutorials maybe confusing with specific setups not applicable for us.

However basically like any debugger the following is available:

F5 – “Step Into” (‘s’)

F6 – “Step Over” (‘n’)

F7 – “Step Return” (‘’)

F8 – “Resume” (‘c’)

Also If you gently fly-over with the mouse on some variable in the source code window, its value should be displayed etc.

6.3. Validation

To check that the program run as it should, in the end:

```
cd $MY_PULP_IRUN
cat sort_out.txt
```

Check that the names are sorted

6.4. Troubleshooting

If you exit the simulation process abnormally (Violent CTRL-C etc) some hidden background processes might yet run or hang which may conflict with a new invocation, it is recommended in such case to run the following from the terminal

kill % # Should kill all sub-processes.

That’s it for now, enjoy and share any findings or matters that need to be further clarified.