

Atmel AVR Basics


Atmel Studio 7 & Xmega

Arduino Demo

A. A. Arroyo


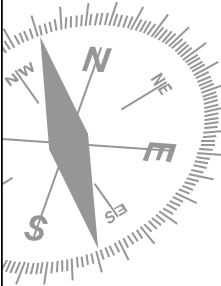
University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

1




Overview

- ▶ Basic AVR Knowledge
- ▶ Atmel Studio
- ▶ Programming



University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

2




AVR Basics

- ▶ The AVR microcontrollers are divided into three groups:
 - 1. tiny AVR
 - 2. AVR (Classic AVR)
 - 3. mega AVR
 - 4. xmega AVR
- ▶ The difference between these devices lies in the available features. The *tinyAVR* μ C are usually devices with lower pin-count or a reduced feature set compared to the *mega & xmega AVR's*. All AVR devices have identical instructions and memory organization.

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

3




AVR Basics

- ▶ AVR's contain SRAM, EEPROM, External SRAM interface, ADC, Hardware Multiplier, UART, USART, etc. A *tinyAVR* and a *megaAVR* stripped off all the peripheral modules, leaves us with the AVR Core — the same for all AVR's.
- ▶ **Datasheets are complete technical documents — a reference on how a given peripheral/feature works.**
 - 1. One Page—Key information and Feature List
 - 2. Architectural Overview
 - 3. Peripheral Descriptions
 - 4. Memory Programming
 - 5. Characteristics
 - 6. Register Summary
 - 7. Instruction Set Summary
 - 8. Packaging Information

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

4




Common Peripherals

- ▶ **UART**
 - Provides serial communication interface
 - Typically RS232
- ▶ **Analog/Digital Channels**
 - Many sensor are analog
 - 4 channels minimum
- ▶ **SPI/I2C/CAN interfaces**
 - Synchronous serial, very common interfaces
- ▶ **JTAG**
 - Allows in circuit debugging and programming
- ▶ **ISP**
 - In system programming, serial communication
- ▶ **Timers**
 - The more the better
- ▶ **PWM / frequency generation modules**
 - Allows hardware generation of motor driving signals

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

5




ATMega & ATxMega


- ▶ Most popular here at UF & in Arduinos
- ▶ Kitchen sink of peripherals
 - UARTS, multi-channel A/D, multiple counters and PWM channels, SPI/I2C/CAN, ISP, JTAG
- ▶ RISC based 8Bit core
- ▶ 16/32MIPS @ 16/32Mhz
- ▶ Development Software free from Atmel
- ▶ Large Hobbyist user base on the internet

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

6



To use all the xMega's capabilities, get the Datasheets!




- ◆ Go to www.atmel.com and search for xMega128A1


Datasheet

PDF	Software	Description
		Atmel AVR XMEGA A Manual Preliminary Manual <i>(file size: 8.53MB, 432 pages, revision I, updated: 11/2012)</i>
		ATxmega64A1/128A1 Preliminary <i>(file size: 2.6MB, 122 pages, revision O, updated: 06/2013)</i>

- ◆ Download Atmel-8067...pdf and doc8077.pdf



University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Block Diagram


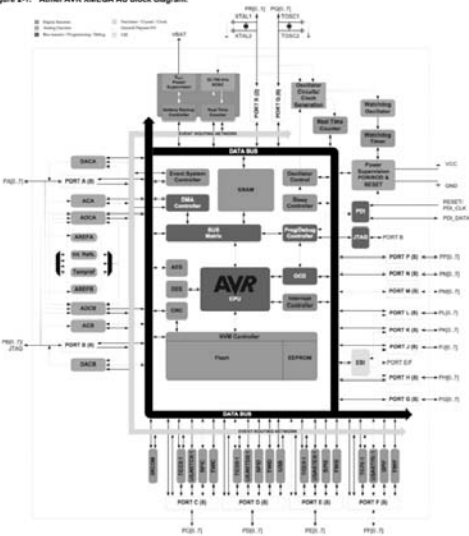





Figure 2-1. Atmel AVR XMEGA AU block diagram.



- ◆ xMega AU
- ◆ Shows available peripheral devices and the ports they use
- ◆ In IMDL, we extensively use A/Ds, Timers, and USARTs.



University of Florida, EEL 5666
© Dr. A. Antonio Arroyo






Atmel Studio 7

- ▶ **Faster and more powerful than Studio 6**
- ▶ **A free Integrated Development Environment (IDE) for AVR Software**
- ▶ **Allows chip simulation and in-circuit emulation**
- ▶ **Supports the whole AVR family of Microcontrollers (MCUs)**
- ▶ **Has an easy to use User Interface (UI) and gives complete overview**
- ▶ **Uses the same UI for simulation & emulation**

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

9





Atmel Studio 7

- ▶ **Uses for Atmel Studio 5-7**
 - Programming
 - Simulating
 - Debugging
- ▶ **Improvements over AVR Studio 4**
 - Integrated Drivers and services
 - The entire user interface
 - Updated/improved simulator/debugger

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

10




Atmel Studio 7

- ▶ Built upon Visual Studio
 - Uses 2015 Visual Studio
 - Code Completion
 - Integrated Datasheet
 - ▶ Need more info on your MCU, just go to properties to download your MCU's datasheet.
 - I/O View
 - ▶ See the register layout of the processor
 - ▶ Use it in the Simulator or real-time with the debugger
 - Support for ARM Processors

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

11




Atmel Studio 7

- ▶ Watch
 - Use this window to monitor variables when you debug or simulate.
- ▶ ASF(AVR Studio Framework)
 - Need help with getting some feature working?
 - ▶ It has drivers & services downloadable into your code.
 - ▶ Ex. I2C just download the TWI driver/service
- ▶ Useful Links in the Atmel Studio 6 Website
 - ▶ ASF for Atxmega A Family
 - ▶ User's Guide & Getting Started

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

12



Atmel Studio 7

► Step 1: Download and Install Atmel Studio:

<http://www.atmel.com/System/BaseForm.aspx?target=tcm:26-77368>
 This installer contains Atmel Studio 7.0 with Atmel Software Framework 3.28.1 and Atmel Toolchains. It is recommended to use this installer if you have internet access while installing, since it enables incremental updates in the future.


<http://www.atmel.com/System/BaseForm.aspx?target=tcm:26-77367>
 This installer contains Atmel Studio 7.0 with Atmel Software Framework 3.28.1 and Atmel Toolchains. Use this installer if you do not have internet access while installing. It is highly recommended to use the smaller web installer if you can since it provides the ability to get incremental updates in the future.

http://www.atmel.com/Images/Atmel-YYYYYB-atmelstudio_Release-Note.pdf

► Atmel Studio 7.0
 (file size: 1044 KB, 82 pages, updated: 11/2015) for Atmel Studio 7.0.634

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

13



Atmel Studio 7

► Step 0: Apply any patches {Currently none}


► Step 1: Create a New Project

► Step 2: Configure Project Settings
 What kind of project we want to create, and setting up filenames and locations.


► Step 3: Write your code

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

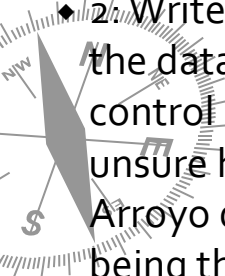
14




Using Peripherals




- ◆ There are two ways engineers write code to control a peripheral module on the chip:
 - ◆ 1: Copy someone else's code. This is fine in this class but when it doesn't work, you're stuck.
 - ◆ 2: Write it from scratch by reading every word of the datasheet. This method gives you a lot more control but can be frustrating when you're unsure how it should work. {This is the way Dr. Arroyo develops code. This is another example of being the tortoise and not the hare}



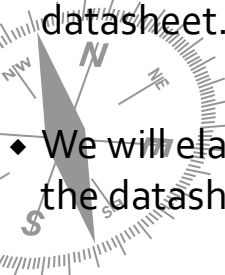
University of Florida, EEL 5666
© Dr. A. Antonio Arroyo




Using Peripherals




- ◆ Ideally in IMDL your code should be a combination of both methods. If you can find some code to give you a starting point, it can save a lot of time. Otherwise, don't be afraid to dig through the datasheet.
- ◆ We will elaborate on how to get information from the datasheet



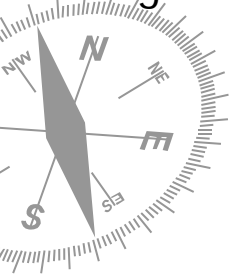
University of Florida, EEL 5666
© Dr. A. Antonio Arroyo




Using Peripherals




- ◆ The three parts you are interested in
 - ◆ Overview
 - ◆ Description/Features
 - ◆ Register Map!



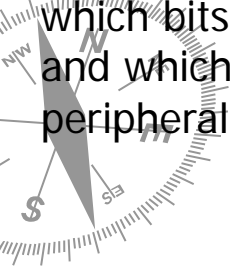
University of Florida, EEL 5666
© Dr. A. Antonio Arroyo




Using Peripherals



- ◆ Make sure you understand how the peripheral is supposed to work, what pins it uses, what resources it uses (Timing channels, Ports, etc)
- ◆ The register map is where you actually see which bits to set to configure the peripheral, and which registers to use to interact with the peripheral




University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Register Names

- ◆ 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
- ◆ Register Name from doc8077
- ◆ 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 5666
© Dr. A. Antonio Arroyo




Register Names

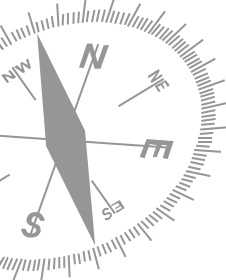
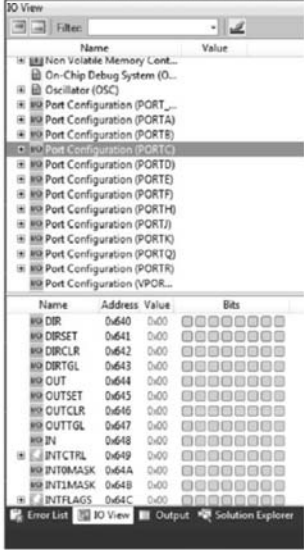
- ◆ Syntax/Hierarchy
 - ◆ When programming for any Atmel processor a required dependency is in <io.h>. This header file points to the device specific header containing definitions of all the registers, i.e., assignments to their respective memory map locations. To access the device specific IO header file in Atmel Studio first locate the "Solution Explorer" window. Now expand the "Dependencies" folder to find a file like "iox64a1.h". Note the exact name varies based upon the chip in your board. The '64' is the memory size, and could be '128' in some cases. The 'a1' or 'a1u' specifies the series.

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

Register Names




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

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

Register Names




- ◆ Here is the register summary for the USART. We can see that serial communication uses 7 regs
 - ◆ 1 for Data
 - ◆ 1 for status
 - ◆ 3 for control
 - ◆ 2 for baud rate


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			RXCINTLV[1:0]		TXCINTLV[1:0]		DREINTLV[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



Find Examples



- ◆ Type "xmega usart" into Google



About 6,180 results (0.25 seconds)

Everything

Images

Videos

[PDF] AVR1307: Using the XMEGA USART ☆ - 5:21pm


File Format: PDF/Adobe Acrobat

The XMEGA USART is by default set in UART (asynchronous) mode. Transmit and receive are enabled individually, and the transfers can be implemented using ...


www.atmel.com/dyn/resources/prod_documents/doc8049.pdf - Similar

- ◆ 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 5666
© Dr. A. Antonio Arroyo




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.

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Sample Code



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

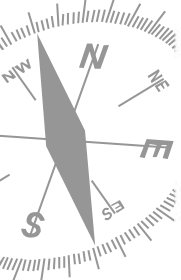
```
PORTD.OUTSET = 1<<5; //sets bit 5. Requires just a store instruction
```

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.
```



University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Sample Code



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 PINCTRL registers these features can be harnessed. For details on all these features consult the device datasheets.

Example6:


```
PORTF.PINCTRL = 0x10; //enables a pull-down resistor on pin 5 of port C
PORTF.PINCTRL = 0x18; //enables a pull-up resistor on pin 6 of port C
```



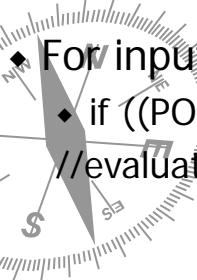
University of Florida, EEL 5666
© Dr. A. Antonio Arroyo




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



University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Assembly Code Sample




```

1  /*
2  * Test_DAQ_ASM.asm
3  * Created: 01/10/2015 7:07:13 PM
4  * Author: Arroyo
5  */
6  .....
7  *
8  * Title:      Test_DAQ_Asm.asm
9  * Programmer: A. Antonio Arroyo
10 * Date:       January 10, 2015
11 * Version:    1.0
12 *
13 * Description:
14 * Test the Atmel ATxMEGA32A4U in Tim's DAQ Board using the Assembler
15 * Blink Internal Debug LED on PC0 and External LEDs in PD0 & PD1
16 * ...../
17
18 ;.include "m128def.inc"           ;The regular Atmega128 definitions file
19 ;.include "ATxmega128A1Udef.inc" ;The DIY AtxMegal28A1U definitions file
20 ;.include "ATxmega32A4Udef.inc"  ;The DAQ AtxMega32A4U definitions file
21 .org 0x0000                      ;Places the following code from address 0x0000
22 rjmp RESET                      ;Take a Relative Jump to the RESET Label
23
24 RESET:                          ;Reset Label (start of Main)
25 ldi R16, 0x08                   ;Store 12 in R16 (Repeat Count)
26
27 ldi R17, 0x5                    ;Store 5 in R17 (i, Outer Loop)
28 ldi R18, 0xFF                   ;Store 255 in R18 (j, Middle Loop)
29 ldi R19, 0xFF                   ;Store 255 in R19 (k, Inner Loop)
30 LDI EH, HIGH(PORTD_OUT)


```

Assembly language source file length: 3308 lines: 90 Ln:1 Col:1 Sel:0|0 Dot/Windows UTF-8 INS




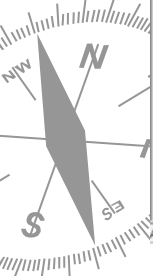
University of Florida, EEL 5666
© Dr. A. Antonio Arroyo

28



Assembly Code Sample






```


31  LDI ZL, LOW(PORTD_OUT) ; Z --> PORTD_OUT
32  LDI YH, HIGH(PORTC_OUT)
33  LDI YL, LOW(PORTC_OUT) ; Y --> PORTC_OUT
34  ldi R20, 0x03 ;Set PD0 & PD1 direction as Output
35  STS PORTD_DIR,R20
36  ldi R20, 0x01 ;Set PC0 direction as Output
37  STS PORTC_DIR,R20
38
39 ; Set PD0 pin High & PD1 Low
40  ldi R20, 0x01 ;Set PD0 High & PD1 Low
41  ST Z, R20
42  ldi R20, 0x00 ;Set PC0 High
43  ST Y, R20 ;Turn on Debug LED (Active Low)
44  Loop: ;Delay Loop 1
45  dec R19 ;Decrement R19
46  brne Loop ;If not zero jump to the Loop label
47  ldi R19, 0xFF ;Restore 255 in R19
48  dec R18 ;Decrement R18
49  brne Loop ;If not zero jump to the Loop label
50  ldi R18, 0xFF ;Restore 255 in R18
51  ldi R19, 0xFF ;Restore 255 in R19
52  dec R17 ;Decrement R17
53  brne Loop ;If not zero jump to the Loop label
54
55 ;Set Set PD0 pin Low & PD1 High
56  ldi R17, 0x05 ;Store 05 in R17
57  ldi R18, 0xFF ;Store 0xFF in R18
58  ldi R19, 0xFF ;Store 0xFF in R19
59  ldi R20, 0x02 ;Set PD0 Low & PD1 High (R20)
60  ST Z, R20
                
```

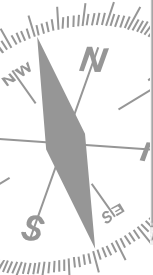
Assembly language source file
length: 3308 lines: 90
Ln: 9 Col: 74 Set: 0|0
Dot/Windows UTF-8
INS

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo
29



Assembly Code Sample






```


61  ldi R20, 0x01 ;Turn off Debug LED (R20)
62  ST Y, R20 ;Turn off Debug LED (Active Low)
63
64  Loop2: ;Delay Loop 2
65  dec R19 ;Decrement R19
66  brne Loop2 ;If not zero jump to the Loop label
67  ldi R19, 0xFF ;Restore 255 in R19
68  dec R18 ;Decrement R18
69  brne Loop2 ;If not zero jump to the Loop label
70  ldi R18, 0xFF ;Restore 255 in R18
71  ldi R19, 0xFF ;Restore 255 in R19
72  dec R17 ;Decrement R17
73  brne Loop2 ;If not zero jump to the Loop label
74
75 ;Repeat Count times
76  ldi R17, 0x05 ;Restore 05 to R17
77  ldi R18, 0xFF ;Restore 255 to R18
78  ldi R19, 0xFF ;Restore 255 to R19
79  ldi R20, 0x01 ;Set PD0 High & PD1 Low
80  ST Z, R20
81  ldi R20, 0x00 ;Set PC0 High
82  ST Y, R20 ;Set PC0 High
83  dec R16 ;Decrement COUNT
84  brne Loop ;If not zero jump to the Loop label
85
86  ldi R20, 0x03 ;Set PD0 High & PD1 High to end
87  ST Z, R20
88  ldi R20, 0x00 ;Turn on Debug LED (Active Low)
89  ST Y, R20
90  Here: rjmp Here
                
```

Assembly language source file
length: 3308 lines: 90
Ln: 9 Col: 74 Set: 0|0
Dot/Windows UTF-8
INS

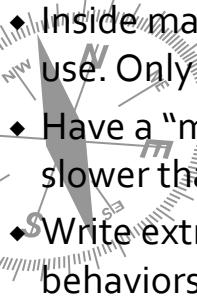
University of Florida, EEL 5666
© Dr. A. Antonio Arroyo
30




Writing C Programs




- ◆ Robots in IMDL 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.
- ◆ 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 as fast as possible (never slower than 1Hz, ~10-20Hz is ideal)
- ◆ Write extra functions outside of main() for specific behaviors



University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Writing C Programs




```


#include
main(){
    Initialize LCD, Servos, PWM, A/D, Serial,
    DIR, variables
while(1){
    read sensors
    interpret sensors
    function1()
    update LCD } // while
} // end of main

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


```

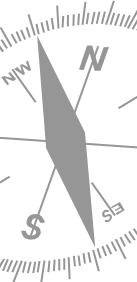


University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Writing C Programs






```

16  L  *****/
17  #include <avr/io.h>
18
19  int main(void) {
20
21      PORTC.DIRSET = 0b00000001;    //Debug Led
22      PORTD.DIRSET = 0b00000011;    //External LEDs on PD0 & PD1
23
24      uint32_t i=0;
25      int16_t j=0;
26
27      for (j=0;j<5;j++) {    // number of times to blink LEDs
28
29          PORTC.OUTCLR = 0x01;    // Turns the debug led on. The led is active low
30          PORTD_OUT = 0x01;    // turn on External LED on PD0, off LED on PD1
31          for(i=0;i<200000;i++); // delay
32          // _delay_ms(1000);    // built-in delay function
33
34          PORTC.OUTSET = 0x01;    // Turns the debug led off. The led is active low
35          PORTD_OUT = 0x02;    // turn off External LED on PD0, on LED on PD1
36          for(i=0;i<200000;i++); // delay
37          // _delay_ms(1000);    // built-in delay function
38      }
39      PORTC.OUTCLR = 0x01;    // Turns the debug led on. The led is active low
40      PORTD_OUT = 0x03;    // turn on External LEDs on PD0, and LED on PD1
41
42      while(1) {    // main Loop
43          // TODO    Write code for the main loop
44      }
45  }


```

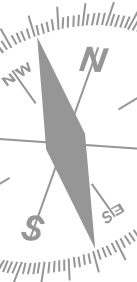
C source file length: 1534 lines: 45 Ln: 18 Col: 1 Sel: 0|0 Dos/Windows UTF-8 INS

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Writing C Programs






```

16  * Include header files for all drivers that have been imported from
17  * AVR Software Framework (ASF).
18  */
19  #include <asf.h>
20  #include <avr/io.h>
21  #include <util/delay.h>
22  #include "clock.h"
23  int main(void) {
24      // board init();
25      clockInit();    //clock initialization for DAQ
26      PORTC.DIRSET = 0x01;    //Debug Led
27      PORTD.DIRSET = 0x03;    //External LEDs on PD0 & PD1
28      int16_t j=0;
29      for (j=0;j<6;j++) {    // number of times to blink LEDs
30
31          PORTC.OUTCLR = 0x01;    // Turns the debug led on. The led is active low
32          PORTD_OUT = 0x01;    // turn on External LED on PD0, off LED on PD1
33          _delay_ms(1000);    // delay
34
35          PORTC.OUTSET = 0x01;    // Turns the debug led off. The led is active low
36          PORTD_OUT = 0x02;    // turn off External LED on PD0, on LED on PD1
37          _delay_ms(1000);    // delay
38      }
39
40      PORTC.OUTCLR = 0x01;    // Turns the debug led on. The led is active low
41      PORTD_OUT = 0x03;    // turn on External LEDs on PD0, and LED on PD1
42      while(1) {    // main Loop
43          // TODO Code that runs forever in the robot
44      }
45  }


```


C source file length: 1630 lines: 45 Ln: 1 Col: 1 Sel: 0|0 Dos/Windows UTF-8 INS

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Writing C Programs






```


50 #include <asf.h>
51 #include <avr/io.h>
52 #include <util/delay.h>
53 #include "clock.h"
54
55 int main(void) {
56     // board_init();
57
58     clockInit(); //clock initialization for DAQ
59     PORTC.DIRSET = 0x01; //Debug Led
60     PORTD.DIRSET = 0x03; //External LEDs on PD0 & PD1
61     int16_t j=0;
62
63     for (j=0;j<7;j++) { // number of times to blink LEDs
64
65         PORTC.OUTCLR = 0x01; // Turns the debug led on. The led is active low
66         PORTD_OUT = 0x01; // turn on External LED on PD0, off LED on PD1
67         delay_ms(500); // delay
68
69         PORTC.OUTSET = 0x01; // Turns the debug led off. The led is active low
70         PORTD_OUT = 0x02; // turn off External LED on PD0, on LED on PD1
71         delay_ms(500); // delay
72     }
73
74     PORTC.OUTCLR = 0x01; // Turns the debug led on. The led is active low
75     PORTD_OUT = 0x03; // turn on External LEDs on PD0, and LED on PD1
76     while(1) { // main Loop
77         // TODO Code that runs forever in the robot
78     }
79 }
                    
```

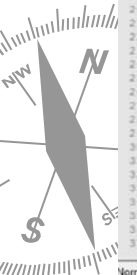
C source file length: 2217 lines: 79 Ln: 1 Col: 1 Sel: 0|0 Dos/Windows UTF-8 INS

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo



Arduino Programs





```

1 #include <LiquidCrystal.h>
2 /*
3  EEL-5666 Example 1: Use of Digital Output Pins. Turn on/off external
4  LEDs connected to Pins 11 & 12 alternatively & the internal LED
5  on Pin 13 on/off. Add a delay in each state of 1 sec.
6  Arroyo August 26 2015
7  */
8
9 int extLED1 = 11; // External LED connected to digital pin 11
10 int extLED2 = 12; // External LED connected to digital pin 12
11 int intLED = 13; // Internal LED connected to digital pin 13
12
13 void setup() {
14     pinMode(extLED1, OUTPUT); // sets digital pin 12 as output
15     pinMode(extLED2, OUTPUT); // sets digital pin 12 as output
16     pinMode(intLED, OUTPUT); // sets digital pin 13 as output
17 }
18
19 int count=0;
20 void loop() {
21     if (count < 1) {
22         for (int j=0; j<8; j++){
23             digitalWrite(13, HIGH); // set the internal LED on
24             digitalWrite(12, LOW); // set the external LED1 off
25             digitalWrite(11, HIGH); // set the external LED2 on
26             delay(500); // wait for a 1/2 second
27             digitalWrite(13, LOW); // set the internal LED off
28             digitalWrite(12, HIGH); // set the external LED on
29             digitalWrite(11, LOW); // set the external LED off
30             delay(500); // wait for a 1/2 second
31         }
32         digitalWrite(13, HIGH); // set the internal LED on
33         digitalWrite(12, HIGH); // set the external LED1 on
34         digitalWrite(11, HIGH); // set the external LED2 on
35     }
36     count=1;
37 }
                    
```

Normal text file length: 1403 lines: 37 Ln: 1 Col: 27 Sel: 0|0 UNIX UTF-8 INS

University of Florida, EEL 5666
© Dr. A. Antonio Arroyo