MeshNetics

ZigBeeNet[™] Software 1.0 Application Note

Creating, building & debugging ZigBeeNet applications in Eclipse

Document Summary

This document gives a brief introduction to the process of creating, building and debugging ZigBeeNet-based projects using the open-source IDE, Eclipse, WinAVR compiler and JTAGICE mkII hardware on a PC running Microsoft® Windows[™] 2000/XP.

In this tutorial we will create a small project that controls one of the LEDs on MeshNetics' MeshBean development board and try to debug it using Eclipse and JTAGICE mkII hardware.

Document Conventions

Buttons	Dialog button names are denoted in Courier: OK, Cancel			
Menu commands	Menu items are denoted in Courier and shown in order they must be selected: File -> Open			
Keyboard shortcuts	Several keys should be pressed simultaneously in the order they are listed: F7, Ctrl-Shift-F5			
Source code	Code snippets are shown in colored text: /************************************			
	User's entry.			

	<pre>void fw_userEntry(FW_ResetReason_t resetReason)</pre>			
File and directory names	andFile and directory names are taken in single quotes:ctory'bin', 'avr-gdb.exe'nes			

Intended Audience

This document is intended for developers, wanting to get familiar with writing ZigBee/802.15.4 applications using MeshNetics ZigBeeNet ZigBee stack.

Related Documents:

- [1] ZigBit[™] Development Kit. User's Guide. MeshNetics Doc. S-ZDK-451
- [2] JTAGICE mkll Quick Start Guide http://www.atmel.com/dyn/resources/prod_documents/doc2562.pdf
- [3] WinAVR User Manual / Ed. by Eric B. Weddington
- [4] AVR Studio User Guide. Available in HTML Help with the product. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

Pre-requisites

Make sure you have the latest versions of JAVA Run-time Environment (JRE) (download from <u>http://www.javasoft.com</u>), Eclipse for C++ development (download from <u>http://www.eclipse.org</u>) and WinAVR (<u>http://winavr.sourceforge.net</u>) installed on your PC.

You'll also need 1 MeshBean board [1] and 1 Atmel JTAGICE mkll device [2].

You will also have to install the JTAGICE mkII USB driver that comes with WinAVR [3] (see next chapter).

Installing the JTAGICE mkll driver

WARNING:

If you install this driver, you won't be able to use JTAGICE with AVR Studio **Error! Reference source not found.** If you want to use AVR Studio, you'll have to reinstall the driver (located in the 'usb' folder of AVR Studio installation, usually '\Program files\Atmel\AVR tools\usb').

The following instructions are for Windows XP.

1. Connect your JTAGICE mkll to your PC, turn it on and open Windows Device manager. If you did not install the driver that comes with AVR Studio, you will see a USB Device with question mark under the Other devices branch (see Figure 1).

Meshietics CREATING, BUILDING & DEBUGGING ZIGBEENET APPLICATIONS IN ECLIPSE

🖳 Device Manager	
Eile Action View Help	
Em S Computer	
E Display adapters	
E S DVD/CD-ROM drives	
E G Floppy disk controllers	
English drives	
DE ATA/ATAPI controllers	
H	
Em O Mice and other pointing devices	
E Ports (COM & LPT)	
E Sound, video and game controllers	
Fin System devices	
🗄 🖶 Universal Serial Bus controllers	
v	•

Figure 1. Device Manager screen

2. If you did install the driver from Atmel, there will be a JTAGICE mkII item under the Jungo branch. Right-click either of these items and select Update driver from the menu. In the next window (see Figure 2), select Install from a list or specific location (Advanced):



Figure 2. Hardware Update Wizard, starting screen

3. Click Next > to continue. In the next window (see Figure 3), select Don't

search. I will choose the driver to install. and click Next > to continue.

Hardware Update Wizard				
Please choose your search and installation options.				
Search for the best driver in these locations.				
Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.				
Search removable media (floppy, CD-ROM)				
Include this location in the search:				
C:\WinAVR\utils\libusb\bin				
Don't search. I will choose the driver to install.				
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.				
< <u>B</u> ack <u>N</u> ext > Cancel				

Figure 3. Hardware Update Wizard, search options screen

4. Click Next > again, then click the Have Disk... button in the window that appears (see Figure 4):

Hardware Update Wizard			
Select the device driver you want to install for this hardware.			
Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.			
Manufacturer Model Standard CD-ROM drives) CD-ROM Drive (force CDDA accurate) (Standard IDE ATA/ATAPI cor (Standard keyboards) CD-ROM Drive (force CDDA inaccurate) (Standard system devices) CD-ROM Drive (force IMAPI disable) CD-ROM Drive (IMAPI settings 0.1) CD-ROM Drive (IMAPI settings 0.2) This driver is digitally signed. Have Disk Tell me why driver signing is important Have Disk			
< <u>B</u> ack <u>N</u> ext > Cancel			

Figure 4. Hardware Update Wizard, device driver screen

- Click Browse..., navigate to the WinAVR installation directory and then to the 'Utils\Libusb\Bin' folder, select 'jtagice2.inf' and click the Open button, then click OK.
- 6. Click Next > in the window that appears and wait while the driver is installed. Click Finish to close the wizard.

Creating a New project in Eclipse

1. Start Eclipse IDE by running 'eclipse.exe'. If this is the first time you're running Eclipse, a similar screen will appear (see Figure 5).

Neshletics CREATING, BUILDING & DEBUGGING ZIGBEENET APPLICATIONS IN ECLIPSE APPLICATION NOTE



Figure 5. Eclipse starting screen

- 2. Click the rightmost button (Go to Workbench) or click the x button next to Welcome on top of the window. Like many modern IDEs, Eclipse has a workspace concept a common space for several projects, say, you can have a separate workspace for embedded development and another one for writing Windows applications. A project can belong to different workspaces. By default, Eclipse creates a new workspace in the Windows user directory (C:\Documents and Settings\ or whatever it is called on your computer). This is not the best place to keep projects in, because WinAVR tools do not like spaces in directory names. We recommend creating a separate directory for ZigBeeNet projects, for example, C:\ZigBeeNet.We will use this directory from now on.
- Select File -> Switch workspace -> Other from Eclipse application menu and enter "C:\ZigBeeNet" in the Workspace: textbox (the directory will be created if it does not exist). Eclipse will restart and open new workspace.

4. Select File -> New -> Project... from menu, expand the C section is the list that appears, select C Project and click Next > button to continue. Enter the project name ("DebugTest") in the Project name: text box and click Finish. Sometimes the following window will pop up:

Copen Associated Perspective?
This kind of project is associated with the C/C++ perspective. Do you want to open this perspective now?
<u>Remember my decision</u>
<u>Y</u> es <u>N</u> o

Figure 6. Open Associated Perspective dialog

5. Just click Yes to continue.

Setting up external tools

To use the programming/debugging features of JTAGICE mkll in Eclipse, we'll have to set up 2 "external tools" that will be run every time you need to program the device with new image and before starting the debugging session. Since WinAVR is a command-line set of tools, programming the module with image we build is done by a command-line tool, 'avarice.exe'. We just set up a "wrapper" for this tool, so that we do not have to type in anything every time we need to run it.

- Select Run -> External Tools -> Open External Tools Dialog... from the menu (see Figure 5).
- 2. Click the New launch configuration button in toolbar in the upper left corner of the External Tools dialog (see Figure 7).
- 3. Enter Burn image in the Name: text box. Click the Browse File System... button under the Location: text box, navigate to the WinAVR installation directory and select the 'avarice.exe' file (usually located in the 'bin' subdirectory of WinAVR).
- 4. Click the Browse Workspace... button under the Working Directory: text box and select your current project (DebugTest), then press OK. Note that text box contents change to "\${workspace_loc:/DebugTest}", which is not a standard path name, but an environment variable (we will discuss these later in this document).
- 5. Type "-2 -repf DebugTest.elf -j usb" in the Arguments: text box.
- 6. Go to the Common subtab (see Figure 8) and check the External tools checkbox in the Display in favourites menulist.
- 7. Click the Apply button to save new settings.

Meshiletics IN ECLIP	G, BUILDING & DEBUGGING ZIGBEENET APPLICATIONS SE APPLICATION NOTE
External Tools Create, manage, and run cor Run a program	ifigurations
Image: Second state Image: Second sta	Name: Burn image Main Refresh Environment Common Location: C:\WinAVR\bin\avarice.exe C:\WinAVR\bin\avarice.exe Browse Workspace Working Directory: #(workspace_loc:/DebugTest) Browse Workspace Browse File System Variables Variables Arguments: -2 -repf DebugTest.elf -j usb Variables Variables
Filter matched 3 of 3 items	Appí <u>v</u> Re <u>v</u> ert
0	<u>R</u> un Close

Figure 7. External Tools dialog, Main subtab

Meshhetics CREATING, BUILDING & DEBUGGING ZIGBEENET APPLICATIONS APPLICATION NOTE

🖶 External Tools	X
Create, manage, and run co Run a program	nfigurations
Image: The second se	Name: Burn image Main Refresh Environment Save as Local file Shared file: Display in favorites menu Console Encoding Default (Cp1251) Othgr Standard Input and Output Standard Input and Output File: Workspace File System Variables Variables
Filter matched 3 of 3 items	Apply
0	<u>R</u> un Close

Figure 8. External Tools dialog, Common subtab

The 'avarice.exe' program also needs to be run every time you want to start a debugging session, so we set up another "external tool" entry for this, called Debug mode (see Figure 7). Click the New launch configuration button again and enter Debug mode in the Name: text box. Once again, enter full path to 'avarice.exe' in the Location: text box and current project path in the Working Directory: text box. In the Arguments: text box, enter "-2 -j usb :2525" (note the space between 'usb' and ':'), then click Apply. Click Close to close the dialog.

Creating a makefile

WinAVR toolchain requires the use of a so-called "makefile" – a plain text file that contains all the settings (compiler switches/paths/libraries/etc.) required to build the target image. We supply with a sample makefile that you can use a reference in future projects (see below).

To add a makefile to your project, select File -> New -> File (see Figure 5) or right-click the Project Explorer window and select New -> File from the pop-up menu.

🖶 New File	
File	
Create a new file resource.	
Enter or select the parent folder:	
DebugTest	
DebugTest	
File na <u>m</u> e: makefile	
<u>A</u> dvanced >>	
0	<u>F</u> inish Cancel

Figure 9. New File dialog

Enter "makefile" in the File name: text box and click Finish to close the dialog (see Figure 9). An empty makefile will be created and opened in the source code editor. Paste the following sample makefile contents into the editor.

Meshletics CREATING, BUILDING & DEBUGGING ZIGBEENET APPLICATIONS IN ECLIPSE APPLICATION NOTE

```
*****
# Makefile for the project DebugTest
****
CROSS COMPILE = avr
CPU = atmega1281
PROJNAME = DebugTest
PROJECT = $(PROJNAME).elf
SHELL = /bin/bash
#### COMPILER FLAGS #######
CFLAGS = -mmcu=$(CPU)
CFLAGS += -Os
CFLAGS += -q
CFLAGS += -Wall -W
CFLAGS += -ffunction-sections
CFLAGS += -Wl, --gc-sections
#Initial DebugTest interval, ms
CFLAGS += -DDebugTest PERIOD=1000
#### DEFINES FLAGS #######
# Can be AT86RF230, AT86RF230B, AT86RF231, AT86RF212
MAC=AT86RF230
# Can be ATMEGA1281, AT91SAM7X256
HAL=ATMEGA1281
ifeq ($(MAC), AT86RF230)
MAC LIB=MACrf230
else
ifeq ($(MAC), AT86RF230B)
MAC LIB=MACrf230b
else
ifeq ($(MAC), AT86RF231)
MAC LIB=MACrf231
else
ifeq ($(MAC), AT86RF212)
MAC LIB=MACrf212
else
MAC LIB=MACrf230
endif
endif
endif
endif
```

Meshipetics CREATING, BUILDING & DEBUGGING ZIGBEENET APPLICATIONS IN ECLIPSE

```
ifeq ($(HAL), ATMEGA1281)
HAL PATH=HAL/atmega1281
HAL LIB=HALatmega1281
else
HAL PATH=HAL/at91sam7x256
HAL LIB=HALat91sam7x256
endif
STACK DIR = $(ZBN DIR)/Components
##### PATHS FLAGS OF INCLUDES #########
INCLUDEDIRS = \
              -I./include \
              -I$(STACK DIR)/SystemEnvironment/include \
              -I$(STACK DIR)/APS/include \
              -I$(STACK DIR)/NWK/include \
              -I$(STACK DIR)/ZDO/include \
              -I$(STACK DIR)/MAC PHY/include \
              -I$(STACK DIR)/MAC PHY/MAC HWD PHY/include \
              -I$(STACK DIR)/MAC PHY/MAC HWI/include \
              -I$(STACK DIR)/$(HAL PATH)/HAL HWI/include \
              -I$(STACK DIR)/$(HAL PATH)/HAL HWD/include \
              -I$(STACK DIR)/BSP/include \
              -I$(STACK DIR)/ConfigServer/include \
              -I$(STACK DIR)/PersistDataServer/include \
             -I$(STACK DIR)/Security/BuildingBlocks/include
###### LIB #########
LIBDIRS = \setminus
          -L$(STACK DIR)/APS/lib \
          -L$(STACK DIR)/ZDO/lib \
          -L$(STACK DIR)/NWK/lib \
          -L$(STACK DIR)/MAC PHY/lib \
          -L$(STACK DIR)/$(HAL PATH)/lib \
          -L$(STACK DIR)/SystemEnvironment/lib \
          -L$(STACK DIR)/BSP/lib \
          -L$(STACK_DIR)/Security/BuildingBlocks/lib \
          -L$(STACK DIR)/PersistDataServer/lib
## Libraries
LIBS = -1Main -1APS -1ZDO -1$(HAL LIB) -1BSP -1NWK -
lSystemEnvironment -1$(MAC LIB) -IAPS -1ZDO -1$(HAL LIB) -
lBSP -lNWK -lSystemEnvironment -lSSPsw -lPersistDataServer
AS
                = $(CROSS COMPILE)-as
LD
                = $(CROSS COMPILE)-ld
CC
                = $(CROSS COMPILE)-gcc
```

```
Meshietics
CREATING, BUILDING & DE
```

```
CREATING, BUILDING & DEBUGGING ZIGBEENET APPLICATIONS
IN ECLIPSE APPLICATION NOTE
```

```
CPP
                = $(CROSS COMPILE)-g++
AR
                = $(CROSS COMPILE)-ar
NM
                = $(CROSS COMPILE)-nm
                = $(CROSS COMPILE)-strip
STRIP
OBJCOPY
                = $ (CROSS COMPILE) - objcopy
OBJDUMP
               = $(CROSS COMPILE)-objdump
               = $(CROSS COMPILE)-size
SIZE
BUILDDIR = $(PRJ HOME)
objects = \setminus
          $(BUILDDIR)/objs/DebugTest.o \
          $(STACK DIR)/ConfigServer/objs/ConfigServer.o
## Build
all: $(objects) $(PROJECT) $(PROJNAME).srec $(PROJNAME).hex
$(objects):
     $(CC) $(CFLAGS) $(INCLUDEDIRS) -c $^ -o $@
$(BUILDDIR)/objs/DebugTest.o: $(BUILDDIR)/DebugTest.c
$(STACK DIR)/ConfigServer/objs/ConfigServer.o:
$(STACK DIR)/ConfigServer/src/configServer.c
$(PROJECT): $(objects)
     $(CC) $(objects)
$(STACK DIR)/$(HAL PATH)/lib/WdtInit.o $(CFLAGS)
$(INCLUDEDIRS) $(LIBDIRS) $(LIBS) -lm -o $(PROJECT)
     $(SIZE) -td $(PROJECT)
$(PROJNAME).srec:
     $(OBJCOPY) -O srec --srec-len 128 $(PROJECT)
$(PROJNAME).srec
$(PROJNAME).hex:
     $(OBJCOPY) -O ihex $(PROJECT) $(PROJNAME).hex
#burn:
    avarice -2epf ${TARGET} -j /dev/ttyS0
#
## Clean target
clean:
    -rm -rf $(TARGET) $(objects) $(PROJECT)
$(PROJNAME).hex $(PROJNAME).eep $(PROJNAME).srec
## End of makefile
```

Adding source files to the project

Now that we can add a source (.c) file that contains the code we're going to build to the project.

Select File -> New -> Source file from the application menu (see Figure 5) or right-click the Project Explorer window and select New -> Source file from the pop-up menu. Enter "DebugTest.c" in the Source File: text box and click the Finish button to continue. Note that you have to specify the '.c' extension of the file. Here's the sample code we're going to use in this example below:

```
LED Blinking Implementation Project: C source
#include <apsTimer.h>
#include <leds.h>
#include <taskManager.h>
#include <zdo.h>
#include <configServer.h>
#include <aps.h>
// variables/defines
#define BLINK DELAY 500 // Period of blinking
static HAL AppTimer t blinkTimer;
// functions
void StartBlinkTimer();
void TimerFired();
void ZDO StartNetworkConf(ZDO StartNetworkConf t*
confirmInfo);
void ZDO MgmtNwkUpdateNotf(ZDO MgmtNwkUpdateNotf t
*nwkParams);
void ZDO WakeUpInd();
void ZDO SleepInd();
Application task.
void APL TaskHandler()
{
 StartBlinkTimer();
}
void StartBlinkTimer()
 blinkTimer.interval = BLINK DELAY;
 blinkTimer.mode = TIMER REPEAT MODE;
 blinkTimer.callback = TimerFired;
 HAL StartAppTimer(&blinkTimer);
```

```
void TimerFired()
{
  BSP ToggleLed(LED RED);
}
// The following functions MUST be present to build the
executable image
void ZDO StartNetworkConf(ZDO StartNetworkConf t*
confirmInfo)
{
}
void ZDO MgmtNwkUpdateNotf(ZDO MgmtNwkUpdateNotf t*
nwkParams)
{
}
void ZDO WakeUpInd()
{
}
void ZDO SleepInd()
{
}
   eof DebugTest.c
```

Using environment variables in makefile

Notice the following lines in the makefile text:

```
## Path to Stack
STACK_DIR = ZBN_DIR/Components
## Modules directories paths.
BUILDDIR = $(PRJ HOME)
```

ZBN_DIR and PRJ_HOME are environment variables that point to the API folder of ZigBeeNet installation and directory containing your project files, respectively. By using variables you can avoid specifying paths directly in the makefile and thus the need to modify the makefile for every project you create. Instead, you can use the variables in Eclipse to specify the paths.

1. Select Project -> Properties from the application menu or right-click the

project name in Project explorer and select Properties from the pop-up menu.

Meshhetics IN ECLIPSE	BUILDING & DEBUGGING ZIGBEEI	NET APPLICATIONS	TION NOTE
Properties for DebugTest type filter text	Environment Configuration: Default Environment variables to set Variable Val PWD C (1) CWD C (1) CW	lue veZeeNet\DebugTest veZeeNet\DebugTest	Manage configurations Manage configurations New Select Edit Remove Undefine
3			OK Cancel

Figure 10. Properties dialog

Click the New... button in Properties dialog (see Figure 10). Enter 2. "ZBN_DIR" in the Name: text box and full path to the ZigBeeNet directory of your ZDK installation in the Value: text box, like seen in Figure 11 below.

ŧ		×	
Name:	ZBN_DIR		
Value:	C:/ZDK/ØIGBEENET	Variables	
Add to all configurations			
OK Cancel			

Figure 11. Variable definition dialog

IMPORTANT:
You have to use forward slash ('/') rather than back slash ('\') in path names.
0 Oligh on the place the window A group with he could alte the list

- Repeat the same steps to create the second variable, PRJ_HOME with value set to the directory of your current project ('c:/ZigBeeNet/debugtest' or similar).
- 5. Click Apply, then OK to close the dialog.

Setting up the debugger

Select Run -> Open Debug Dialog... from application menu. Right-click the C/C++ Local application and select New from the pop-up menu.

In the dialog that appears take the following steps:

- 1. Enter "Debug" in the Name: text box.
- 2. Click Browse next to Project: text box, select your project (DebugTest) from the list that appears and click OK.
- 3. Enter "DebugTest.elf" in the C/C++ Application: text box.
- 4. Click Apply button.
- 5. Click Debugger tab.
- 6. Select gdbserver Debugger from the Debugger: list.
- 7. Uncheck the Stop on startup at: checkbox.
- Click Browse next to GDB Debugger: text box and navigate to 'bin' subfolder of WinAVR installation directory. Select the 'avr-gdb.exe' file and click Open (or just enter the full path to 'avr-gdb.exe', usually something like 'C:\WinAVR\bin\avr-gdb.exe').
- 9. Clear the GDB command file: text box.
- 10. Click the Connection subtab.
- 11. Change connection type to TCP and port number to 2525.
- 12. Click the Common subtab.
- 13. Check the Debug item in the Display in favourites menu list.
- 14. Click Apply, then Close.

Building the image

Once we've set everything up, it is time to build the image we're going to debug. Select Project -> Build project from the application menu. WinAVR tools output is shown in the console window in the bottom of the screen (see Figure 12).

Programming the device with newly built image

Once the image is successfully built, we need to program the MeshBean with it via JTAG.

1. Connect your JTAG device to the MeshBean by using supplied connector and to your PC using USB cable and power up both devices.

 Now select Run -> External tools -> Burn image (we've created this entry earlier). If all connections and settings are fine, then node programming will start with progress indicator in the console window.

EC/C++ - DebugTest/DebugTest.c - Eclipse Platform				
<u>File E</u> dit Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch <u>R</u> un <u>P</u> r	oject <u>A</u> VR Target <u>Wi</u> ndow <u>H</u> elp			
📬 • 🗄 🖆 🗟 💥 🙋 • 😂 •	C • G • G •	·] 🥙 🔗] 🖪 🖨] 🐓 👻	$\bullet \Leftrightarrow \diamondsuit \bullet \bullet \bullet$	🖹 🔤 c/c++
Project Explorer 🛛 🗖 🗖	Makefile C DebugTest.c 🛛			
E SebugTest → dep B SebugTest.c → Makefile	<pre>void ZDD_MgmtNvkUpdateNotf(ZDO_E void ZDD_WakeUpInd(); void ZDD_SleepInd(); /************************************</pre>	<pre>gmtNwkUpdateNotf_t *nwkP DELAY; REPEAT_MODE; ired; r); be present to build the artNetworkConf_t* confir igmtNwkUpdateNotf_t* nwkP</pre>	arams); executable image mInfo) arams)	A gasTimer.h leds.h leds.h gastimer.h leds.h gastasManager.h zoo.h configServer.h gas.h totastellarKTimer(): void totastellarKTimer(): void totastellarKTimer(): void APL_TaskHandler(): void SasttBinKTimer(): void Coo_StartNetworKconf(ZC Zoo_WakeUpInd(): void Zoo_StartNetworKconf(ZC Zoo_WakeUpInd()
	void ZDO_WakeUpInd() () Tacks 22 Console Properties			
0 kens				
	V ! Description	Resource Path	Location	
L] ∎◆			Writable Smart Insert 80 : 15	

Figure 12. Debug screen

Once the programming is finished, an error message "USB bulk read error: usb_reap_async: error: A device attached to the system is not functioning." might be displayed. Just ignore it.

Don't forget you have to reprogram the MeshBean every time you make changes to your code and rebuild the image.

Debugging the application

Once the MeshBean is programmed with the executable image, we can start the debugging session.

1. First, let's set up a breakpoint in one of the functions, void TimerFired()

(see Figure 12). Position the cursor on the second line of this function and

select Run -> Toggle breakpoint from the menu or press Ctrl-Shift-B. The line will be marked by a round symbol to the left of it.

2. To start the debugging, first select Run -> External tools -> Debug mode (this is the second entry we've created earlier). This will put your JTAGICE into debug mode and allow further control of the target device. <u>You need to</u> <u>run this command every time you want to start a debugging session.</u>

The output in the console window should look like this:

```
AVaRICE version 2.6, May 15 2007 17:07:24
Defaulting JTAG bitrate to 1 MHz. Make sure that the
target frequency is at least 4 MHz or you will likely
encounter failures controlling the target.
```

 Next, select Run -> Debug from the application menu or press F11. The following window may appear (see Figure 13):

🖶 Confirm Perspective Switch			
2	This kind of launch is configured to open the Debug perspective when it suspends.		
	This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.		
	Do you want to open this perspective now?		
Remember my decision			
	<u>Y</u> es <u>N</u> o		

Figure 13. Perspective confirmation

 Click Yes to switch the view (or, in Eclipse terminology, perspective) to debugging mode. The screen layout will change into something like this, seen in Figure 14:



	_ 5 ×
Elle Edit Refactor Navigate Search Run Project Window Help	
$ \stackrel{\text{\tiny (1)}}{=} \cdots \stackrel{\text{\tiny (2)}}{=} \stackrel{\text{\tiny (2)}}{=} \cdot \stackrel{\text{\tiny (2)}}{=} \stackrel{\text{\tiny (2)}}{=} \cdot \stackrel{\text{\tiny (2)}}{=} \cdot \stackrel{\text{\tiny (2)}}{=} \stackrel{\text{(2)}}{=} \stackrel{\text{(2)}}{=} \stackrel{\text{(2)}}{=} \stackrel{\text{(2)}}{=} \text{(2$	😭 🏇 Debug 🔤 C/C++
🏇 Debug 🕄 🔰 🙀 🎲 🗊 🗉 😹 Modules 😥 🤕 😥 🧠 👘 🐨 🏱 🗖 🕅 🕬 Variables 🖄 💊 Breakpoints) 🚻 Registers) 🔜 Modules 🔅	
🖻 💁 Debug mode (Program) Name Value	
C:\WinAVR\bin\avarice.exe	
E-C Debug [C/C++ Local Application]	
G dg gabserver Debugger (23.10.07 17:56) (Suspended)	
🖃 🔐 Thread [0] (Suspended: Breakpoint hit.)	
S Timer Tuerd () citezeenet(debug/cst()/boug/est.ci+5 uxuuuuuuzdo)	
- 5 mind_mine() 0x001406	
= 3 TOSH_run_next_task() 0x00011228	
-== 2 TOSH_run_task() 0x00011232	
□ = 1 main() 0x000112e0	
C:\WinAVR\bin\evr-gdb.exe (23.10.07 17:56)	<u>_</u>
□ pi C:\eZeNet\DebugTest.DebugTest.Def (23.10.07 17:56)	
st.	7 1
	+2 ~ ~ ~ -
Main loop.	
apptimer.h	
void mainLoop() # LED	
biov : Optimized a state of the	
// Additonal user's activities.	
7 fw_userEntry(FW_	ResetReason_t) : void
Blink timer handler.	
void timerFired()	
static bool on = 0;	
gpio_setState(LED, on);	
on = on ? 0 : 1; // Toggle.	
// eof DebugTest.c	
Jebug [C/C++ Local Application] C:\e2eNet\Debug1est\Debug1est\Debug1est.elr (23.10.0/ 17:56)	
	-
4	

Figure 14. Project perspective screen

You can switch between debug and project perspectives at any time using the Debug and C/C++ buttons in the top right corner of the Eclipse window.

Execution should now break on the line we've set a breakpoint at. You can use the vast Eclipse debugging facilities to see stack trace, memory, variables, registers and many other things. The Window -> Show view menu gives you quick access to any of the debugging windows.

The Run menu contains most of the debugging features you need, like "step into", "step over", "run to line" and others.

- 5. Press F8 or select Run -> Resume from the menu to resume execution.
- 6. Select Run -> Terminate to stop debugging.