ZigBeeNet™ Software 1.0

Application Note

Creating, building & debugging ZigBeeNet applications in AVR Studio

Document Summary

This document gives a brief introduction to the process of creating, building and debugging ZigBeeNet-based projects using Atmel's IDE, AVR Studio [1], WinAVR compiler [2], [3] and JTAGICE mkII hardware [4] 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 AVR studio 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.
***********************************************/
void fw_userEntry(FW_ResetReason_t resetReason)
```

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] AVR Studio User Guide. Available in HTML Help with the product.
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

[2] WinAVR User Manual / Ed. by Eric B. Weddington

[3] Using the GNU Compiler Collection/ By Richard M. Stallman and the GCC Developer Community

[4] JTAGICE mkII Quick Start Guide
http://www.atmel.com/dyn/resources/prod_documents/doc2562.pdf

[5] ZigBit™ Development Kit User's Guide. MeshNetics Doc. S-ZDK-451

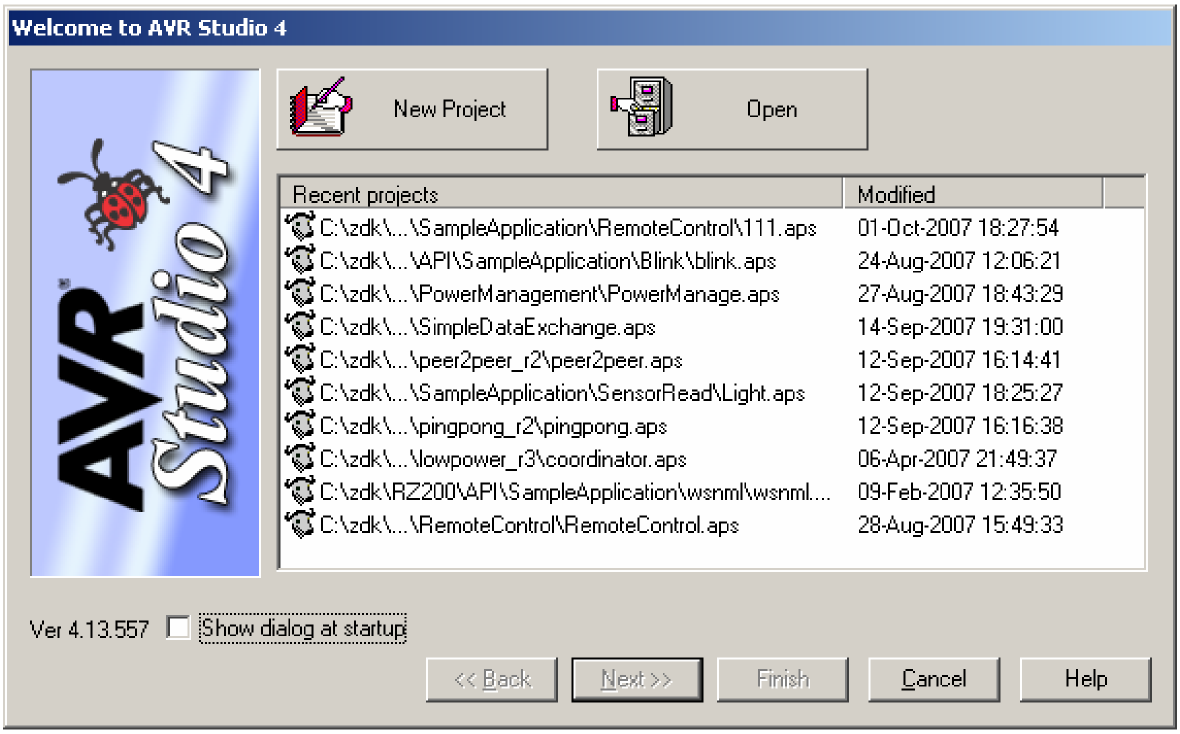
Pre-requisites

Make sure you have the latest versions of AVR Studio/Service Pack (http://atmel.com/dyn/products/tools_card.asp?family_id=607&family_name=AVR+8%2DBit+RISC+&tool_id=2725) and WinAVR (http://winavr.sourceforge.net) installed on your PC.

You'll also need one MeshBean board [5] and Atmel JTAGICE mkII device.

Creating a New project in AVR Studio

1. Start AVR Studio. A project wizard window appears (see Figure 1).



Welcome to AVR Studio 4

Recent projects	Modified
C:\zdk\...\SampleApplication\RemoteControl\111.aps	01-Oct-2007 18:27:54
C:\zdk\...\API\SampleApplication\Blink\blink.aps	24-Aug-2007 12:06:21
C:\zdk\...\PowerManagement\PowerManage.aps	27-Aug-2007 18:43:29
C:\zdk\...\SimpleDataExchange.aps	14-Sep-2007 19:31:00
C:\zdk\...\peer2peer_r2\peer2peer.aps	12-Sep-2007 16:14:41
C:\zdk\...\SampleApplication\SensorRead\Light.aps	12-Sep-2007 18:25:27
C:\zdk\...\pingpong_r2\pingpong.aps	12-Sep-2007 16:16:38
C:\zdk\...\lowpower_r3\coordinator.aps	06-Apr-2007 21:49:37
C:\zdk\RZ200\API\SampleApplication\wsnml\wsnml....	09-Feb-2007 12:35:50
C:\zdk\...\RemoteControl\RemoteControl.aps	28-Aug-2007 15:49:33

Ver 4.13.557 □ Show dialog at startup

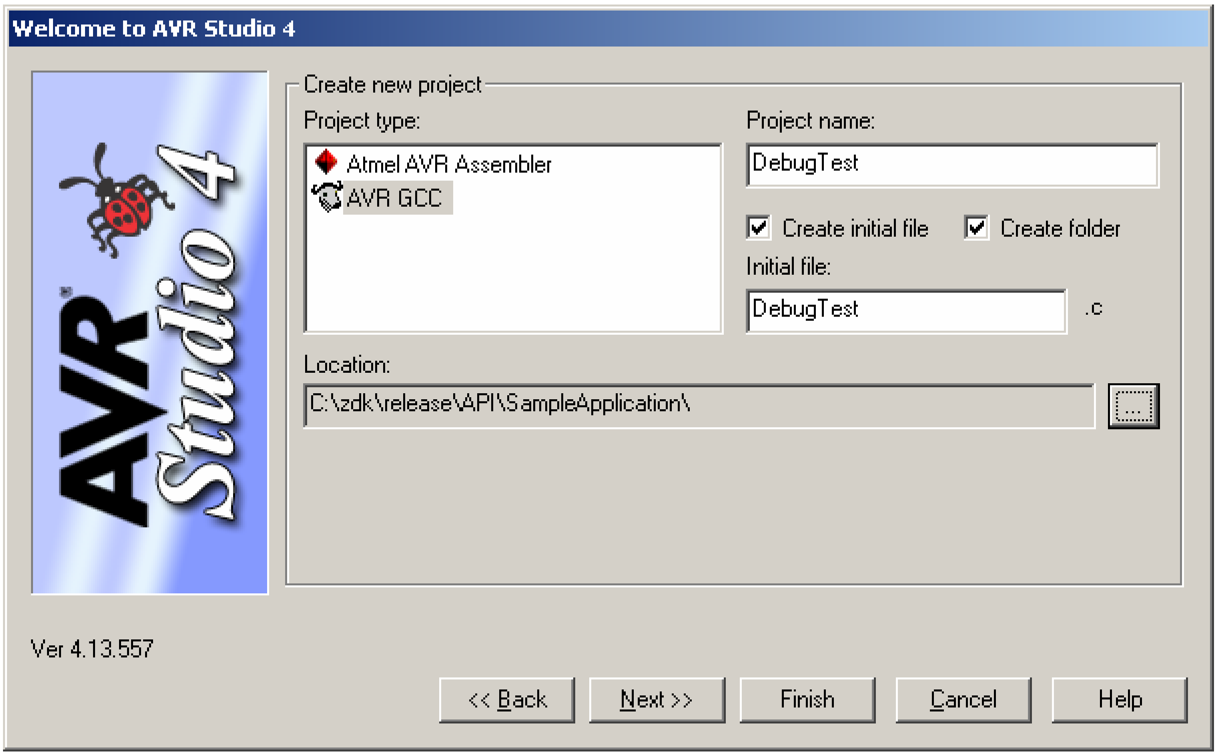
Figure 1. Project wizard window

2. Press the `New Project` button. If the wizard window does not appear, select `Project` -> `New project` from AVR Studio main menu.

3. In the window that appears next (see Figure 2), select `AVR GCC` in the `Project type:` list, enter project name in the `Project name:` text box. If you want AVR studio to create initial source file for you, check the `Create initial file` checkbox and enter file name (without extension) in the `Initial file:` text box.

   If you want AVR studio to create a folder with the name of the project, check the `Create folder` checkbox. Finally, press the ... button next to the `Location:` text box and select a directory for your project.

   Press `Next` >> to go to the next screen.


Welcome to AVR Studio 4

Create new project

Project type:
- Atmel AVR Assembler
- AVR GCC

Project name: DebugTest

☑ Create initial file ☑ Create folder

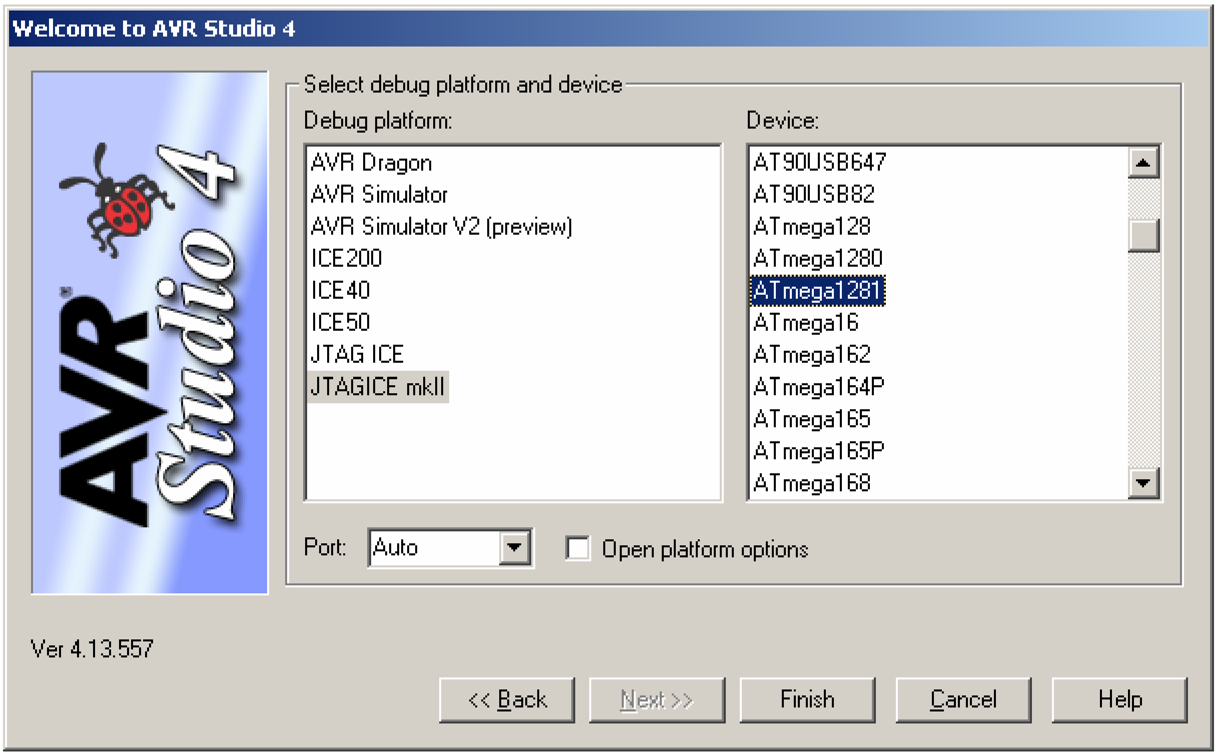
Initial file: DebugTest .c

Location: C:\zdk\release\API\SampleApplication\

Ver 4.13.557

Figure 2. Creating new project

4. In the next screen that appears (see Figure 3), select `JTAGICE mkII` from the `Debug platform:` list and `ATmega1281` from the `Device:` list. Press `Finish` to close the wizard.



Debug platform: JTAGICE mkII

Device: ATmega1281

Port: Auto

Figure 3. Selecting debug platform and device

Setting the Project Options

The WinAVR toolchain requires using a so-called "makefile" – a plain text file that contains all the project options, source/target/library file names and everything else needed to build the target image. This is somewhat more difficult, than using GUI, but gives more control over the build process.

To use makefile in AVR studio, go to `Project` -> `Configuration Options` menu and check the `Use external makefile` checkbox. Press the … button next to the text box and select your makefile (usually located in the same folder as AVR Studio project). Press `OK` to close the window.

Here's a sample makefile that you can use as a reference in future projects:

```
########################################################
# 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 += -g
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

ifeq ($(HAL), ATMEGA1281)
HAL_PATH=HAL/atmega1281
HAL_LIB=HALatmega1281
else
HAL_PATH=HAL/at91sam7x256
HAL_LIB=HALat91sam7x256
endif

STACK_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 = \
          -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 = -lMain -lAPS -lZDO -l$(HAL_LIB) -lBSP -lNWK -
lSystemEnvironment -l$(MAC_LIB) -lAPS -lZDO -l$(HAL_LIB) -
lBSP -lNWK -lSystemEnvironment -lSSPsw -lPersistDataServer

AS              = $(CROSS_COMPILE)-as
LD              = $(CROSS_COMPILE)-ld
CC              = $(CROSS_COMPILE)-gcc
CPP             = $(CROSS_COMPILE)-g++
AR              = $(CROSS_COMPILE)-ar
NM              = $(CROSS_COMPILE)-nm
STRIP           = $(CROSS_COMPILE)-strip
OBJCOPY         = $(CROSS_COMPILE)-objcopy
OBJDUMP         = $(CROSS_COMPILE)-objdump
SIZE            = $(CROSS_COMPILE)-size
```

```
BUILDDIR = .
objects = \
          $(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
```

Writing Source Code

You can write your source code either in AVR Studio or using any suitable text editor. General file structure is the same you would use in any other C-based project (headers, definitions, code). Here's the code we're going to use in this tutorial:

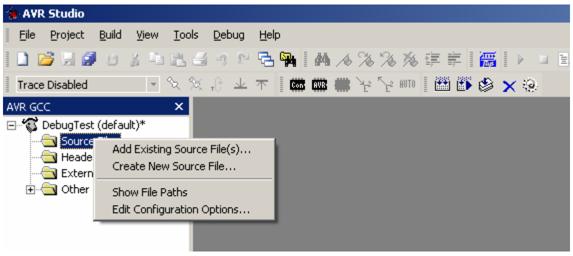
```
/*********************************************************
  LED Blinking Implementation Project: C source
*********************************************************/
#include <apsTimer.h>
#include <leds.h>
#include <taskManager.h>
#include <zdo.h>
```

```
#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
```

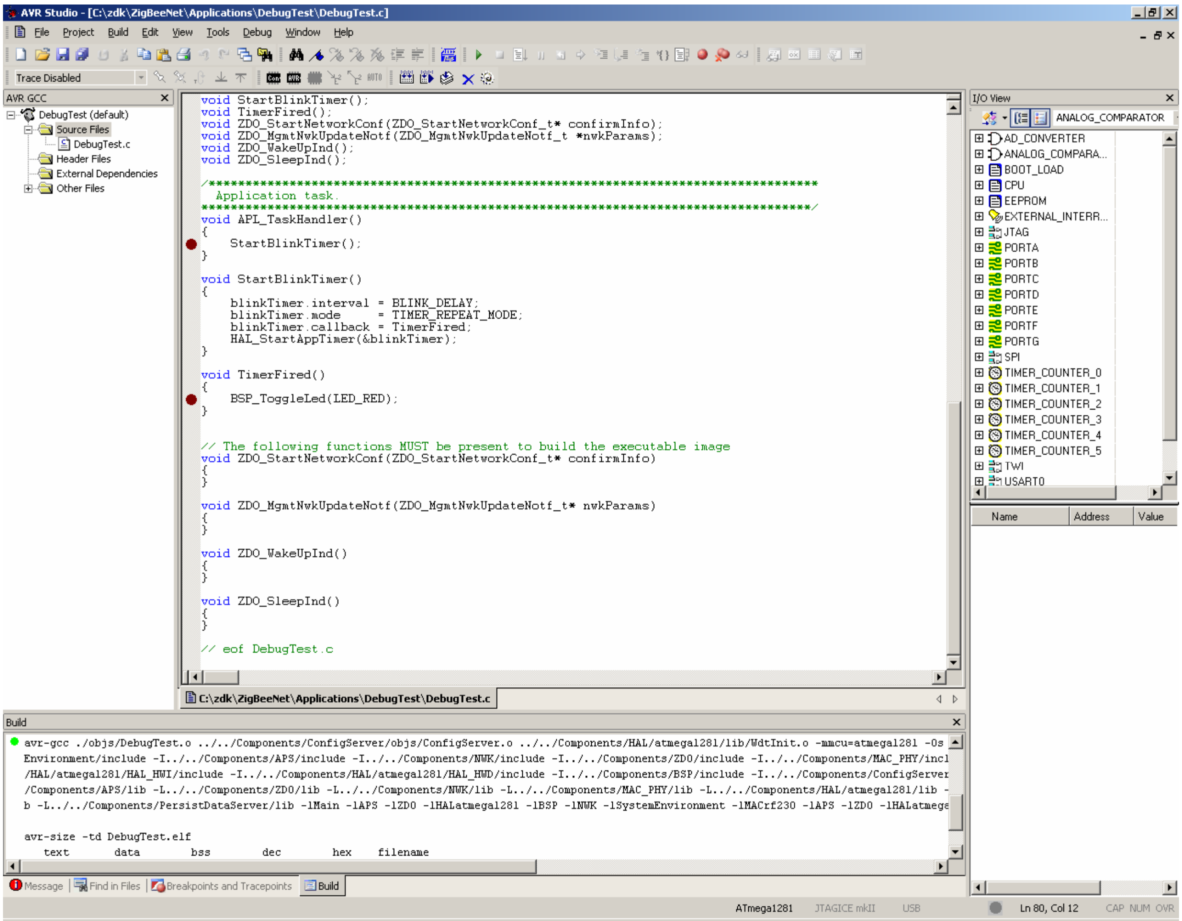
Adding existing source file to the project

To add an existing source code file to the AVR Studio project, click the “`Source files`” item in the AVR GCC panel (normally located on the left of the AVR Studio window), and select “`Add Existing Source File(s)`” from the pop-up menu. Select your source file using the dialog that appears and press “`OK`” to add this file to the project.


AVR Studio
File Project Build View Tools Debug Help
Trace Disabled
AVR GCC
DebugTest (default)*
Source
Heade
Extern
Other
Add Existing Source File(s)...
Create New Source File...
Show File Paths
Edit Configuration Options...

Building the Image

Once you've finished writing your code, you can build the executable image from it by selecting `Build` -> `Build` from menu or just pressing F7. You can also use the `Rebuild all` command if you have a large project and want to be sure all changes to code are compiled. Build window (bottom of the screen) will display output from the avr-gcc compiler (see Figure 4).



```c
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
```

```
avr-gcc ./objs/DebugTest.o ../../Components/ConfigServer/objs/ConfigServer.o ../../Components/HAL/atmega1281/lib/WdtInit.o -mmcu=atmega1281 -Os
Environment/include -I../../Components/APS/include -I../../Components/NWK/include -I../../Components/ZDO/include -I../../Components/MAC_PHY/incl
/HAL/atmega1281/HAL_HWI/include -I../../Components/HAL/atmega1281/HAL_HWD/include -I../../Components/BSP/include -I../../Components/ConfigServer
/Components/APS/lib -L../../Components/ZDO/lib -L../../Components/NWK/lib -L../../Components/MAC_PHY/lib -L../../Components/HAL/atmega1281/lib -
b -L../../Components/PersistDataServer/lib -lMain -lAPS -lZDO -lHALatmega1281 -lBSP -lNWK -lSystemEnvironment -lMACrf230 -lAPS -lZDO -lHALatmega

avr-size -td DebugTest.elf
   text      data       bss       dec       hex   filename
```

Figure 4. Building the image

If the code contains no errors, you will get a `DebugTest.elf` file in the same directory where project/source files are located (unless you've specified a different output directory in makefile). This is the executable image that will be used by debugger.

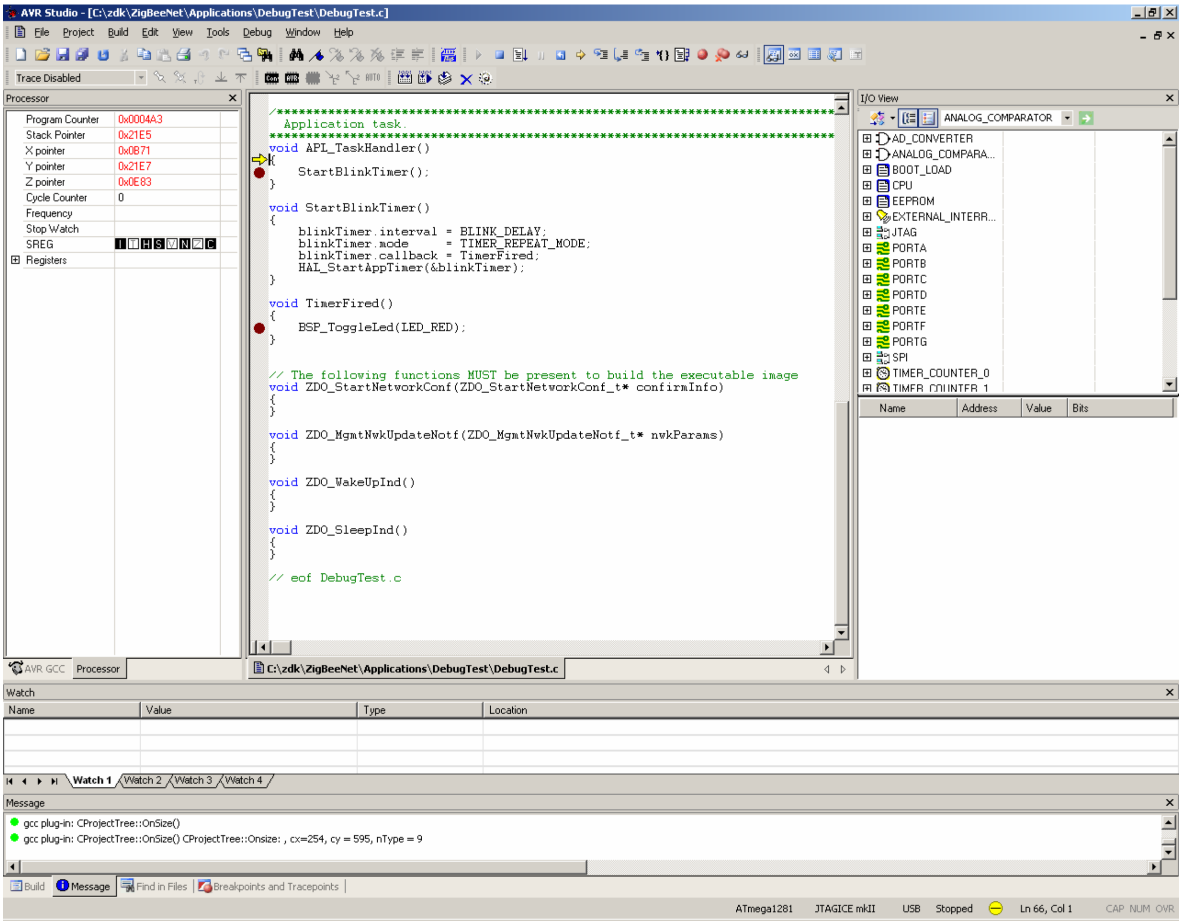
Debugging the Image

You can now start the debugging session for the image you've just built. First, connect your MeshBean to JTAGICE device and power up the devices (both can be powered from USB).

To test the debugging features of AVR Studio, set a breakpoint (press F9) on any line of code that is sure to be executed. In our case, we do it for the `TimerFired()` function, that is going to be called periodically to blink on of the LEDs on MeshBean board.

Now, select `Debug` -> `Start debugging` from AVR Studio application menu or press `Ctrl-Alt-Shift-F5`. Note that this command becomes available only after you've run the `Build` command. If you restart AVR Studio, you will have to build the target image again. AVR Studio will start programming the device with the built image, indicating the progress with a progress bar at the bottom of the window.

Once the image download is complete, AVR Studio will break execution at the first line of your program. Press F5 to continue execution. AVR Studio will now break and pause execution at the first breakpoint you've set in your code (see Figure 5):



AVR Studio - [C:\zdk\ZigBeeNet\Applications\DebugTest\DebugTest.c]

Program Counter	0x0004A3
Stack Pointer	0x21E5
X pointer	0x0B71
Y pointer	0x21E7
Z pointer	0x0E83
Cycle Counter	0
Frequency	
Stop Watch	
SREG	I T H S V N Z C
⊞ Registers	

```
/*****************************************************************************
  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
```

C:\zdk\ZigBeeNet\Applications\DebugTest\DebugTest.c

```
gcc plug-in: CProjectTree::OnSize()
gcc plug-in: CProjectTree::OnSize() CProjectTree::Onsize: , cx=254, cy = 595, nType = 9
```

ATmega1281 JTAGICE mkII USB Stopped Ln 66, Col 1 CAP NUM OVR

Figure 5. Debugging

In the `View` menu you can select additional debug windows: processor, I/O, disassembler, watch, memory, registers, etc. Pressing the right mouse button in the code editor also offers a few debugging options, such as adding/removing breakpoints and watch expressions.

You can resume execution any time by pressing F5. Press `Shift-F5` to restart the debugger session.

To stop the debugger, select `Debug -> Stop` debugging from menu or press `Ctrl-Shift-F5`. This command is available in "stop mode" only, i.e. you first have to pause execution (break) and then stop it completely.