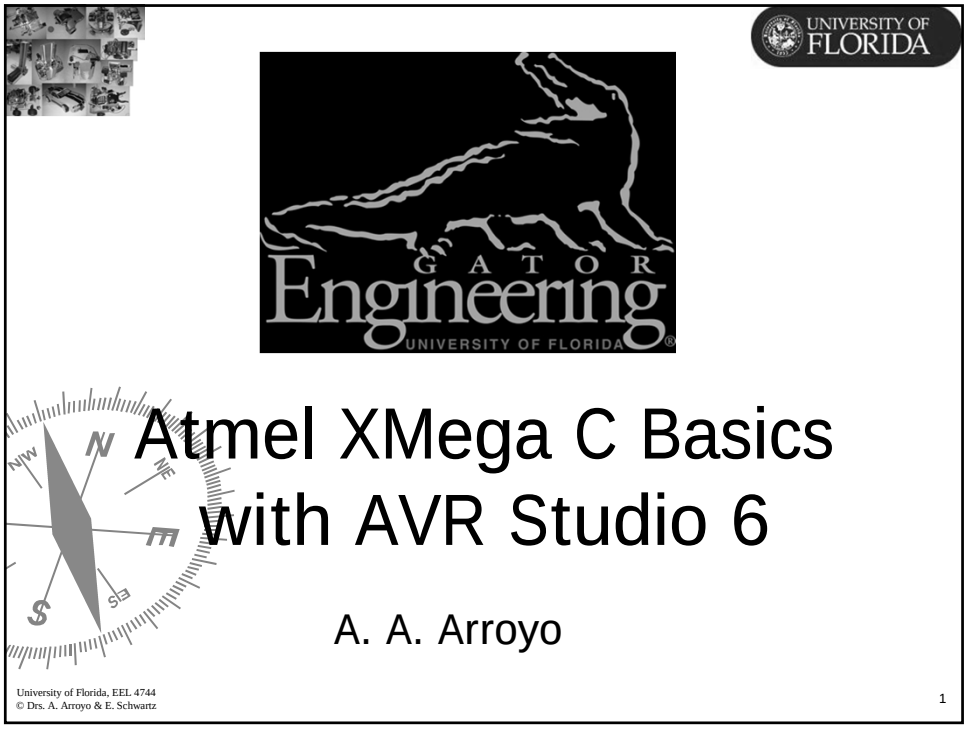
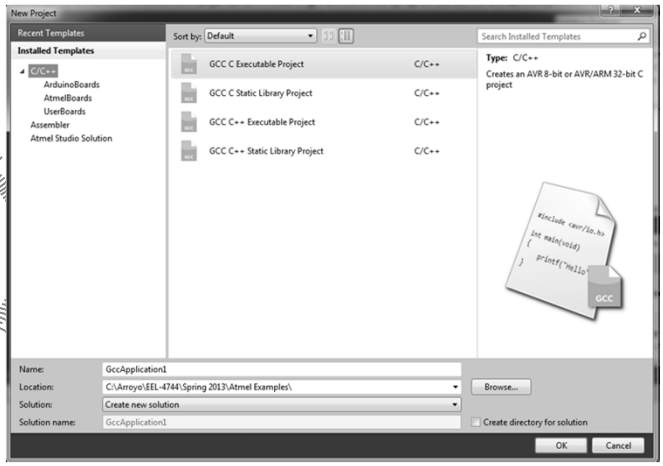

UNIVERSITY OF FLORIDA
GATOR Engineering
UNIVERSITY OF FLORIDA
Atmel XMega C Basics with AVR Studio 6
A. A. Arroyo
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
1

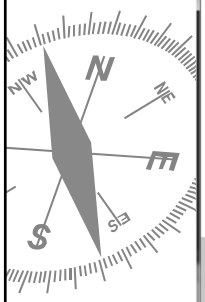

UNIVERSITY OF FLORIDA
AVR Studio 6 & GCC
Creating new C Projects using AVR Studio 6 for the Epiphany DIY/uTinkerer Board(s)
Step 1: Select Open New Project on the AVR Studio 6 Opening Screen or select New>Project from the File Menu (ctrl+shift+n)
Start Page - AtmelStudio
ATMEL STUDIO 6
Get Started
Tools Help
Latest News
New Project...
New Example Project from ASF...
Open Project...
Welcome
Links and Resources
Welcome to Atmel Studio
Get to know Atmel Studio.
Close page after project load
Show page on startup
Solution Explorer
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
2

UNIVERSITY OF FLORIDA

AVR Studio 6 & GCC

Step 2: Select GCC Executable Project and enter a name for your project and select a directory for your project in the New Project Window. Hit OK when ready.

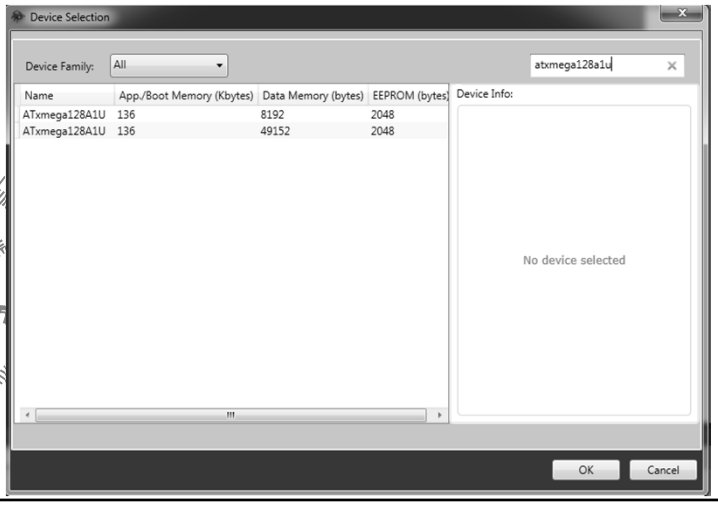


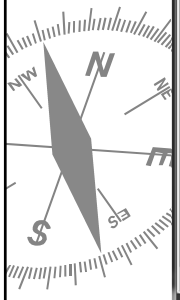


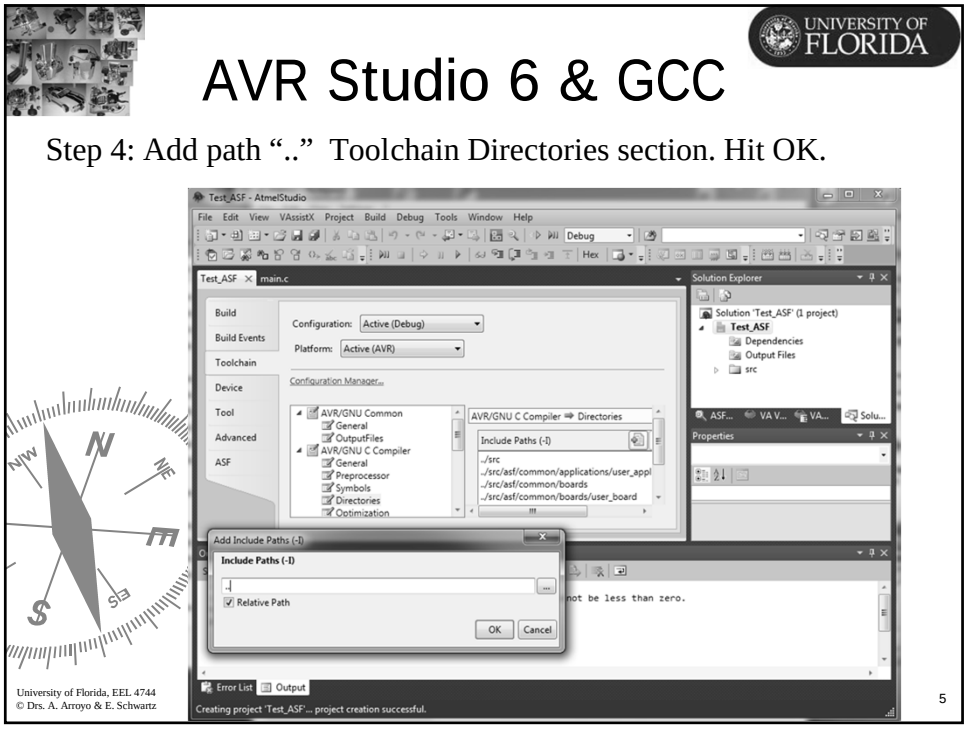
UNIVERSITY OF FLORIDA

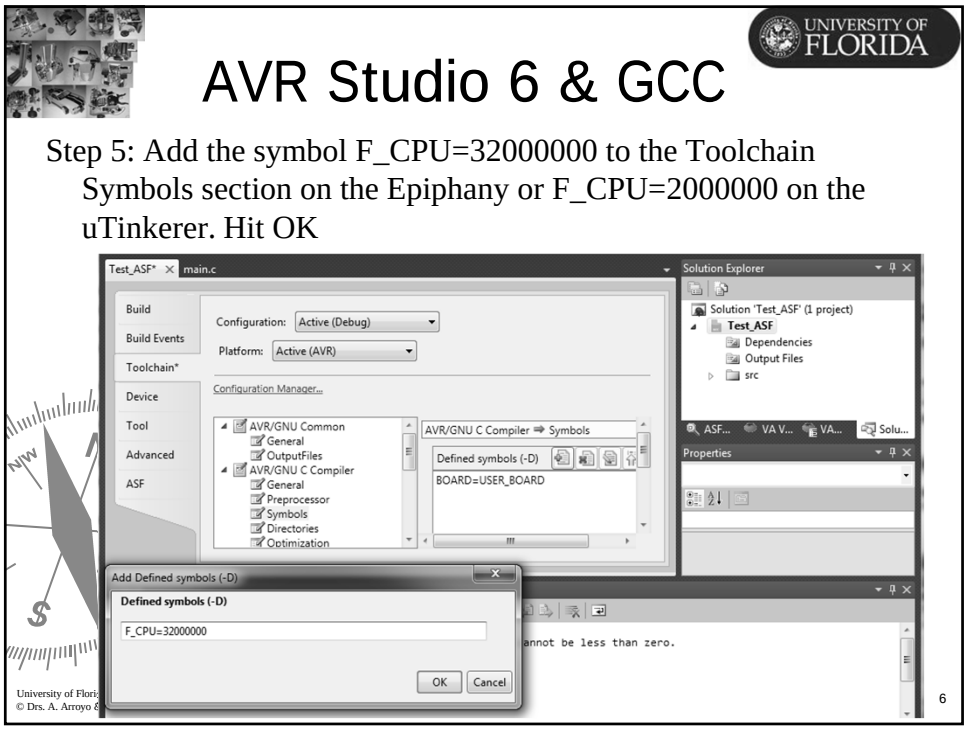
AVR Studio 6 & GCC

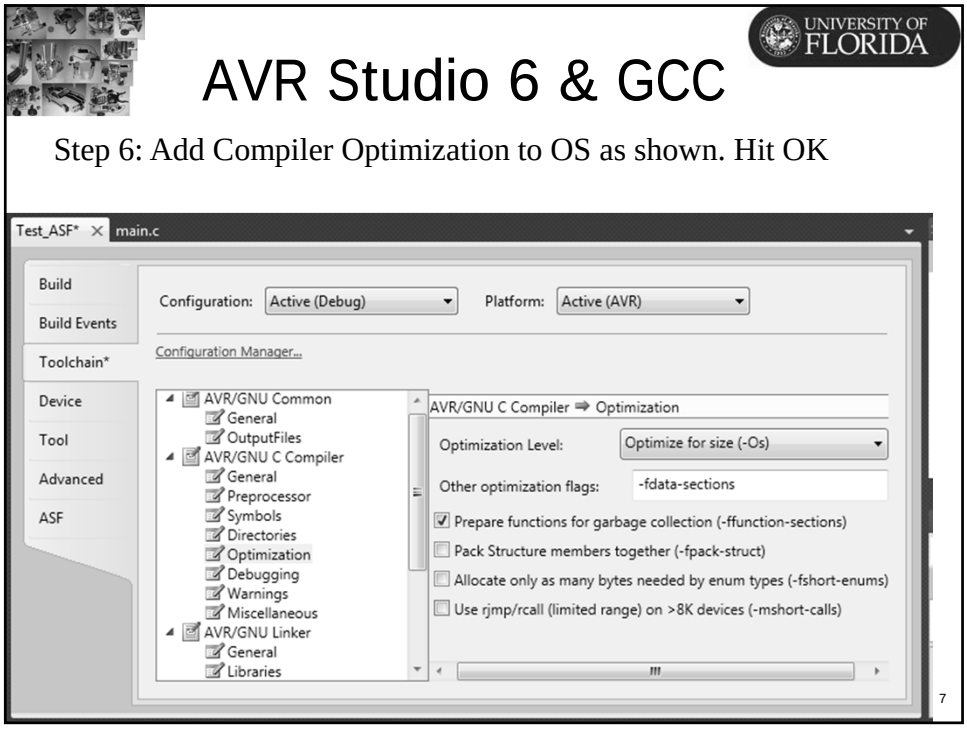
Step 3: Select ATxmega128A1U in the device selection window (Epiphany DIY/uTinkerer uses the ATxmega128A1U). Hit OK.

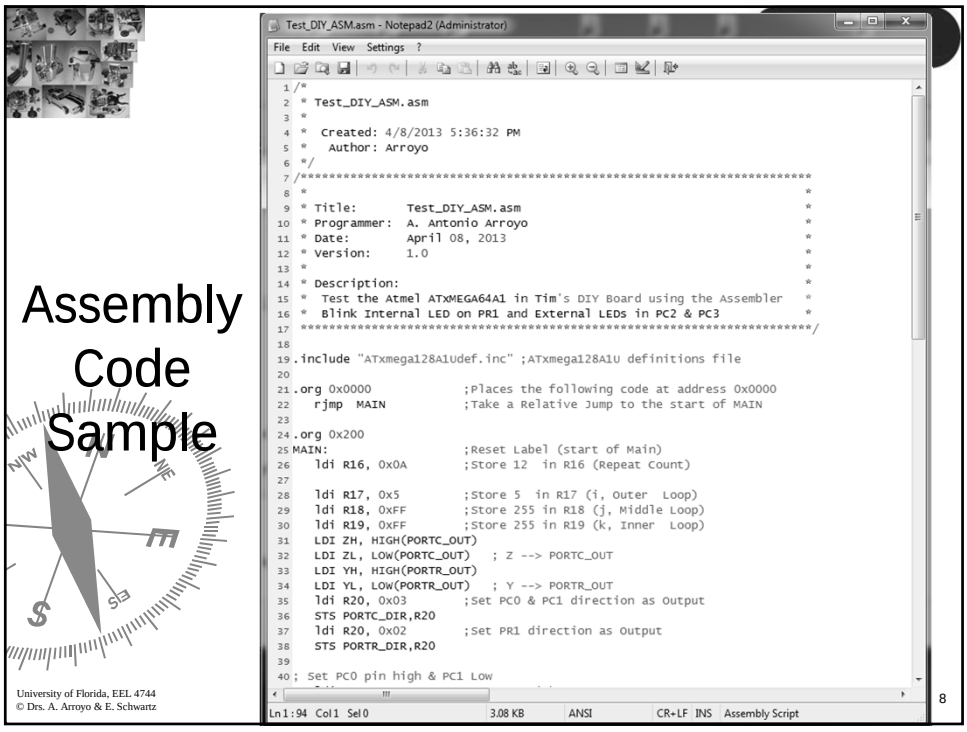


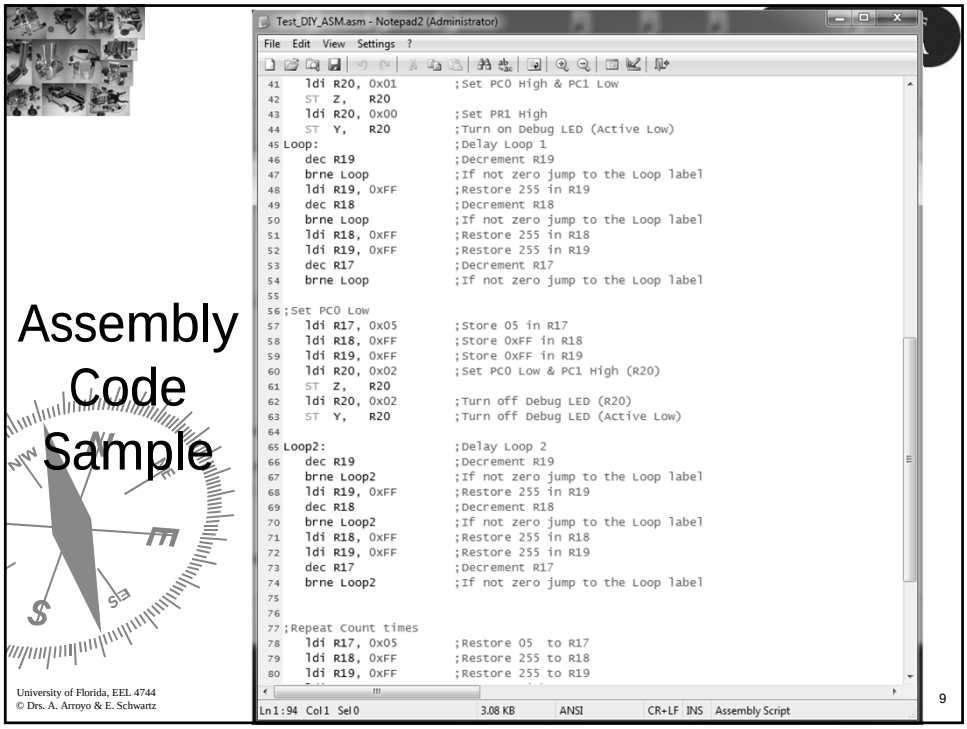


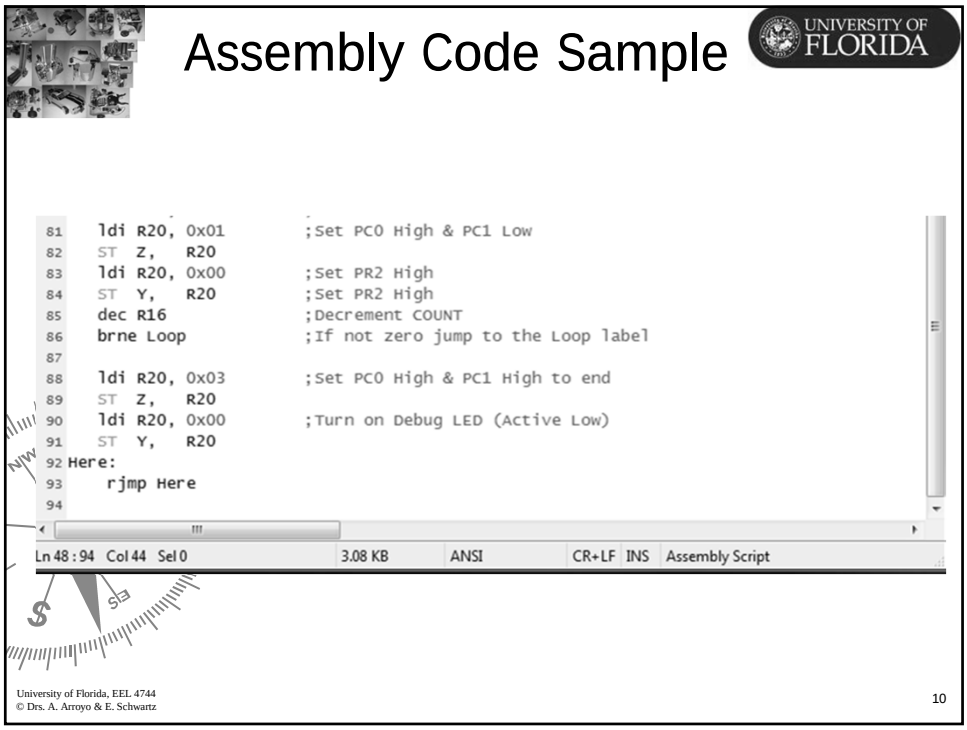

AVR Studio 6 & GCC
UNIVERSITY OF FLORIDA
Step 4: Add path ".." Toolchain Directories section. Hit OK.
Test_ASF - AtmelStudio
Build
Build Events
Toolchain
Device
Tool
Advanced
ASF
Configuration: Active (Debug)
Platform: Active (AVR)
Configuration Manager...
AVR/GNU Common
General
OutputFiles
AVR/GNU C Compiler
General
Preprocessor
Symbols
Directories
Optimization
AVR/GNU C Compiler ⇒ Directories
Include Paths (-I)
../src
../src/asf/common/applications/user_appl
../src/asf/common/boards
../src/asf/common/boards/user_board
Add Include Paths (-I)
Include Paths (-I)
Relative Path
OK
Cancel
Solution Explorer
Solution 'Test_ASF' (1 project)
Test_ASF
Dependencies
Output Files
src
Properties
Error List
Output
Creating project 'Test_ASF'... project creation successful.
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
5

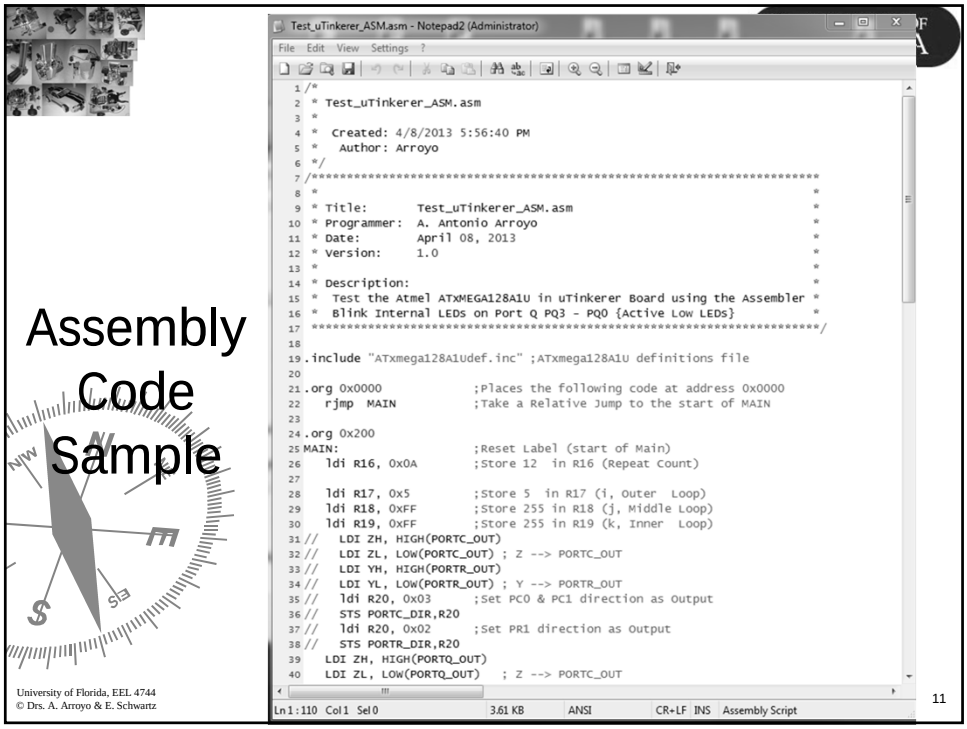

AVR Studio 6 & GCC
UNIVERSITY OF FLORIDA
Step 5: Add the symbol F_CPU=32000000 to the Toolchain Symbols section on the Epiphany or F_CPU=2000000 on the uTinkerer. Hit OK
Test_ASF*
main.c
Build
Build Events
Toolchain*
Device
Tool
Advanced
ASF
Configuration: Active (Debug)
Platform: Active (AVR)
Configuration Manager...
AVR/GNU Common
General
OutputFiles
AVR/GNU C Compiler
General
Preprocessor
Symbols
Directories
Optimization
AVR/GNU C Compiler ⇒ Symbols
Defined symbols (-D)
BOARD=USER_BOARD
Add Defined symbols (-D)
Defined symbols (-D)
F_CPU=32000000
OK
Cancel
Solution Explorer
Solution 'Test_ASF' (1 project)
Test_ASF
Dependencies
Output Files
src
Properties
6

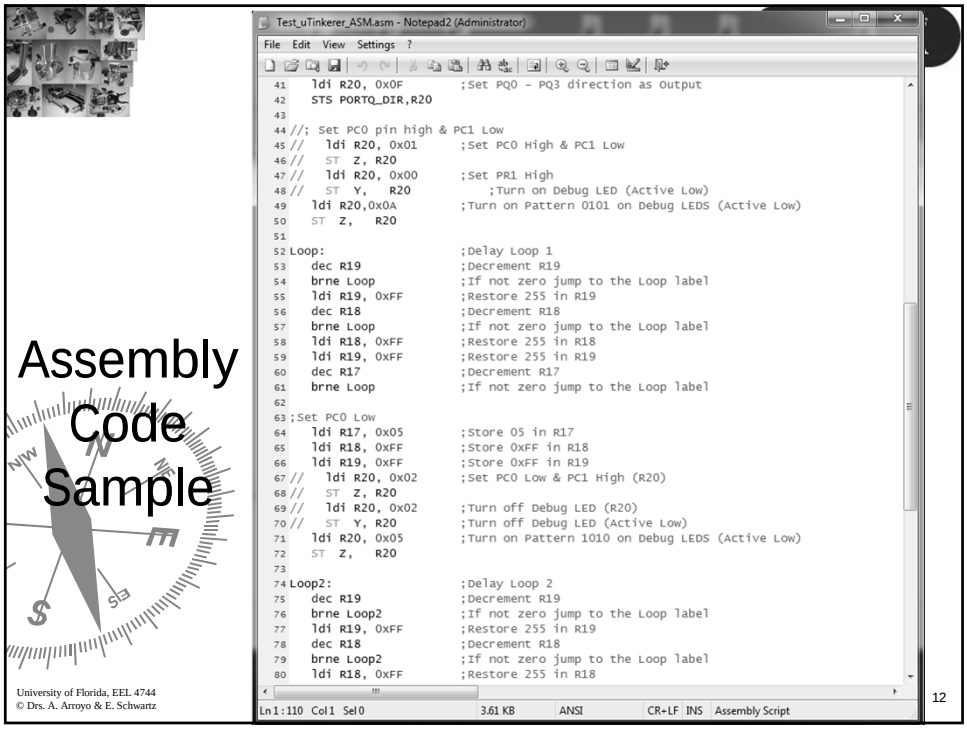

UNIVERSITY OF FLORIDA
AVR Studio 6 & GCC
Step 6: Add Compiler Optimization to OS as shown. Hit OK
Test_ASF*
main.c
Build
Build Events
Toolchain*
Device
Tool
Advanced
ASF
Configuration: Active (Debug)
Platform: Active (AVR)
Configuration Manager...
AVR/GNU Common
General
OutputFiles
AVR/GNU C Compiler
General
Preprocessor
Symbols
Directories
Optimization
Debugging
Warnings
Miscellaneous
AVR/GNU Linker
General
Libraries
AVR/GNU C Compiler ➡ Optimization
Optimization Level: Optimize for size (-Os)
Other optimization flags: -fdata-sections
Prepare functions for garbage collection (-ffunction-sections)
Pack Structure members together (-fpack-struct)
Allocate only as many bytes needed by enum types (-fshort-enums)
Use rjmp/rcall (limited range) on >8K devices (-mshort-calls)
7

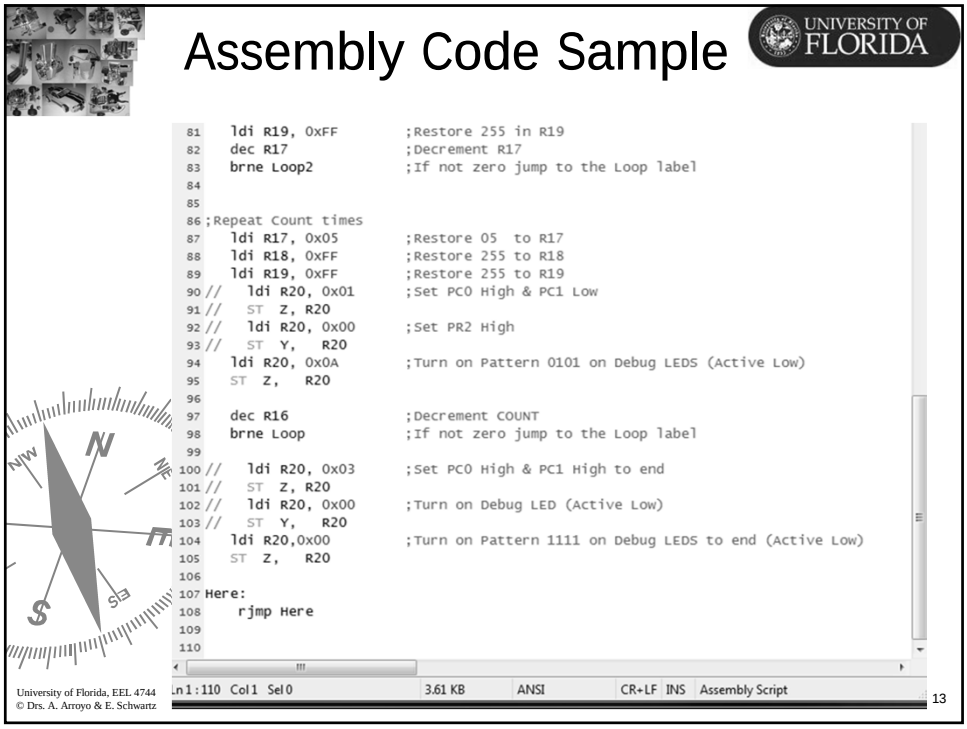

Assembly Code Sample
Test_DIY_ASM.asm - Notepad2 (Administrator)
File Edit View Settings ?
/*
* Test_DIY_ASM.asm
*
* Created: 4/8/2013 5:36:32 PM
* Author: Arroyo
*/
* Title: Test_DIY_ASM.asm
* Programmer: A. Antonio Arroyo
* Date: April 08, 2013
* Version: 1.0
* Description:
* Test the Atmel ATxMEGA64A1 in Tim's DIY Board using the Assembler
* Blink Internal LED on PR1 and External LEDs in PC2 & PC3
.include "ATxmega128A1udef.inc" ;ATxmega128A1U definitions file
.org 0x0000 ;Places the following code at address 0x0000
rjmp MAIN ;Take a Relative Jump to the start of MAIN
.org 0x200
MAIN: ;Reset Label (start of Main)
ldi R16, 0x0A ;Store 12 in R16 (Repeat Count)
ldi R17, 0x5 ;Store 5 in R17 (i, Outer Loop)
ldi R18, 0xFF ;Store 255 in R18 (j, Middle Loop)
ldi R19, 0xFF ;Store 255 in R19 (k, Inner Loop)
LDI ZH, HIGH(PORTC_OUT)
LDI ZL, LOW(PORTC_OUT) ; Z --> PORTC_OUT
LDI YH, HIGH(PORTR_OUT)
LDI YL, LOW(PORTR_OUT) ; Y --> PORTR_OUT
ldi R20, 0x03 ;Set PC0 & PC1 direction as Output
STS PORTC_DIR,R20
ldi R20, 0x02 ;Set PR1 direction as Output
STS PORTR_DIR,R20
; Set PC0 pin high & PC1 Low
Ln 1 : 94 Col 1 Sel 0 3.08 KB ANSI CR+LF INS Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
8

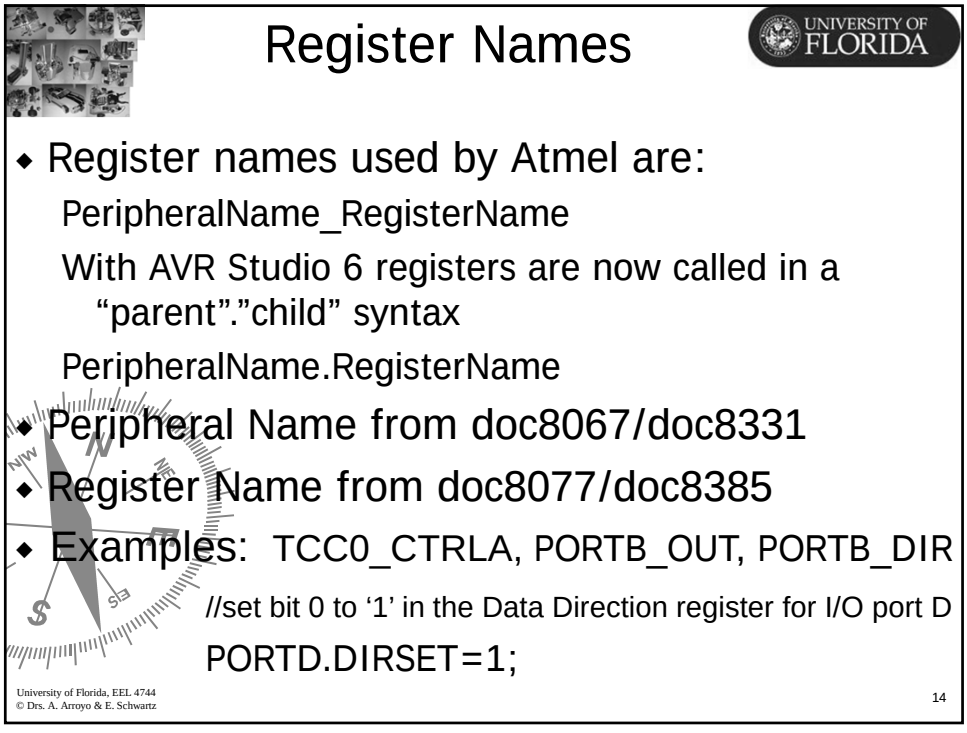

Assembly Code Sample
Test_DIY_ASM.asm - Notepad2 (Administrator)
File Edit View Settings ?
41 ldi R20, 0x01 ;Set PC0 High & PC1 Low
42 ST Z, R20
43 ldi R20, 0x00 ;Set PR1 High
44 ST Y, R20 ;Turn on Debug LED (Active Low)
45 Loop: ;Delay Loop 1
46 dec R19 ;Decrement R19
47 brne Loop ;If not zero jump to the Loop label
48 ldi R19, 0xFF ;Restore 255 in R19
49 dec R18 ;Decrement R18
50 brne Loop ;If not zero jump to the Loop label
51 ldi R18, 0xFF ;Restore 255 in R18
52 ldi R19, 0xFF ;Restore 255 in R19
53 dec R17 ;Decrement R17
54 brne Loop ;If not zero jump to the Loop label
55
56 ;Set PC0 Low
57 ldi R17, 0x05 ;Store 05 in R17
58 ldi R18, 0xFF ;Store 0xFF in R18
59 ldi R19, 0xFF ;Store 0xFF in R19
60 ldi R20, 0x02 ;Set PC0 Low & PC1 High (R20)
61 ST Z, R20
62 ldi R20, 0x02 ;Turn off Debug LED (R20)
63 ST Y, R20 ;Turn off Debug LED (Active Low)
64
65 Loop2: ;Delay Loop 2
66 dec R19 ;Decrement R19
67 brne Loop2 ;If not zero jump to the Loop label
68 ldi R19, 0xFF ;Restore 255 in R19
69 dec R18 ;Decrement R18
70 brne Loop2 ;If not zero jump to the Loop label
71 ldi R18, 0xFF ;Restore 255 in R18
72 ldi R19, 0xFF ;Restore 255 in R19
73 dec R17 ;Decrement R17
74 brne Loop2 ;If not zero jump to the Loop label
75
76
77 ;Repeat Count times
78 ldi R17, 0x05 ;Restore 05 to R17
79 ldi R18, 0xFF ;Restore 255 to R18
80 ldi R19, 0xFF ;Restore 255 to R19
Ln 1 : 94 Col 1 Sel 0
3.08 KB
ANSI
CR+LF INS Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
9


Assembly Code Sample
UNIVERSITY OF FLORIDA
81 ldi R20, 0x01 ;Set PC0 High & PC1 Low
82 ST Z, R20
83 ldi R20, 0x00 ;Set PR2 High
84 ST Y, R20 ;Set PR2 High
85 dec R16 ;Decrement COUNT
86 brne Loop ;If not zero jump to the Loop label
87
88 ldi R20, 0x03 ;Set PC0 High & PC1 High to end
89 ST Z, R20
90 ldi R20, 0x00 ;Turn on Debug LED (Active Low)
91 ST Y, R20
92 Here:
93 rjmp Here
94
Ln 48 : 94 Col 44 Sel 0
3.08 KB
ANSI
CR+LF INS Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
10


Assembly Code Sample
Test_uTinkerer_ASM.asm - Notepad2 (Administrator)
File Edit View Settings ?
1 /*
2 * Test_uTinkerer_ASM.asm
3 *
4 * Created: 4/8/2013 5:56:40 PM
5 * Author: Arroyo
6 */
7 /***
8 * *
9 * Title: Test_uTinkerer_ASM.asm *
10 * Programmer: A. Antonio Arroyo *
11 * Date: April 08, 2013 *
12 * Version: 1.0 *
13 * *
14 * Description: *
15 * Test the Atmel ATxMEGA128A1U in uTinkerer Board using the Assembler *
16 * Blink Internal LEDs on Port Q PQ3 - PQ0 {Active Low LEDs} *
17 ***/
18
19 .include "ATxmega128A1Udef.inc" ;ATxmega128A1U definitions file
20
21 .org 0x0000 ;Places the following code at address 0x0000
22 rjmp MAIN ;Take a Relative Jump to the start of MAIN
23
24 .org 0x200
25 MAIN: ;Reset Label (start of Main)
26 ldi R16, 0x0A ;Store 12 in R16 (Repeat Count)
27
28 ldi R17, 0x5 ;Store 5 in R17 (i, Outer Loop)
29 ldi R18, 0xFF ;Store 255 in R18 (j, Middle Loop)
30 ldi R19, 0xFF ;Store 255 in R19 (k, Inner Loop)
31 // LDI ZH, HIGH(PORTC_OUT)
32 // LDI ZL, LOW(PORTC_OUT) ; Z --> PORTC_OUT
33 // LDI YH, HIGH(PORTR_OUT)
34 // LDI YL, LOW(PORTR_OUT) ; Y --> PORTR_OUT
35 // ldi R20, 0x03 ;Set PC0 & PC1 direction as Output
36 // STS PORTC_DIR,R20
37 // ldi R20, 0x02 ;Set PR1 direction as Output
38 // STS PORTR_DIR,R20
39 LDI ZH, HIGH(PORTQ_OUT)
40 LDI ZL, LOW(PORTQ_OUT) ; Z --> PORTC_OUT
Ln 1 : 110 Col 1 Sel 0 3.61 KB ANSI CR+LF INS Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
11

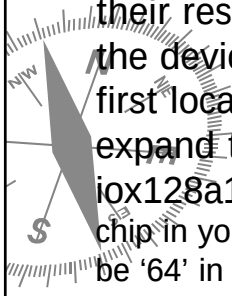

Assembly Code Sample
Test_uTinkerer_ASM.asm - Notepad2 (Administrator)
File Edit View Settings ?
41 ldi R20, 0x0F ;Set PQ0 - PQ3 direction as Output
42 STS PORTQ_DIR,R20
43
44 //; Set PC0 pin high & PC1 Low
45 // ldi R20, 0x01 ;Set PC0 High & PC1 Low
46 // ST Z, R20
47 // ldi R20, 0x00 ;Set PR1 High
48 // ST Y, R20 ;Turn on Debug LED (Active Low)
49 ldi R20,0x0A ;Turn on Pattern 0101 on Debug LEDS (Active Low)
50 ST Z, R20
51
52 Loop: ;Delay Loop 1
53 dec R19 ;Decrement R19
54 brne Loop ;If not zero jump to the Loop label
55 ldi R19, 0xFF ;Restore 255 in R19
56 dec R18 ;Decrement R18
57 brne Loop ;If not zero jump to the Loop label
58 ldi R18, 0xFF ;Restore 255 in R18
59 ldi R19, 0xFF ;Restore 255 in R19
60 dec R17 ;Decrement R17
61 brne Loop ;If not zero jump to the Loop label
62
63 ;Set PC0 Low
64 ldi R17, 0x05 ;Store 05 in R17
65 ldi R18, 0xFF ;Store 0xFF in R18
66 ldi R19, 0xFF ;Store 0xFF in R19
67 // ldi R20, 0x02 ;Set PC0 Low & PC1 High (R20)
68 // ST Z, R20
69 // ldi R20, 0x02 ;Turn off Debug LED (R20)
70 // ST Y, R20 ;Turn off Debug LED (Active Low)
71 ldi R20, 0x05 ;Turn on Pattern 1010 on Debug LEDS (Active Low)
72 ST Z, R20
73
74 Loop2: ;Delay Loop 2
75 dec R19 ;Decrement R19
76 brne Loop2 ;If not zero jump to the Loop label
77 ldi R19, 0xFF ;Restore 255 in R19
78 dec R18 ;Decrement R18
79 brne Loop2 ;If not zero jump to the Loop label
80 ldi R18, 0xFF ;Restore 255 in R18
Ln 1 : 110 Col 1 Sel 0 3.61 KB ANSI CR+LF INS Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
12


Assembly Code Sample
UNIVERSITY OF FLORIDA
81 ldi R19, 0xFF ;Restore 255 in R19
82 dec R17 ;Decrement R17
83 brne Loop2 ;If not zero jump to the Loop label
84
85
86 ;Repeat Count times
87 ldi R17, 0x05 ;Restore 05 to R17
88 ldi R18, 0xFF ;Restore 255 to R18
89 ldi R19, 0xFF ;Restore 255 to R19
90 // ldi R20, 0x01 ;Set PC0 High & PC1 Low
91 // ST Z, R20
92 // ldi R20, 0x00 ;Set PR2 High
93 // ST Y, R20
94 ldi R20, 0x0A ;Turn on Pattern 0101 on Debug LEDS (Active Low)
95 ST Z, R20
96
97 dec R16 ;Decrement COUNT
98 brne Loop ;If not zero jump to the Loop label
99
100 // ldi R20, 0x03 ;Set PC0 High & PC1 High to end
101 // ST Z, R20
102 // ldi R20, 0x00 ;Turn on Debug LED (Active Low)
103 // ST Y, R20
104 ldi R20,0x00 ;Turn on Pattern 1111 on Debug LEDS to end (Active Low)
105 ST Z, R20
106
107 Here:
108 rjmp Here
109
110
Ln 1 : 110 Col 1 Sel 0
3.61 KB
ANSI
CR+LF
INS
Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
13


Register Names
UNIVERSITY OF FLORIDA
Register names used by Atmel are:
PeripheralName_RegisterName
With AVR Studio 6 registers are now called in a "parent"."child" syntax
PeripheralName.RegisterName
Peripheral Name from doc8067/doc8331
Register Name from doc8077/doc8385
Examples: TCC0_CTRLA, PORTB_OUT, PORTB_DIR
//set bit 0 to '1' in the Data Direction register for I/O port D
PORTD.DIRSET=1;
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
14

Register Names

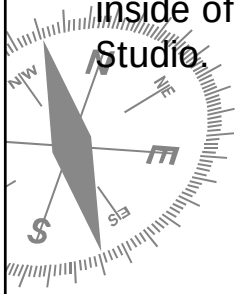
- Syntax/Hierarchy
  - When programming for any Atmel processor the required dependency is in <io.h>. This header file points to the device specific header containing definitions of all the registers, i.e., assignments to their respective memory map locations. To access the device specific IO header file in Atmel Studio first locate the “Solution Explorer” window. Now expand the “Dependencies” folder to find a file like iox128a1u.h. Note the exact name varies based upon the chip in your board. The ‘128’ is the memory size, and could be ‘64’ in some cases. The ‘a1’ or ‘a1u’ specifies the series.

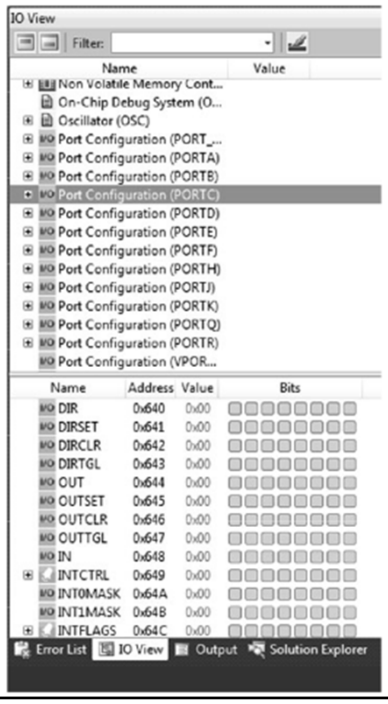


University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
15

Register Names

- Syntax/Hierarchy
  - The easiest method is to use the “IO View” inside of AVR/ATMEL Studio.





University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
16

Register Names

- Here is the register summary for the USART.
- Can see that serial communication uses 7 regs
  - 1 for Data USARTC0.DATA
  - 1 for status USARTC0.STATUS
  - 3 for control USARTC0.CTRL<A,B,C>
  - 2 for baud rate USARTC0.BAUDCTRL<A,B>

21.16 Register Summary

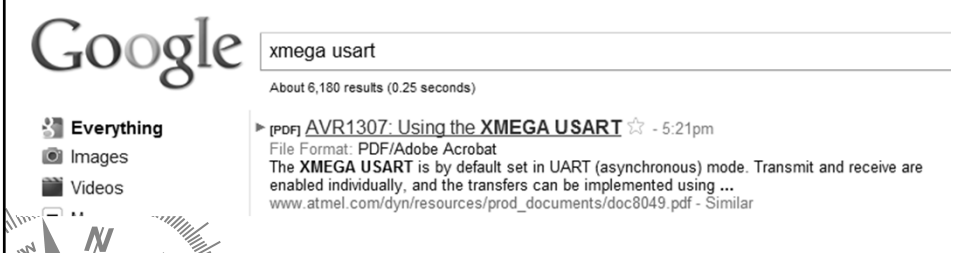
21.16.1 Register Description - USART

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DATA	DATA[7:0]								249
+0x01	STATUS	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	-	RXB8	249
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	CTRLA	-	-	RXCINTLVL[1:0]		TXCINTLVL[1:0]		DREINTLVL[1:0]		251
+0x04	CTRLB	-	-	-	RXEN	TXEN	CLK2X	MPCM	TXB8	251
+0x05	CTRLC	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]			253
+0x06	BAUDCTRLA	BSEL[7:0]								255
+0x07	BAUDCTRLB	BSCALE[3:0]				BSEL[11:8]				254

17

Find Examples

- Type "xmega usart" into Google



- The first result is a nice 7-page document explaining EXACTLY how to set up the serial USART on the Xmega. The source code for this example is easily found on the internet.

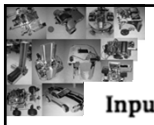
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
18

Sample Code

- Important concept: There are two ways to write/read a pin.
  - Direct pin access
  - Using a peripheral.
- When using direct pin access, use PORTX_IN to read and PORTX_OUT to write. Remember to set data direction register, PORTX_DIR.

Sample Code

- Use bit masking to access individual pins
  - PORTX_OUT |= 0b00001000; //sets pin 3 high
  - PORTX_OUT &= 0b11110111; //sets pin 3 low
- For input, you have to read entire port
  - if ((PORTJ_IN & 0b00010000) == 0b00010000) //evaluates as TRUE is bit 4 of PORTJ is high



Sample Code

UNIVERSITY OF FLORIDA

Input and Output

SET CLR, and TGL

When manipulating bits at in a register the conventional method is to use and/or instructions via bit-masking. The Xmega accelerates this process by having hardware set clear and toggle functions.

- A '1' in a SET register bit will set the same bit in the host register
- A '1' in a CLR register bit will clear the same bit in the host register
- A '1' in a TGL register bit will toggle the same bit in the host register

Example2:
Traditional bitmask

PORTD.OUT |= 1<<5; //sets bit 5. Requires a read, an or and a store instruction

Hardware bitmask

PORTD.OUTSET = 1<<5; //sets bit 5. Requires just a store instruction

Direction Registers

When using an i/o pin the first priority is choosing whether that pin is an input or an output. This is done via the direction register associated with the respective port. The convention for AVR processors is '1' is for output and '0' is for input. It's also worth knowing that by default every pin is set as an input. This prevents bus conflicts due to the chip being unprogrammed.

Example3:
PORTC.DIR = 1<<3;//port C pin 3 is now an output, while the rest of the port is an input.

21

Sample Code

UNIVERSITY OF FLORIDA

Output Register

The Output register is used for controlling the voltage level of an output pin. The relationship is high true.

Example4:
PORTC.DIRSET = 0xFF;//set the port to all outputs

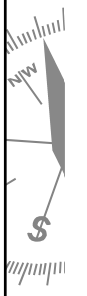
PORTC.OUTCLR = 0x55;//send out hex '55' to port C

Input Register

The input register is used to read the data in from a pin/port.

Example5:
PORTE.DIRCLR = 0x00;//set all of pins of port E to inputs

If(PORTE.IN & 1<<2)someFunctionCall();//This is an example of not only reading but masking a bit



Configuring Ports

Each pin of the Xmega has pull-up and pull-down resistors available, as well as other useful features. Manipulating the PIN#CTRL registers these features can be harnessed. For details on all these features consult the device datasheets.

Example6:
PORTF.PIN5CTRL = 0x10;//enables a pull-down resistor on pin 5 of port C

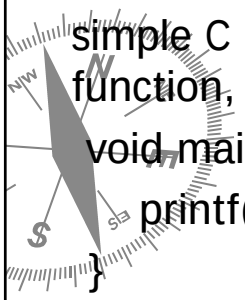
PORTF.PIN6CTRL = 0x18;//enables a pull-up resistor on pin 6 of port C

22

Writing C Programs

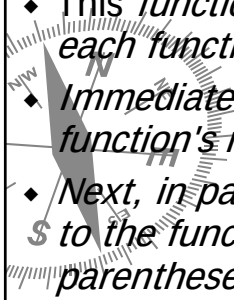
- C Tutorial http://www.physics.drexel.edu/courses/Comp_Phys/General/C_basics/
- Most C programs consist of function definitions and data structures. Here is a simple C program that defines a single function, called main.

```
void main() {
    printf("Hello, world!\n");
}
```



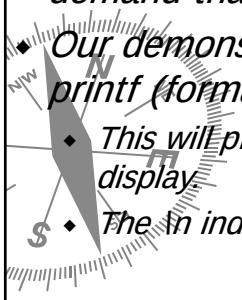
Writing C Programs

- All functions must have a return type.
- Since main does not return a value, it uses void, the null type, as its return type.
- Other types include integers (int) and floating point numbers (float).
- This *function declaration information must precede each function definition.*
- *Immediately following the function declaration is the function's name (in this case, main).*
- *Next, in parentheses, are any arguments (or inputs) to the function. main has none, but an empty set of parentheses is still required.*



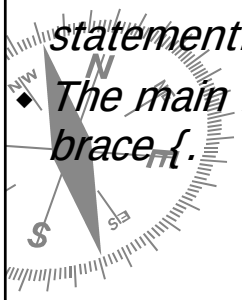
Writing C Programs

- *After the function arguments is an open curly-brace {.*
  - *This signifies the start of the actual function code. Curly-braces signify program blocks, or chunks of code.*
- *Next comes a series of C statements. Statements demand that some action be taken.*
- *Our demonstration program has a single statement, a printf (formatted print).*
  - *This will print the message "Hello, world!" to the LCD/CRT display.*
  - *The \n indicates end-of-line.*



Writing C Programs

- *The printf statement ends with a semicolon (;). All C statements must be ended by a semicolon. Beginning C programmers commonly make the error of omitting the semicolon that is required at the end of each statement.*
- *The main function is ended by the close curly-brace }.*

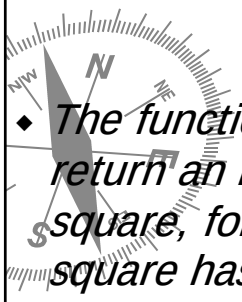


```
void main() {
    printf("Hello, world!\n");
}
```

Writing C Programs

- The following code defines the function *square, which returns the mathematical square of a number.*

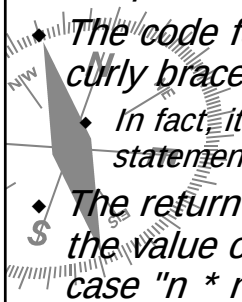
```
int square(int n) {
    return(n * n);
}
```



- *The function is declared as type int, which will return an integer value. Next the function name square, followed by its argument list inside (). square has one argument, n, an integer.*

Writing C Programs

- *When a function has arguments declared, those argument variables are valid within the "scope" of the function*
  - *They only have meaning within the function's own code.*
- *Other functions may use the same variable names independently*
- *The code for square is contained within the set of curly braces.*
  - *In fact, it consists of a single statement: the return statement.*
- *The return statement exits the function and returns the value of the C expression that follows it (in this case "n * n").*

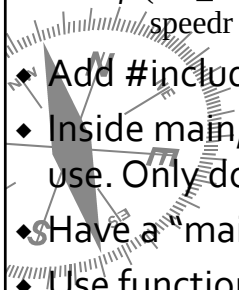


Writing C Programs

UNIVERSITY OF FLORIDA

- Software in a µ*P* can have many different styles of code and program design.
  - This program model is purely a suggestion, but it has been proven to be reliable.
- Simple *if* statements

```
if  (left_ir >125)    /* If the Left IR indicates we are close */
    speedr = -100;  /* to an obstacle, turn right  */
```

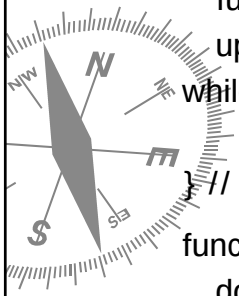


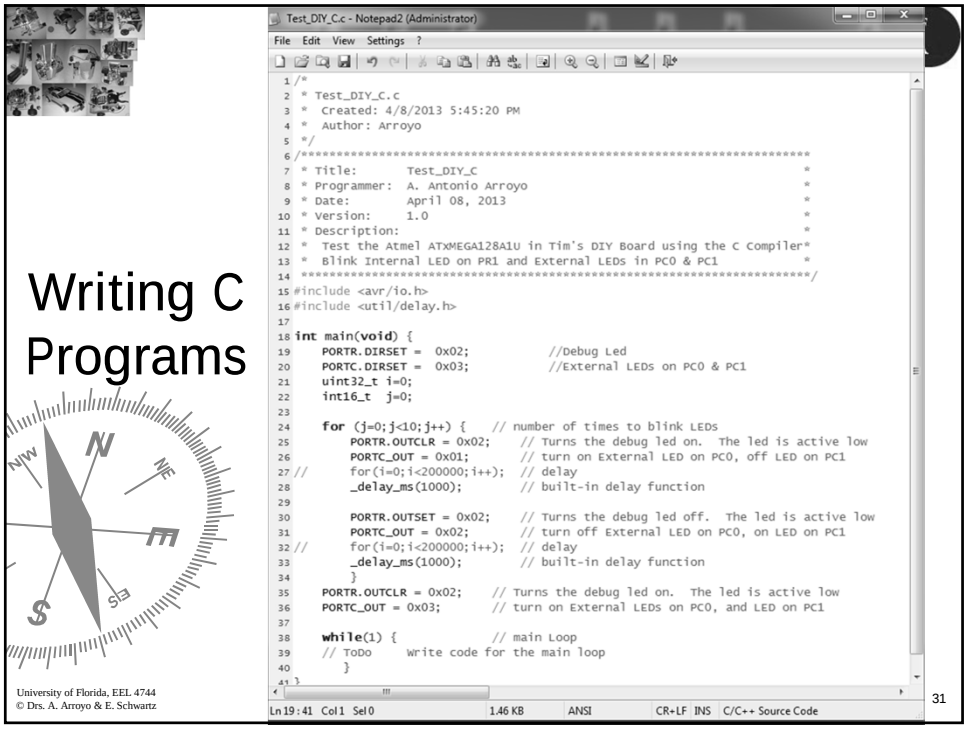
- Add #includes and function definitions.
- Inside main, start by initializing all peripherals you will use. Only do this once.
- Have a "main loop" that runs forever
- Use functions outside of main for specific requirements

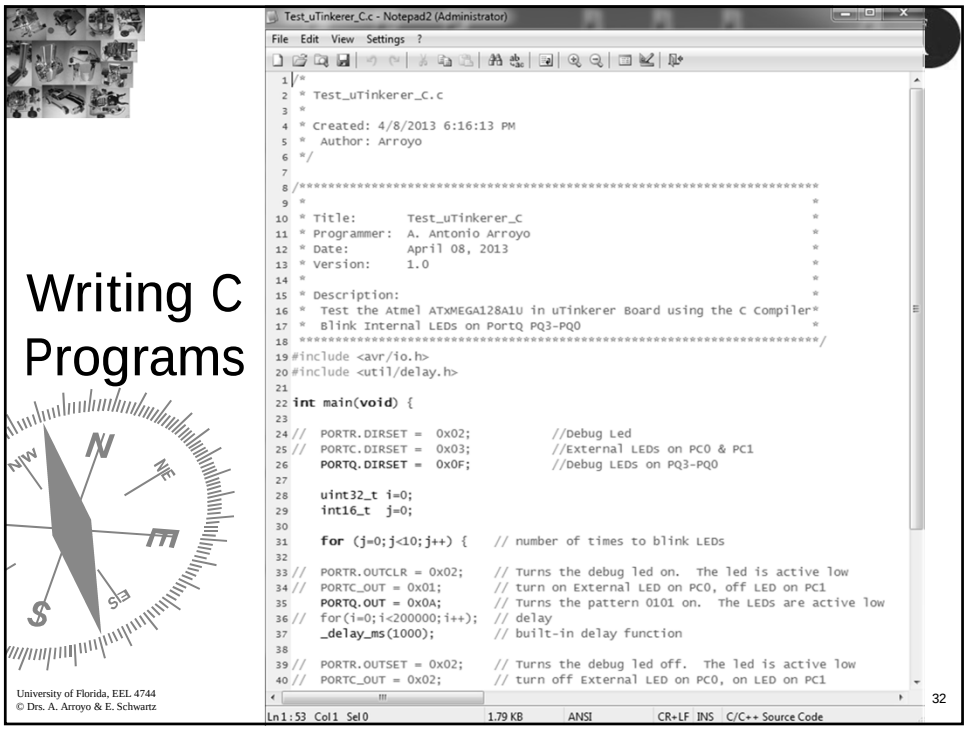
Writing C Programs

UNIVERSITY OF FLORIDA

```
#includes
void function1(int);
main(){
   Initialize Ports, PortX_DIR, LCD, Servos, PWM,
        A/D, Serial, variables
   Requirement #1, #2, etc.
   function1();
   update LCD;
while(1){
        } // while
} // end of main

function1(int args) {
   do stuff } // end of function 1
```




Writing C Programs
Test_DIY_C.c - Notepad2 (Administrator)
File Edit View Settings ?
/*
* Test_DIY_C.c
* Created: 4/8/2013 5:45:20 PM
* Author: Arroyo
*/
/***
* Title: Test_DIY_C *
* Programmer: A. Antonio Arroyo *
* Date: April 08, 2013 *
* Version: 1.0 *
* Description: *
* Test the Atmel ATxMEGA128A1U in Tim's DIY Board using the C Compiler*
* Blink Internal LED on PR1 and External LEDs in PC0 & PC1 *
***/
#include <avr/io.h>
#include <util/delay.h>
int main(void) {
PORTR.DIRSET = 0x02; //Debug Led
PORTC.DIRSET = 0x03; //External LEDs on PC0 & PC1
uint32_t i=0;
int16_t j=0;
for (j=0;j<10;j++) { // number of times to blink LEDs
PORTR.OUTCLR = 0x02; // Turns the debug led on. The led is active low
PORTC_OUT = 0x01; // turn on External LED on PC0, off LED on PC1
// for(i=0;i<200000;i++); // delay
_delay_ms(1000); // built-in delay function
PORTR.OUTSET = 0x02; // Turns the debug led off. The led is active low
PORTC_OUT = 0x02; // turn off External LED on PC0, on LED on PC1
// for(i=0;i<200000;i++); // delay
_delay_ms(1000); // built-in delay function
}
PORTR.OUTCLR = 0x02; // Turns the debug led on. The led is active low
PORTC_OUT = 0x03; // turn on External LEDs on PC0, and LED on PC1
while(1) { // main Loop
// ToDo Write code for the main loop
}
Ln 19 : 41 Col 1 Sel 0
1.46 KB
ANSI
CR+LF INS C/C++ Source Code
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
31


Writing C Programs
Test_uTinkerer_C.c - Notepad2 (Administrator)
File Edit View Settings ?
/*
* Test_uTinkerer_C.c
*
* Created: 4/8/2013 6:16:13 PM
* Author: Arroyo
*/
/***
*
* Title: Test_uTinkerer_C *
* Programmer: A. Antonio Arroyo *
* Date: April 08, 2013 *
* Version: 1.0 *
*
* Description: *
* Test the Atmel ATxMEGA128A1U in uTinkerer Board using the C Compiler*
* Blink Internal LEDs on PortQ PQ3-PQ0 *
***/
#include <avr/io.h>
#include <util/delay.h>
int main(void) {
// PORTR.DIRSET = 0x02; //Debug Led
// PORTC.DIRSET = 0x03; //External LEDs on PC0 & PC1
PORTQ.DIRSET = 0x0F; //Debug LEDs on PQ3-PQ0
uint32_t i=0;
int16_t j=0;
for (j=0;j<10;j++) { // number of times to blink LEDs
// PORTR.OUTCLR = 0x02; // Turns the debug led on. The led is active low
// PORTC_OUT = 0x01; // turn on External LED on PC0, off LED on PC1
PORTQ.OUT = 0x0A; // Turns the pattern 0101 on. The LEDs are active low
// for(i=0;i<200000;i++); // delay
_delay_ms(1000); // built-in delay function
// PORTR.OUTSET = 0x02; // Turns the debug led off. The led is active low
// PORTC_OUT = 0x02; // turn off External LED on PC0, on LED on PC1
Ln 1 : 53 Col 1 Sel 0
1.79 KB
ANSI
CR+LF INS C/C++ Source Code
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
32

Writing C Programs

```
41    PORTQ.OUT = 0x05;        // Turns the pattern 1010 on.  The LEDs are active low
42 //  for(i=0;i<200000;i++);   // delay
43    _delay_ms(1000);         // built-in delay function
44    }
45
46 //  PORTR.OUTCLR = 0x02;     // Turns the debug led on.  The led is active low
47 //  PORTC_OUT = 0x03;        // turn on External LEDs on PC0, and LED on PC1
48    PORTQ.OUTCLR = 0x0F;     // Turns the pattern 1111 on.  The LEDs are active low
49
50    while(1) {               // main Loop
51    // ToDo     Write code for the main loop
52    }
53 }
```

Ln 1 : 53 Col 1 Sel 0 | 1.79 KB | ANSI | CR+LF | INS | C/C++ Source Code

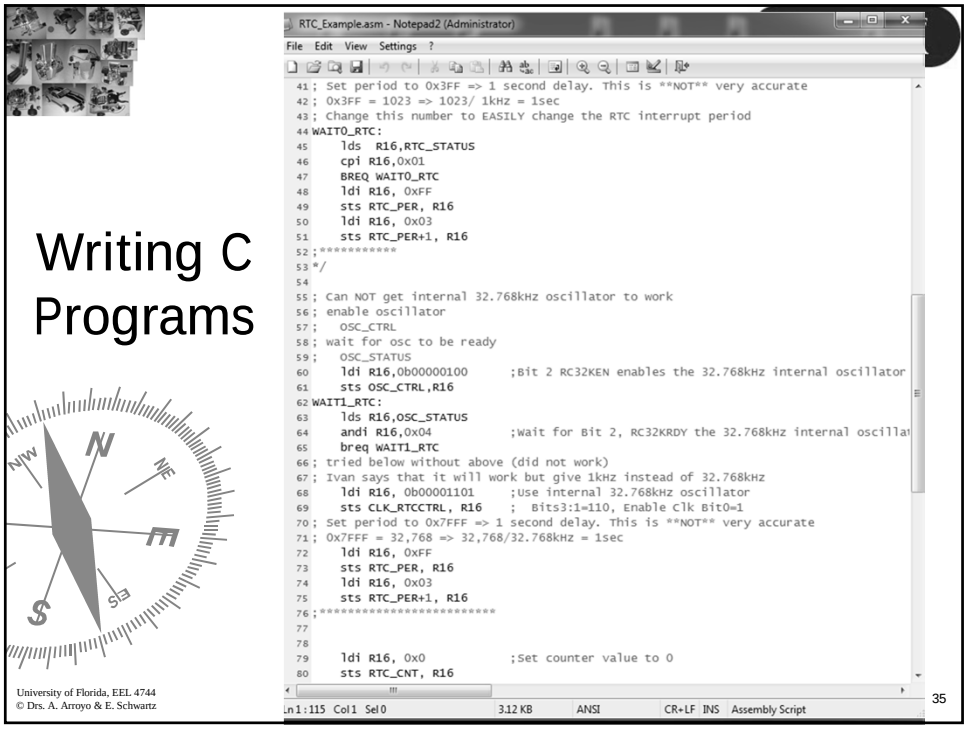
Writing C Programs

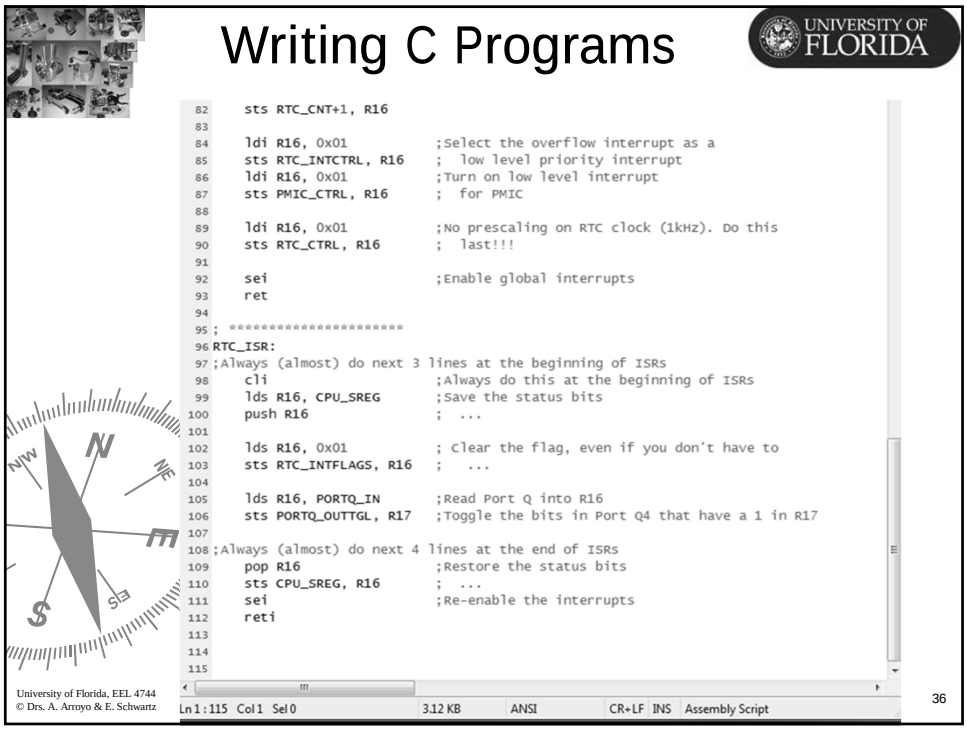
RTC_Example.asm - Notepad2 (Administrator)

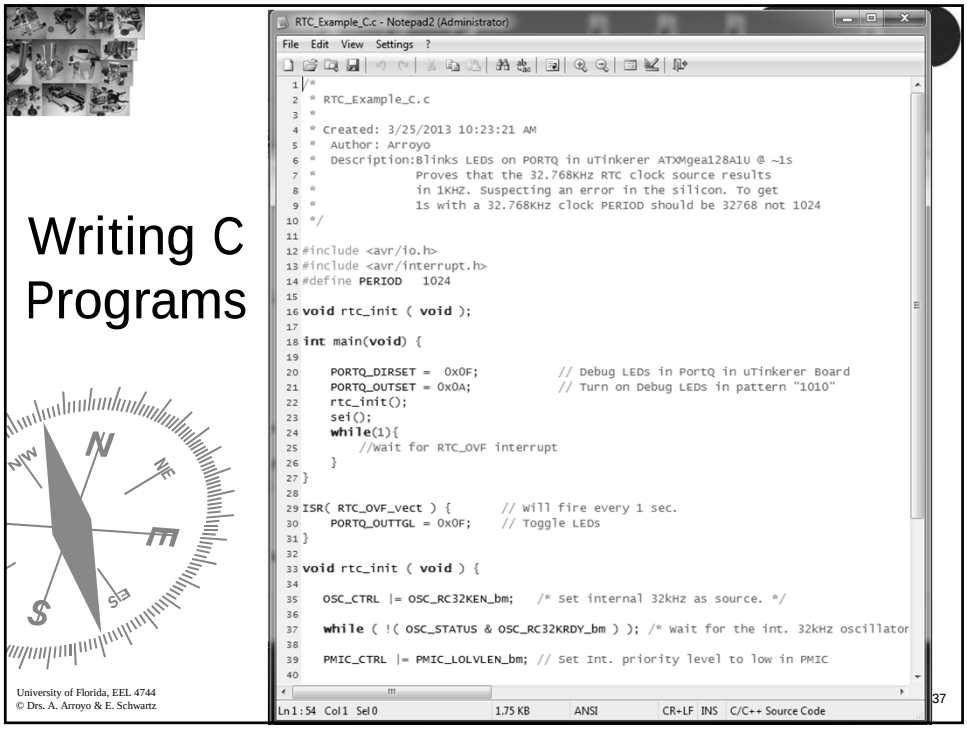
File Edit View Settings ?

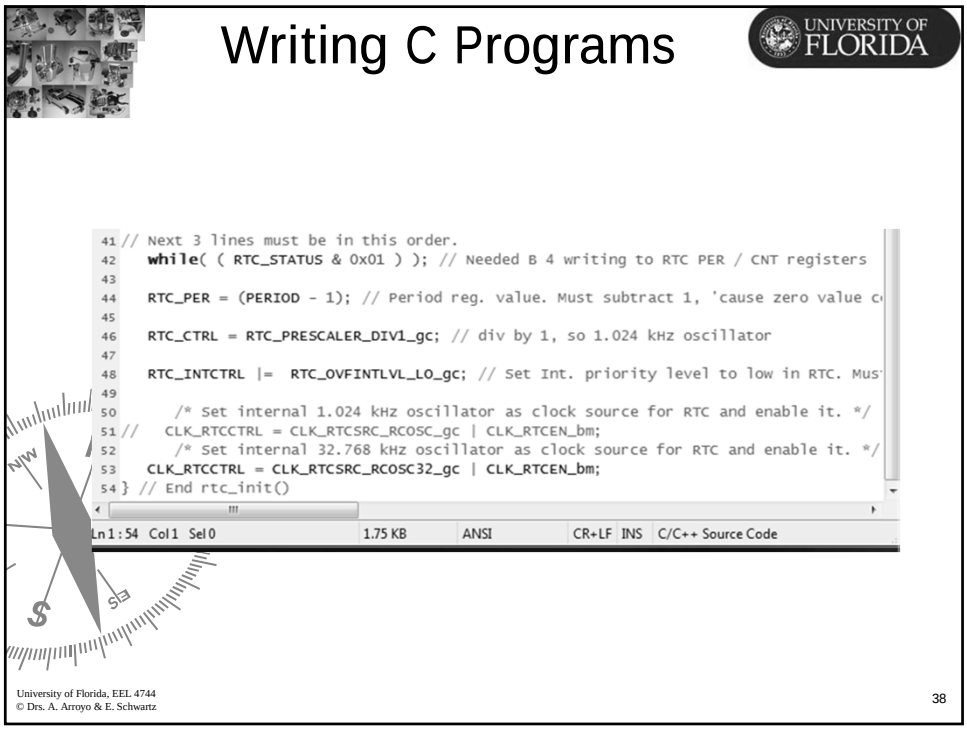
```
1  /*
2   * RTC_Example.asm
3   *
4   *  Created: 3/22/2013 9:27:16 AM
5   *   Author: Arroyo (from Schwartz)
6   *
7   * Use RTC to blink lights on uTinkerer every 1 second.
8   */
9
10 .include "ATxmega128A1Udef.inc" ;ATxmega128A1U definitions file
11
12 .org 0x0000         ;At reset, PC points to 0x0000
13    rjmp MAIN         ;Jump to start of program
14
15 .org RTC_OVF_vect   ;place code at the interrupt vector for the RTC_OVF_vect interru
16    rjmp RTC_ISR     ;relative jump to our interrupt routine
17
18 .org 0x200
19 MAIN:
20
21    ldi R16, 0x0F       ;4-bit Port Q, set the direction register for output
22    sts PORTQ_DIR, R16
23
24    ldi R16, 0x0A
25    ldi R17, 0x0F          ;Set Port Q bits to toggle (R17 used elsewhere)
26    sts PORTQ_OUTSET, R16  ;Set starting values on outputs
27
28    rcall   INIT_RTC
29
30 HERE:                    ;Infinite loop
31    rjmp HERE
32
33 ; **********************
34 INIT_RTC:
35
36 ; Below 6 instructions work together to get RTC interrupt every 1s
37
38 /*
39    ldi R16, 0b00000001     ;Use defalut ULP (1kHz=32 kHz/32) oscillator,
40    sts CLK_RTCCTRL, R16    ;  Bits3:1=000, Enable Clk Bit0=1
```

Ln 1 : 115 Col 1 Sel 0 | 3.12 KB | ANSI | CR+LF | INS | Assembly Script


Writing C Programs
RTC_Example.asm - Notepad2 (Administrator)
File Edit View Settings ?
41 ; Set period to 0x3FF => 1 second delay. This is **NOT** very accurate
42 ; 0x3FF = 1023 => 1023/ 1kHz = 1sec
43 ; Change this number to EASILY change the RTC interrupt period
44 WAIT0_RTC:
45 lds R16,RTC_STATUS
46 cpi R16,0x01
47 BREQ WAIT0_RTC
48 ldi R16, 0xFF
49 sts RTC_PER, R16
50 ldi R16, 0x03
51 sts RTC_PER+1, R16
52 ;***********
53 */
54
55 ; Can NOT get internal 32.768kHz oscillator to work
56 ; enable oscillator
57 ; OSC_CTRL
58 ; wait for osc to be ready
59 ; OSC_STATUS
60 ldi R16,0b00000100 ;Bit 2 RC32KEN enables the 32.768kHz internal oscillator
61 sts OSC_CTRL,R16
62 WAIT1_RTC:
63 lds R16,OSC_STATUS
64 andi R16,0x04 ;Wait for Bit 2, RC32KRDY the 32.768kHz internal oscilla
65 breq WAIT1_RTC
66 ; tried below without above (did not work)
67 ; Ivan says that it will work but give 1kHz instead of 32.768kHz
68 ldi R16, 0b00001101 ;Use internal 32.768kHz oscillator
69 sts CLK_RTCCTRL, R16 ; Bits3:1=110, Enable Clk Bit0=1
70 ; Set period to 0x7FFF => 1 second delay. This is **NOT** very accurate
71 ; 0x7FFF = 32,768 => 32,768/32.768kHz = 1sec
72 ldi R16, 0xFF
73 sts RTC_PER, R16
74 ldi R16, 0x03
75 sts RTC_PER+1, R16
76 ;*************************
77
78
79 ldi R16, 0x0 ;Set counter value to 0
80 sts RTC_CNT, R16
Ln 1 : 115 Col 1 Sel 0
3.12 KB
ANSI
CR+LF INS
Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
35

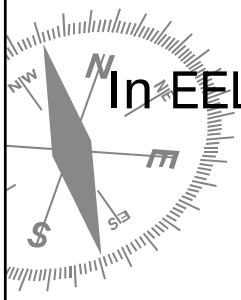

Writing C Programs
UNIVERSITY OF FLORIDA
82 sts RTC_CNT+1, R16
83
84 ldi R16, 0x01 ;Select the overflow interrupt as a
85 sts RTC_INTCTRL, R16 ; low level priority interrupt
86 ldi R16, 0x01 ;Turn on low level interrupt
87 sts PMIC_CTRL, R16 ; for PMIC
88
89 ldi R16, 0x01 ;No prescaling on RTC clock (1kHz). Do this
90 sts RTC_CTRL, R16 ; last!!!
91
92 sei ;Enable global interrupts
93 ret
94
95 ; **********************
96 RTC_ISR:
97 ;Always (almost) do next 3 lines at the beginning of ISRs
98 cli ;Always do this at the beginning of ISRs
99 lds R16, CPU_SREG ;Save the status bits
100 push R16 ; ...
101
102 lds R16, 0x01 ; Clear the flag, even if you don't have to
103 sts RTC_INTFLAGS, R16 ; ...
104
105 lds R16, PORTQ_IN ;Read Port Q into R16
106 sts PORTQ_OUTTGL, R17 ;Toggle the bits in Port Q4 that have a 1 in R17
107
108 ;Always (almost) do next 4 lines at the end of ISRs
109 pop R16 ;Restore the status bits
110 sts CPU_SREG, R16 ; ...
111 sei ;Re-enable the interrupts
112 reti
113
114
115
Ln 1 : 115 Col 1 Sel 0
3.12 KB
ANSI
CR+LF INS
Assembly Script
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
36


Writing C Programs
RTC_Example_C.c - Notepad2 (Administrator)
File Edit View Settings ?
/*
* RTC_Example_C.c
*
* Created: 3/25/2013 10:23:21 AM
* Author: Arroyo
* Description:Blinks LEDs on PORTQ in uTinkerer ATXMgea128A1U @ ~1s
* Proves that the 32.768KHz RTC clock source results
* in 1KHZ. Suspecting an error in the silicon. To get
* 1s with a 32.768KHz clock PERIOD should be 32768 not 1024
*/
#include <avr/io.h>
#include <avr/interrupt.h>
#define PERIOD 1024
void rtc_init (void);
int main(void) {
PORTQ_DIRSET = 0x0F; // Debug LEDs in PortQ in uTinkerer Board
PORTQ_OUTSET = 0x0A; // Turn on Debug LEDs in pattern "1010"
rtc_init();
sei();
while(1){
//wait for RTC_OVF interrupt
}
}
ISR(RTC_OVF_vect) { // Will fire every 1 sec.
PORTQ_OUTTGL = 0x0F; // Toggle LEDs
}
void rtc_init (void) {
OSC_CTRL |= OSC_RC32KEN_bm; /* Set internal 32kHz as source. */
while (!(OSC_STATUS & OSC_RC32KRDY_bm)); /* Wait for the int. 32kHz oscillator
PMIC_CTRL |= PMIC_LOLVLEN_bm; // Set Int. priority level to low in PMIC
Ln 1 : 54 Col 1 Sel 0 1.75 KB ANSI CR+LF INS C/C++ Source Code
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
37


Writing C Programs
UNIVERSITY OF FLORIDA
// Next 3 lines must be in this order.
while((RTC_STATUS & 0x01)); // Needed B 4 writing to RTC PER / CNT registers
RTC_PER = (PERIOD - 1); // Period reg. value. Must subtract 1, 'cause zero value c
RTC_CTRL = RTC_PRESCALER_DIV1_gc; // div by 1, so 1.024 kHz oscillator
RTC_INTCTRL |= RTC_OVFINTLVL_LO_gc; // Set Int. priority level to low in RTC. Mus
/* Set internal 1.024 kHz oscillator as clock source for RTC and enable it. */
// CLK_RTCCTRL = CLK_RTCSRC_RCOSC_gc | CLK_RTCEN_bm;
/* Set internal 32.768 kHz oscillator as clock source for RTC and enable it. */
CLK_RTCCTRL = CLK_RTCSRC_RCOSC32_gc | CLK_RTCEN_bm;
} // End rtc_init()
Ln 1 : 54 Col 1 Sel 0 1.75 KB ANSI CR+LF INS C/C++ Source Code
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
38

UNIVERSITY OF FLORIDA

See
AVR XMega Getting Started, IO Ports & USART.pptx
In EEL-4744 Spring 2013 Folder
Slides 14-27



UNIVERSITY OF FLORIDA

USART

21.16 Register Summary

21.16.1 Register Description - USART

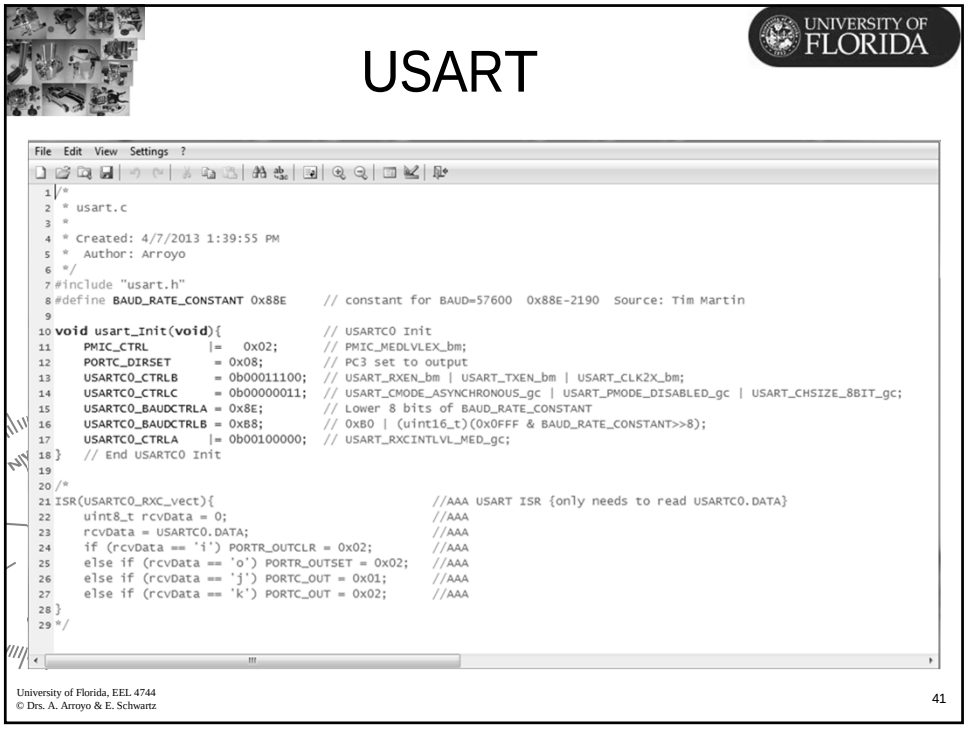
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	DATA	DATA[7:0]								249
+0x01	STATUS	RXCIF	TXCIF	DREIF	FERR	BUFOVF	PERR	-	RXB8	249
+0x02	Reserved	-	-	-	-	-	-	-	-	
+0x03	CTRLA	0	0	1 RXCINTLVL[1:0]		0	0 TXCINTLVL[1:0]	0	0DREINTLVL[1:0]	0 251
+0x04	CTRLB	0 -	0	0	RXEN	TXEN	CLK2X	MPCM	TXB8	251
+0x05	CTRLC	0 CMODE[1:0]		0 PMODE[1:0]		0 SBMODE	0	CHSIZE[2:0]		1 253
+0x06	BAUDCTRLA	1	0	0		0 BSEL[7:0]	1	1	0	255
+0x07	BAUDCTRLB	1	0 BSCALE[3:0]			1	1	0 BSEL[11:8]	0	0 254

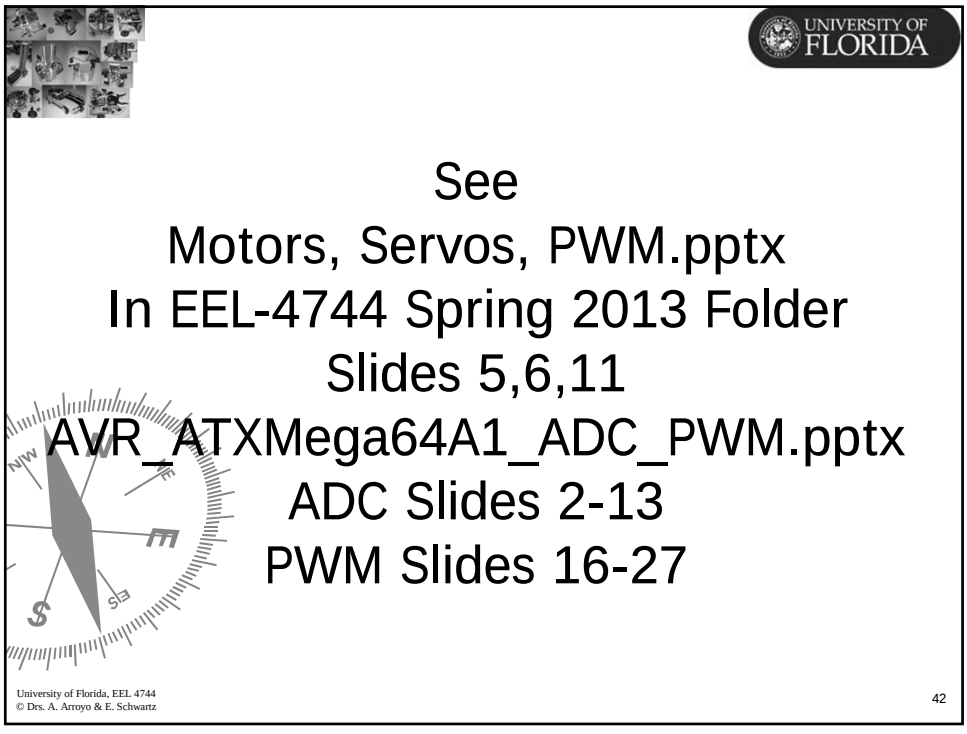
- Here is the register value summary for the USART.
- USARTC0.CTRLA = 0b00100000
- USARTC0.CTRLB = 0b00011100
- USARTC0.CTRLC = 0b00000011
- USARTC0.BAUDCTRLA = 0x8E
- USARTC0.BAUDCTRLB = 0xB8

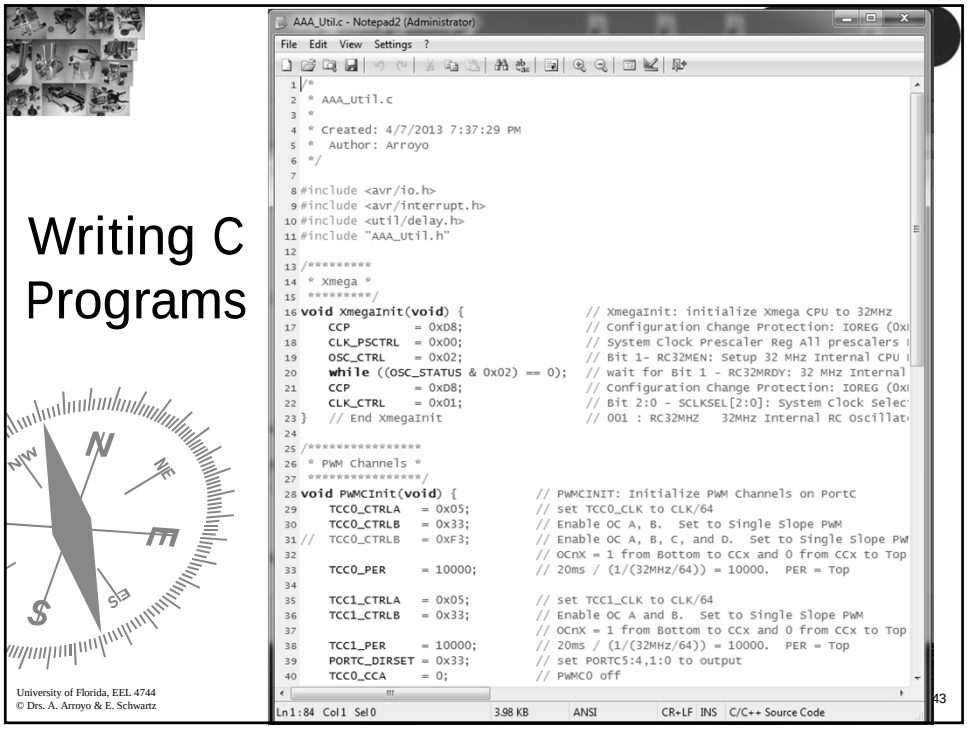
21.17 Interrupt Vector Summary

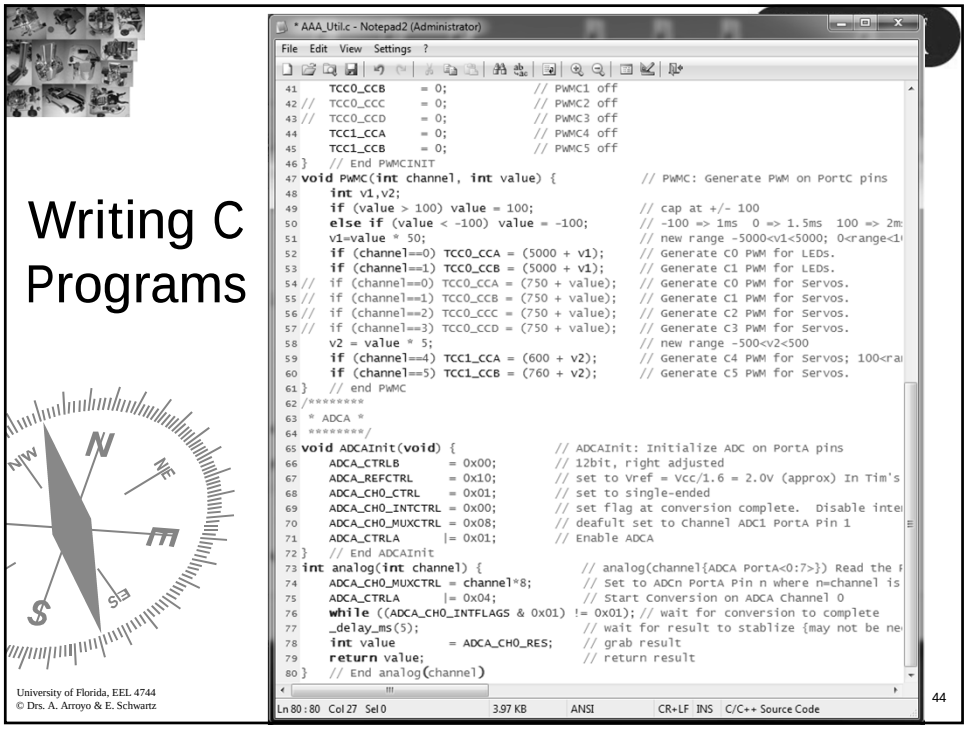
Table 21-10. USART interrupt vectors and their word offset address

Offset	Source	Interrupt Description
0x00	RXC_vect	USART Receive Complete Interrupt vector
0x02	DRE_vect	USART Data Register Empty Interrupt vector
0x04	TXC_vect	USART Transmit Complete Interrupt vector


UNIVERSITY OF FLORIDA
USART
File Edit View Settings ?
/*
* usart.c
*
* Created: 4/7/2013 1:39:55 PM
* Author: Arroyo
*/
#include "usart.h"
#define BAUD_RATE_CONSTANT 0x88E // constant for BAUD=57600 0x88E-2190 Source: Tim Martin
void usart_Init(void){ // USARTC0 Init
PMIC_CTRL |= 0x02; // PMIC_MEDLVLEX_bm;
PORTC_DIRSET = 0x08; // PC3 set to output
USARTC0_CTRLB = 0b00011100; // USART_RXEN_bm | USART_TXEN_bm | USART_CLK2X_bm;
USARTC0_CTRLC = 0b00000011; // USART_CMODE_ASYNCHRONOUS_gc | USART_PMODE_DISABLED_gc | USART_CHSIZE_8BIT_gc;
USARTC0_BAUDCTRLA = 0x8E; // Lower 8 bits of BAUD_RATE_CONSTANT
USARTC0_BAUDCTRLB = 0xB8; // 0xB0 | (uint16_t)(0x0FFF & BAUD_RATE_CONSTANT>>8);
USARTC0_CTRLA |= 0b00100000; // USART_RXCINTLVL_MED_gc;
} // End USARTC0 Init
/*
ISR(USARTC0_RXC_vect){ //AAA USART ISR {only needs to read USARTC0.DATA}
uint8_t rcvData = 0; //AAA
rcvData = USARTC0.DATA; //AAA
if (rcvData == 'i') PORTR_OUTCLR = 0x02; //AAA
else if (rcvData == 'o') PORTR_OUTSET = 0x02; //AAA
else if (rcvData == 'j') PORTC_OUT = 0x01; //AAA
else if (rcvData == 'k') PORTC_OUT = 0x02; //AAA
}
*/
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
41


UNIVERSITY OF FLORIDA
See
Motors, Servos, PWM.pptx
In EEL-4744 Spring 2013 Folder
Slides 5,6,11
AVR_ATXMega64A1_ADC_PWM.pptx
ADC Slides 2-13
PWM Slides 16-27
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
42


Writing C Programs
AAA_Util.c - Notepad2 (Administrator)
File Edit View Settings ?
/*
* AAA_Util.c
*
* Created: 4/7/2013 7:37:29 PM
* Author: Arroyo
*/
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "AAA_Util.h"
/*********
* Xmega *
*********/
void XmegaInit(void) {
CCP = 0xD8;
CLK_PSCTRL = 0x00;
OSC_CTRL = 0x02;
while ((OSC_STATUS & 0x02) == 0);
CCP = 0xD8;
CLK_CTRL = 0x01;
} // End XmegaInit
/****************
* PWM Channels *
****************/
void PWMCInit(void) {
TCC0_CTRLA = 0x05;
TCC0_CTRLB = 0x33;
// TCC0_CTRLB = 0xF3;
TCC0_PER = 10000;
TCC1_CTRLA = 0x05;
TCC1_CTRLB = 0x33;
TCC1_PER = 10000;
PORTC_DIRSET = 0x33;
TCC0_CCA = 0;
Ln 1 : 84 Col 1 Sel 0
3.98 KB
ANSI
CR+LF INS C/C++ Source Code
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
43


Writing C Programs
* AAA_Util.c - Notepad2 (Administrator)
File Edit View Settings ?
TCC0_CCB = 0;
// TCC0_CCC = 0;
// TCC0_CCD = 0;
TCC1_CCA = 0;
TCC1_CCB = 0;
} // End PWMCINIT
void PWMC(int channel, int value) {
int v1,v2;
if (value > 100) value = 100;
else if (value < -100) value = -100;
v1=value * 50;
if (channel==0) TCC0_CCA = (5000 + v1);
if (channel==1) TCC0_CCB = (5000 + v1);
// if (channel==0) TCC0_CCA = (750 + value);
// if (channel==1) TCC0_CCB = (750 + value);
// if (channel==2) TCC0_CCC = (750 + value);
// if (channel==3) TCC0_CCD = (750 + value);
v2 = value * 5;
if (channel==4) TCC1_CCA = (600 + v2);
if (channel==5) TCC1_CCB = (760 + v2);
} // end PWMC
/********
* ADCA *
********/
void ADCAInit(void) {
ADCA_CTRLB = 0x00;
ADCA_REFCTRL = 0x10;
ADCA_CH0_CTRL = 0x01;
ADCA_CH0_INTCTRL = 0x00;
ADCA_CH0_MUXCTRL = 0x08;
ADCA_CTRLA |= 0x01;
} // End ADCAInit
int analog(int channel) {
ADCA_CH0_MUXCTRL = channel*8;
ADCA_CTRLA |= 0x04;
while ((ADCA_CH0_INTFLAGS & 0x01) != 0x01);
_delay_ms(5);
int value = ADCA_CH0_RES;
return value;
} // End analog(channel)
Ln 80 : 80 Col 27 Sel 0
3.97 KB
ANSI
CR+LF INS C/C++ Source Code
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
44

Writing C Programs

AAA_Util.h - Notepad2 (Administrator)

File Edit View Settings ?

```
/*
 * AAA_Util.h
 *
 * Created: 2/2/2012 7:42:11 PM
 *  Author: Arroyo
 */


#ifndef PWM_H_
#define PWM_H_

#include <avr/io.h>
#include <avr/interrupt.h>

void XmegaInit(void);               // XmegaInit: initialize Xmega CPU to 32MHz

void PWMCInit(void);                // PWMCINIT: Initialize PWM Channels on PortC

void PWMC(int channel, int value);  // PWMC: Generate PWM on PortC pins

void ADCAInit(void);                // ADCAInit: Initialize ADC on PortA pins

int analog(int channel);            // analog(channel{ADCA Pins <0:7>}) Read the PortA

#endif /* PWM_H_ */
```


45

Writing C Programs

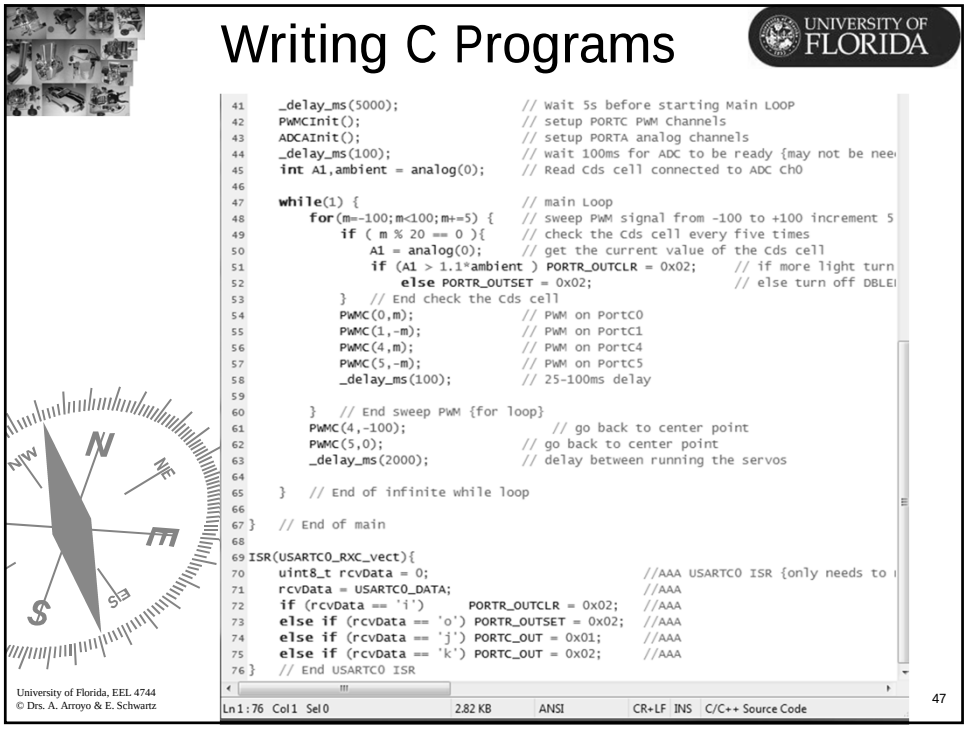
All_in_C.c - Notepad2 (Administrator)

File Edit View Settings ?

```
/*
 * All_in_C.c
 *
 * Created: 4/8/2013 12:17:01 PM
 *  Author: Arroyo
 */

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "usart.h"
#include "AAA_Util.h"

int main(void) {                 // Main program to toggle LEDs & Process USART interr

XmegaInit();                     // XmegaInit: initialize Xmega CPU to 32MHz
usart_Init();                    // USARTC0 Init

PORTR_DIRSET =  0x02;            // Debug Led PortR1 in Tim's Board
PORTC_DIRSET =  0x03;            // External LEDs on PC0 & PC1

sei();                           // Turn on Xmega interrupts

int16_t  m=0;                    // m:loop counter 16-bit integer

for (m=0;m<5;m++) {              // number of times to blink LEDs

PORTR_OUTCLR = 0x02;         // Turns the debug led on.  The led is active low
PORTC_OUT = 0x01;            // turn on External LED on PC0, off LED on PC1
_delay_ms(500);              // built-in delay function

PORTR_OUTSET = 0x02;         // Turns the debug led off.  The led is active low
PORTC_OUT = 0x02;            // turn off External LED on PC0, on LED on PC1
_delay_ms(500);              // wait 500ms using AVR built-in delay function

    }   // End number of times to blink LEDs

    PORTR.OUTCLR = 0x02;     // Turns the debug led on.  The led is active low
    PORTC_OUT = 0x03;        // turn on External LEDs on PC0, and LED on PC1
```

Ln 1 : 76 Col 1 Sel 0 | 2.82 KB | ANSI | CR+LF INS | C/C++ Source Code

46


Writing C Programs
UNIVERSITY OF FLORIDA
41 _delay_ms(5000); // Wait 5s before starting Main LOOP
42 PWMCInit(); // setup PORTC PWM Channels
43 ADCAInit(); // setup PORTA analog channels
44 _delay_ms(100); // wait 100ms for ADC to be ready {may not be nee
45 int A1,ambient = analog(0); // Read Cds cell connected to ADC Ch0
46
47 while(1) { // main Loop
48 for(m=-100;m<100;m+=5) { // sweep PWM signal from -100 to +100 increment 5
49 if (m % 20 == 0){ // check the Cds cell every five times
50 A1 = analog(0); // get the current value of the Cds cell
51 if (A1 > 1.1*ambient) PORTR_OUTCLR = 0x02; // if more light turn
52 else PORTR_OUTSET = 0x02; // else turn off DBLE
53 } // End check the Cds cell
54 PWMC(0,m); // PWM on PortC0
55 PWMC(1,-m); // PWM on PortC1
56 PWMC(4,m); // PWM on PortC4
57 PWMC(5,-m); // PWM on PortC5
58 _delay_ms(100); // 25-100ms delay
59
60 } // End sweep PWM {for loop}
61 PWMC(4,-100); // go back to center point
62 PWMC(5,0); // go back to center point
63 _delay_ms(2000); // delay between running the servos
64
65 } // End of infinite while loop
66
67 } // End of main
68
69 ISR(USARTC0_RXC_vect){
70 uint8_t rcvData = 0; //AAA USARTC0 ISR {only needs to
71 rcvData = USARTC0_DATA; //AAA
72 if (rcvData == 'i') PORTR_OUTCLR = 0x02; //AAA
73 else if (rcvData == 'o') PORTR_OUTSET = 0x02; //AAA
74 else if (rcvData == 'j') PORTC_OUT = 0x01; //AAA
75 else if (rcvData == 'k') PORTC_OUT = 0x02; //AAA
76 } // End USARTC0 ISR
Ln 1 : 76 Col 1 Sel 0
2.82 KB
ANSI
CR+LF INS
C/C++ Source Code
N
NW
NE
W
E
S
ES
University of Florida, EEL 4744
© Drs. A. Arroyo & E. Schwartz
47