



eZeeNet™ Software 1.7 Nota de Aplicación

Creando, construyendo y depurando aplicaciones eZeeNet en AVR Studio



Resumen del documento

Este documento ofrece una breve introducción al proceso de crear, construir y depurar proyectos basados en eZeeNet, utilizando el IDE de Atmel, AVR Studio [1], compilador WinAVR[2], [3] y hardware JTAGICE mkII [4] en un PC bajo Microsoft[®] Windows[™] 2000/XP.

En este tutorial se enseña la forma de crear un pequeño proyecto que controle uno de los LEDs en la placa desarrollo MeshBean de MeshNetic y depurarlo por medio de AVR Studio y el hardware JTAGICE mkII.

Convenciones utilizadas en el documento

Botones	Los nombres de los botones de diálogo son representados en Courier: OK, Cancel			
Comandos de menú	Los ítems son representados en Courier y mostrados en el orden en que deben ser seleccionados: File -> Open			
Atajos de teclado	Muchos atajos deben ser presionadas simultáneamente en el orden en que se muestran: F7, Ctrl-Shift-F5			
Código fuente	Los fragmentos de código se muestran en texto coloreado: /************************************			
	User's entry.			

	void fw userEntry(FW ResetReason t resetReason)			

Intención del documento

Este documento está dirigido a los desarrolladores que quieran familiarizarse con la creación de aplicaciones mediante la pila MeshNetics eZeeNet ZigBee.

Documentos relacionados

- [1] AVR Studio. User Guide. Available in HTML Help with the product.
- [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



Prerrequisitos

Asegúrese de que dispone de las últimas versiones de AVR Studio/Service Pack (http://atmel.com/dyn/products/tools_card.asp?family_id=607&family_name=AVR+8%2DBit+RISC+&tool _id=2725) y WinAVR (http://winavr.sourceforge.net) instaladas en su PC.

También necesitará una tarjeta MeshBean y un dispositivo Atmel JTAGICE.

Creando un nuevo proyecto en AVR Studio

1. Inicie AVR Studio. Aparecerá en pantalla el asistente de proyectos (ver Figura 1).

Welcome to AVR Studio	4 New Project Open	
	Recent projects	Modified
	C:\zdk\\SampleApplication\RemoteControl\111.aps	01-0ct-2007 18:27:54
, 9	C:\zdk\\API\5ampleApplication\Blink\blink.aps	24-Aug-2007 12:05:21
	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 16:15:38
	C:\zdk\\lowpower_i3\coordinator.aps	06-Apr-2007 21:49:37
	C:Vzdk\BZ200\API\SampleApplication\wsnml\wsnml	09-Feb-2007 12:35:50
	C:\zdk\\RemoteControl\FemoteControl.aps	28-Aug-2007 15:43:33
Ver 4 13 557 🗖 Show o	lialog at startup	
	<< Back Next >> Finish	<u>Cancel</u> Help



- Presione el botón New Project. Si no aparece el asistente, seleccione Project -> New project del menú principal de AVR Studio.
- 3. En la ventana que aparece a continuación (ver Figura 2), seleccione AVR GCC en Project type, e introduzca el nombre del proyecto en Project name. Para crear automáticamente un archivo fuente inicial, seleccione la casilla Create initial file e introduzca un nombre para el archivo (sin extensión) en Initial file. Para crear una carpeta con el nombre del proyecto seleccione la casilla Create folder. Finalmente, presione el botón ... junto a Location y seleccione un directorio para su proyecto. Presione Next >> para continuar a la siguiente pantalla.





Welcome to AVR Studio 4				
	Create new project Projec: type: Almel AVR Assembler AVR GCC Location: C:\zdk\release\\PI\SampleApplication\	Project name: DebugTest Create initial file Create folder Intial file: DebugTest .c		
Ver 4.13.557	<< <u>B</u> ack <u>N</u> ext>>	Finish <u>C</u> arcel Help		

Figura 2: Creando un nuevo proyecto

4. En la siguiente pantalla que aparece (ver Figura 3), seleccione JTAGICE mkII de Debug platform y ATmega1281 de Device. Pulse Finish para cerrar el asistente.

Welcome to AVR Studio 4				
Sindio 4	Select debug platform and device Debug platform: AVR Dragon AVR Simulato: AVR Simulato: V2 (preview) ICE 200 ICE 40 ICE 50 JTAG ICE JTAGICE mkII	Device: A^90US8647 A^90US882 A^mega128 A^mega1280 A_mega1281 A^mega1281 A^mega162 A^mega165 A^mega165 A^mega165 A_mega165 A_mega165 A_mega165 A_mega165 A_mega165		
	Port: Auto 🔽 🗖 Open platfor	m options		
Ver 4.13.557	ZZ Back Nevt N	Finish Cancel	Help	

Figura 3: Seleccionando plataforma de depuración y dispositivo



Ajustando las opciones del proyecto

La colección de herramientas WinAVR requiere el uso del llamado *makefile*, un archivo de texto plano que contiene todas las opciones de proyecto, nombres de librerías/fuentes/objetivos y todo lo necesario para construir la imagen objetivo. Esto resulta un tanto más difícil que el uso de GUI, pero ofrece un mayor control sobre el proceso de construcción.

Para usar *makefile* en AVR Studio, vaya a Project -> Configuration en el menú y seleccione la casilla Use external makefile. Pulse el botón ... junto al cuadro de texto y seleccione su *makefile* (normalmente localizado en la misma carpeta que el proyecto AVR Studio). Pulse OK para cerrar la ventana.

Aquí tiene un ejemplo de makefile que puede utilizar como referencia en proyectos futuros:

```
****
# Makefile for the project DebugTest
*****
## General Flags
PROJECT = DebugTest
MCU = atmega1281
TARGET = DebugTest.elf
CC = avr-gcc.exe
## Options common to compile, link and assembly rules
COMMON = -mmcu=\$(MCU)
## Compile options common for all C compilation units.
CFLAGS = $(COMMON)
CFLAGS += -Wall -gdwarf-2 -Os -std=gnu99 -fsigned-char -fpack-
struct
CFLAGS += -MD - MP - MT $(*F).o -MF dep/$(@F).d
## Assembly specific flags
ASMFLAGS = $ (COMMON)
ASMFLAGS += $(CFLAGS)
ASMFLAGS += -x assembler-with-cpp -Wa,-gdwarf2
## Linker flags
LDFLAGS = $ (COMMON)
LDFLAGS += -Wl, -Map=DebugTest.map
## Intel Hex file production flags
#HEX FLASH FLAGS = -R .eeprom
HEX EEPROM FLAGS = -j .eeprom
HEX EEPROM FLAGS += --set-section-flags=.eeprom="alloc,load"
HEX EEPROM FLAGS += --change-section-lma .eeprom=0
## Path to Stack, StackSupport, HAL, TOSLib.
SUPPORT DIR = .../..
#SUPPORT DIR=$(wildcard ../../WSN)$(wildcard
../../trunk/src/WSN)
## Modules directories paths.
APP DIR = .
STACK DIR = $(SUPPORT DIR)/Stack
STACK SUPPORT DIR = $(SUPPORT DIR)/StackSupport
```







```
TOSLIB DIR = $ (SUPPORT DIR) / TOSLib
HAL DIR = $ (SUPPORT DIR) / HAL/HAL R6
FRAMEWORK DIR = $ (SUPPORT DIR) / Framework
## Include Directories.
INCLUDES = -I"$(STACK DIR)/include" \
-I"$(TOSLIB DIR)/include" \
-I"$(HAL DIR)/base/include" \
-I"$(HAL DIR)/eZeeNet/include" \
-I"$(HAL DIR)/meshBean2/include" \
-I"$(STACK SUPPORT DIR)/include" \
-I"$(STACK SUPPORT DIR)/include/stack" \
-I"$(FRAMEWORK DIR)/include"
## Library Directories
LIBDIRS = -L"$(HAL DIR)/lib" \
-L"$(TOSLIB DIR)/lib" \
-L"$(FRAMEWORK DIR)/lib" \
-L"$(STACK SUPPORT DIR)/lib"
## Libraries
LIBS = -ltos \setminus
-lmeshBean2 \
-lfw ∖
-lZigBitInt \
-lstackSupport \
-lc
SRC = $(APP DIR)/DebugTest.c \
$(STACK SUPPORT DIR)/src/ConfigServer.c \
## Objects explicitly added by the user
LINKONLYOBJECTS = $(STACK DIR)/lib/NWKMACLibA.o \
$(STACK DIR)/lib/APLLibA.o \
$(HAL DIR)/lib/wdtinit.o
## Build
all: clean $(TARGET) DebugTest.hex DebugTest.eep size
## Compile
DebugTest.o: DebugTest.c
$(CC) $(INCLUDES) $(CFLAGS) -c $<</pre>
ConfigServer.o: ../../StackSupport/src/ConfigServer.c
$(CC) $(INCLUDES) $(CFLAGS) -c $<</pre>
## Build
all: clean $(TARGET) $(PROJECT).srec $(PROJECT).hex
$(PROJECT).eep size
##Link
$(TARGET):
$(CC) $(CFLAGS) $(INCLUDES) $(SRC) $(LINKONLYOBJECTS) -o
$(TARGET) $(LIBDIRS) $(LIBS)
%.srec: $(TARGET)
avr-objcopy -O srec $(HEX FLASH FLAGS) $< $@
%.hex: $(TARGET)
```



Creando, construyendo y depurando aplicaciones eZeeNet en AVR Studio



```
avr-objcopy -O ihex $(HEX_FLASH_FLAGS) $< $@
%.eep: $(TARGET)
avr-objcopy $(HEX_EEPROM_FLAGS) -O ihex $< $@
%.lss: $(TARGET)
avr-objdump -h -S $< > $@
size: ${TARGET}
@echo
@avr-size -C --mcu=${MCU} ${TARGET}
## Clean target
.PHONY: clean
clean:
-rm -rf $(OBJECTS) DebugTest.elf dep/* DebugTest.hex
DebugTest.eep DebugTest.srec
## Other dependencies
-include $(shell mkdir dep 2>/dev/null) $(wildcard dep/*)
```

Escribiendo el código fuente

Puede escribir su código fuente en AVR Studio o mediante cualquier editor de texto apropiado. La estructura general del archivo es la misma que la que usaría en cualquier otro proyecto basado en C (cabeceras, definiciones, código). A continuación se muestra el código que se va a usar en este tutorial:

```
LED Blinking Implementation Project: C source
#include "framework.h"
#include "gpio.h"
#include "apptimer.h"
#define LED GPIO 0 // Pin connected to LED.
// Functions' declarations.
void mainLoop(); // Main loop.
void timerFired(); // Blink timer handler.
User's entry.
void fw userEntry(FW ResetReason t resetReason)
{
// Initialize the pin connected to the LED.
gpio setConfig(LED, GPIO OUTPUT);
// Open and start blink timer.
{
int handle;
handle = appTimer open(timerFired);
appTimer_start(handle, TIMER_REPEAT_MODE, 1000);
}
// Start main loop.
fw setUserLoop(20, mainLoop);
```





```
/****
                      *****************
Main loop.
void mainLoop()
{
// Addtional user's activities.
}
             /****
Blink timer handler.
      void timerFired()
{
static bool on = 0;
gpio_setState(LED, on);
on = on ? 0 : 1; // Toggle.
}
// eof DebugTest.c
```

Una vez haya terminado de escribir su código, puede construir la imagen ejecutable seleccionando Build -> Build del menú o presionando F7. Pude hacer uso también del comando Rebuild all en proyectos largos para asegurar que se compilan todos los cambios del código. La ventana Build (en la parte inferior de la pantalla) mostrará la salida del compilador *avr-gcc* (ver Figura 4).







Figura 4: Construyendo la imagen

Si el código no contiene errores, se generará un archivo *DebugTest.elf* en la misma carpeta en la que se encuentra los archivos del proyecto (a menos que especifique un directorio de salida diferente en el *makefile*). Esta es la imagen ejecutable que utilizará el depurador.

Depurando la imagen

Puede ahora comenzar la sesión de depuración para la imagen que acaba de construir. Primero, conecte su MeshBean al dispositivo JTAGICE y alimente los dispositivos (pueden ser alimentados por USB).

Para probar las características de depuración de AVR Studio, sitúe un *breakpoint* (pulse F9) en cualquier línea de código de la que esté seguro que va a ser ejecutada. En este caso, se hará en la función *timerFired()*, que va a ser llamada periódicamente para el parpadeo de los LEDs de la placa MeshBeans.

Ahora, seleccione Debug -> Start debugging de la aplicación AVR Studio o pulse Ctrl-Alt-Shift-F5. Este comando se encuentra disponible únicamente después de ejecutar el comando Build. Si reinicia AVR Studio, deberá construir de nuevo la imagen. AVR Studio comenzará a programar el dispositivo con la imagen construida, indicando el progreso con la barra de progreso en la parte inferior de la ventana. Después de que la programación esté completa, saltará la siguiente ventana (ver Figura 5).



Creando, construyendo y depurando aplicaciones eZeeNet en AVR Studio





Figura 5: Confirmación de carga de datos a la EEPROM

eZeeNet utiliza una porción de la EEPROM para almacenar sus parámetros. Pulse en el botón OK para continuar.

En algunas ocasiones, AVR Studio interrumpirá su ejecución y mostrará la ventana de desensamblado con contenidos inválidos (ver Figura 6).

🐌 AVR Studio - [Di	sassembler]						
Ele Project	<u>Buid Edt View T</u> ook	Debug <u>W</u> indow <u>H</u> el	P		_ 5×		
🗅 😂 🗔 剑 👔	法国法国法院	□▲ ■ ●▲ ▲ ③	乳液注意	1 AR 1	토		
The second second			Strate 1 APR 194	- 0			
Trace Disabled	* (가 : KSS KAS 🗰 원	Sinn 1 mm m	N (20 X 18			
Processor	×	+0000FEF9:	FFFF	222	Data or unknown opcode 🚽 Wowew		
Program Counter	0x00FC00	+0000FEFA	FFFF	222	Data or unknown opcode 🦾 🤹 🕼 🔄 USARTO 🔹 🦻		
Steck Pointer	0x21FF	+0000FEFC	FFFF	222	Data or unknown occode		
× pointer	0x0000	+0000FEFD:	FFFF	255	Date or unknown opcode E Salida Ro		
Y conter	0x0000	+0000FEFE:	FFFF	222	Data or unknown opcode		
7 pointer	0/0000	+0000FEPF:	FFFF	666	Data or unknown opcode Englosen i		
Fucie Founter	0	+0000FC01	FFFF	222	Data or unknown opcode		
Esemente		+0000FC02:	FFFF	222	Data or unknown opcode		
Stop Watch		+0000FC03:	FFFF	222	Date or unknown opcode		
Prop March		+0000FC04	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	222	Lata or unknown opcode		
Di anidara		+0000FC06	FFFF	555	Data or unknown opcode		
Hegisters		+0000FC07	FFFF	222	Data or unknown opcode		
		+0000FC08	FFFF	222	Data or unknown opcode		
		+0000FC09:	FFFF	222	Data or uninovn opcode E 2 PORTF		
		+000 OFCOA:	FFFF	777			
		+0000FC0C:	PPPP	222	Date or unknown opcode E S TIMEP_COUNTER_D		
		+0000FC0D	FFFF	222	Data cr unknown opcode IBI R TIMER COUNTER 2		
		+0000FC0E:	FFFF	222	Data or unknown opcode		
		+0000FC0F:	FFFF	222	Date or unincovn opcode		
		+0000FC10	h b b b b	222			
		+0000FC12:	FFFF	222	Data or unknown opcode more counter a		
		+0000FC13:	FFFF	222	Data or unknown opcode		
		+0000FC14:	FFFF	222	Data cr unknown opcode		
		+000.0501.6	FFFF	222	Date or unknown opcode		
		+0000FC15	FFFF	222	Data cr unknown opcode		
		+00000FC18	FFFF	222	Date or unknown opcode		
		+0000FC19:	FFFF	222	Data cr unknown opcode B DAD_CONVERTER		
		+0000FC1Å:	FFFF	222	Data or unknown opcode FIERONI IOAD		
		+0000FC1E	PPPP	222	Data or unknown opcode		
		+0000FCIC:	FFFF	222	Data cr uninovn opcode Nalie Addess Valie Div		
		+0000FC1F	1111	222			
		+0000FC1F:	FFFF	222	Data cr unknown opcode WOF UxUU		
		+0000FC20:	FFFF	222	Data cr unknown opcode WDE 0x00		
		+0000FC21:	PPPP	222	Data or unknown opcode WOP 0x00		
		+0000FC22	FFFF	222	Data cr unknown opcode WDCE Ux00 = =		
		+0000FC24	FFFF	222			
		+0000FC25:	FFFF	222	Data or unknown opcode		
		+0000FC26:	FFFF	222	Data or unknown opcode		
		+0000FC2?	FFFF	222	Date or unknown opcode		
		+0000FC28.	FFFF				
					<u>•</u>		
AVE OCC Proce	sor	Crizckirelease(AP)	1'SampleApp) catio	n)DebugTest/Debu	usTethe Disassembler 4 b		
		G • ()• ()					
lescage					x		
acc plup-in: CProte	ctTree:: 0r5ize()						
🕈 ant physics (Desir	etTrees OrGine/3 (Denied)Tre	- un Die	- 454 plune - 9		-		
 graphing instance graphing instance 	dTreeu OrSine()		- n.n., m.g.a.: = 7				
 got pagninis Crroje ant physics Crroje 	at maan on beau at Transis On Sing () ("Des to the Trans		420 June - 2				
- gar pugen: CProp	scribel: Unicely undjectire	eonsize: , cx=zo4, cy =	- ozo, mype = a				
got plug-in: CProje	ct iree:: 0fSbe()						
 gcc plug-In: CProje 	ctTree:: OnSize() CProjectTre	e:::0nsize: , cx=255, cy =	- 628, nType - 9				
got plug-in: CProje	• got plug-in: Chroled Tree:: OnSize()						
🗣 gox plug-in: CProjectTree:: OnSize() CProjectTree:: OnSize(), cx=254, cy = 628, nType = 9							
<u> </u>					N N N N N N N N N N N N N N N N N N N		
🖃 Build 🕕 Messiad	e 🔜 Find in Files 🚺 🔼 Brea	kpoints and Triacepoints					
					Afmegal281 JTAGCErrkII USB Scopped 😁 CAPINUM JCRL		



Cierre esta ventana y presione F5 para continuar la ejecución. AVR Studio pausará la ejecución en el primer *breakpoint* del código (ver Figura 7).







Figura 7: Depurando

En el menú View, puede seleccionar adicionales ventanas de depuración: procesador, E/S, desensamblador, vigilancia de variables, memoria, registros, etc. Pulsando con el botón derecho del ratón en el editor de código, podrá disponer también de unas pocas opciones de depuración, como añadir y eliminar *brakpoints* y expresiones de vigilancia.

Puede continuar con la ejecución en cualquier momento pulsando F5. Si pulsa Shift-F5, reiniciará la sesión de depuración.

Para detener el depurador, seleccione Debug -> Stop debugging del menú o pulse Ctrl-Shift-F5. Este comando está sólo disponible en modo *stop*, es decir, deberá pausar primero la ejecución y después detenerlo completamente.