CMPE 311 - C Programming & Embedded Systems

Discussion I (Version 2.0)

August 31, 2014

Version History

Version 2.1 - (August 31, 2015) - Addition Pin Connections Section and Document Verification.

Version 2.0 - (August 26, 2014) - Updates based on Atmel's new IDE, installation folders, locations etc.

Version 1.0 - (August 26, 2013) - Initial Document Adapted from Dr. Ryan Robucci's tutorials

CONTENTS

Objectives

1. **Introduce AVR ATmega 169P Microcontroller**[1]

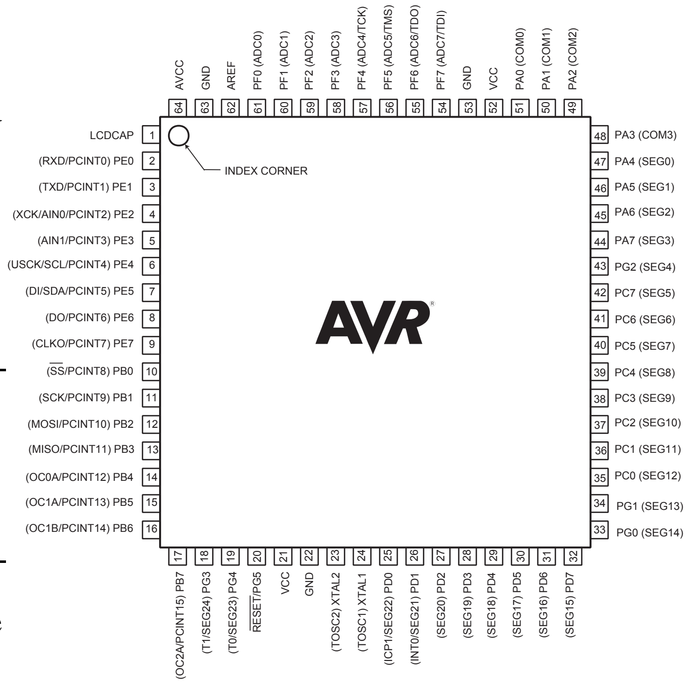
2. **Create a project using Atmel Studio 6.2**

3. **Implement and Debug an assembly level project on the ATmega 169P**

[1] Figures in this document have been adapted from the Atmel ATmega169P datasheet

1 Introduction

1.1 The ATmega169P



1. 8-bit architecture
2. 32 general purpose registers labeled R0. . . R31
3. **General Purpose I/O**
   - Organized in banks of 8
   - Each pin can independently be configured as input or output
   - Internal Pull-Up resister available
   - Pins also have special purpose functions when internal peripherals are used
   - Ports, addresses, registers, masks, etc. specific to each micro-controller are defined in a .inc file for asm and .h file for C.
   - Following is an excerpt from ”m169Pdef.inc” located at

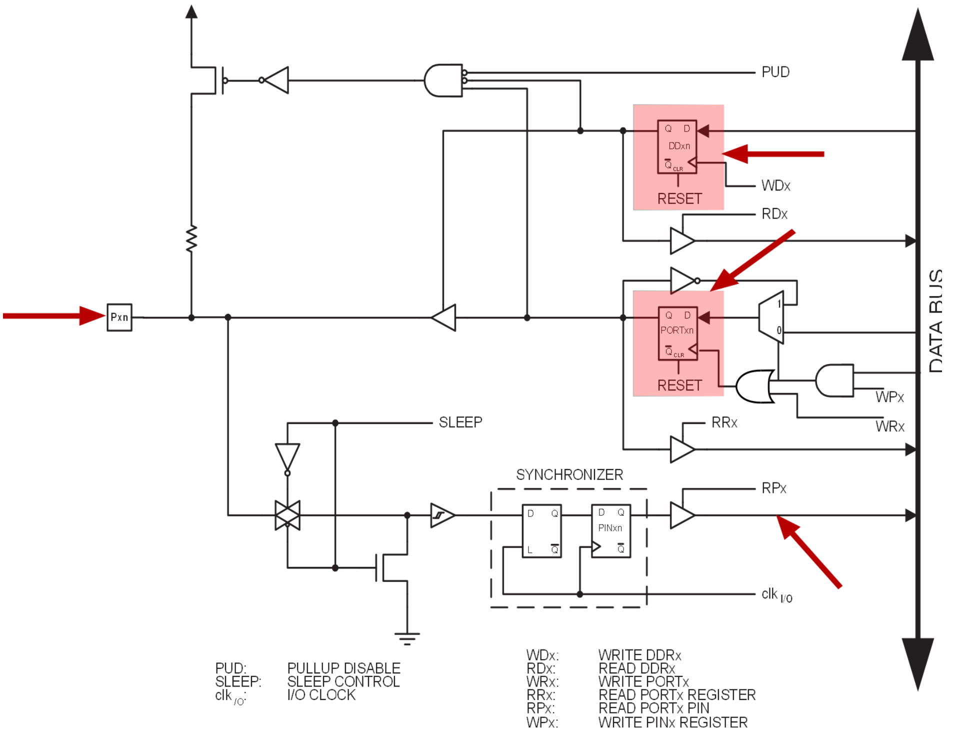
     C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.1117\avrassembler\include

     . . .

     ```
     .equ PINB = 0x03
     .equ DDRB = 0x04
     .equ PORTB = 0x05
     ```

1.2 General I/O Programming

- The **ATmega169P** has 7 Ports designated using the letters A,B,C,D,E,F & G (e.g. PORTA)
- The letter **x** is used to denote the port letter and the letter **n** is used to denote the bit number (7...0)
- Individual pins are referred to as Pxn (e.g. PA5 is the 6th pin of port A)
- General I/O function of each port can accessed using 3 8-bit registers
  - Data Register - PORTx (Read/Write)
  - Data Direction Register - DDRx (Read/Write)
  - Port Input Pins - PINx (Read only)
- Data direction and internal pull-up resisters may be modified on a per pin basis, not just per-port (e.g. PA3 and PA0 can be inputs with an internal pull-up enabled only on PA3 , while PA7, PA6, PA5, PA4, PA2, and PA1 are set as outputs)[1].
- Setting Input Vs Output
  - If DDxn is written logic one, Pxn is configured as an output pin.
  - If DDxn is written logic zero, Pxn is configured as an input pin.



- ▶ Control as Output Pin
  - ▶ If a bit PORTxn, is written logic one when the pin Pxn is configured as an output pin, the port pin is driven high (one).
  - ▶ If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).
- ▶ Control as Input Pin
  - – If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated.
- **NOTE** writing a logic one to a bit in the PINx Register is a special operation, and will result in a toggle in the corresponding bit in the Data Register.

1.3 Useful Tips

Pull-up resisters & push buttons

With an external pull-up resistor, and a normally-open push button, the input pin will read a high state when the button is not pressed. When the button is pressed (the switch closes), it connects the input pin directly to ground, thus creating a low state at the pin.

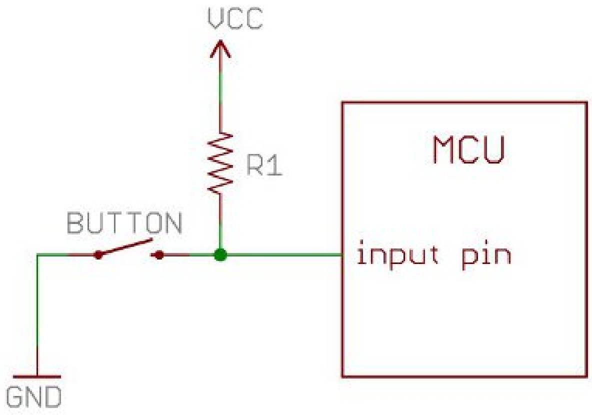
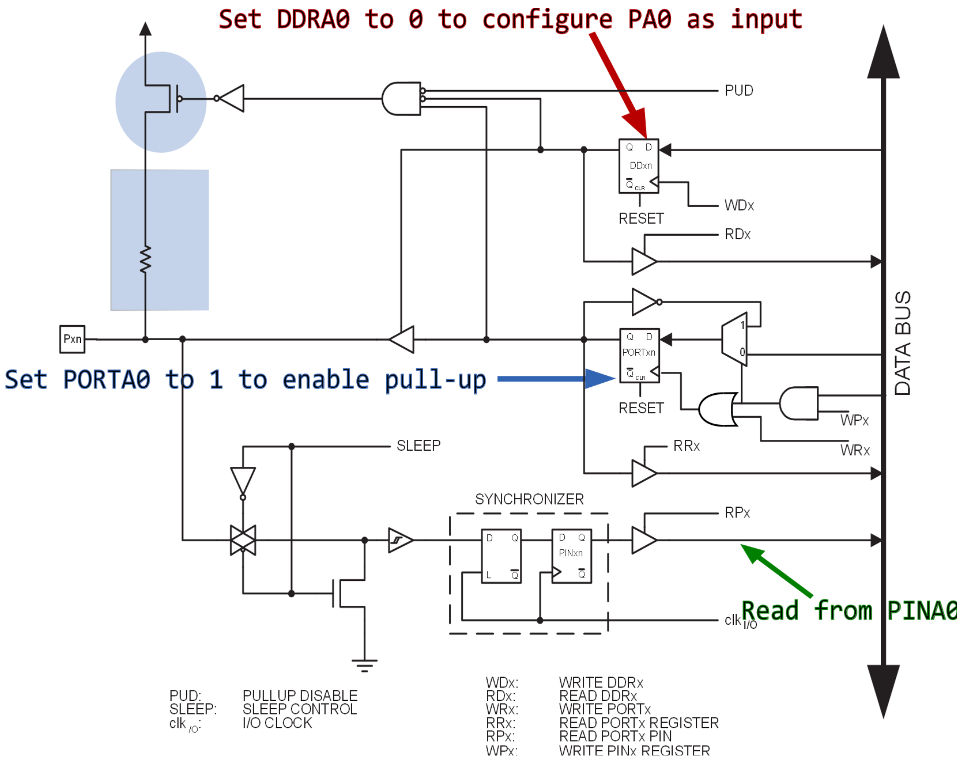


Image obtained from sparkfun.com [2]

AVRs have internal pull-up resistors that can also be used to avoid the need for resistors at the board level.


Set DDRA0 to 0 to configure PA0 as input

Set PORTA0 to 1 to enable pull-up

Read from PINA0

2 Creating Projects in Atmel Studio

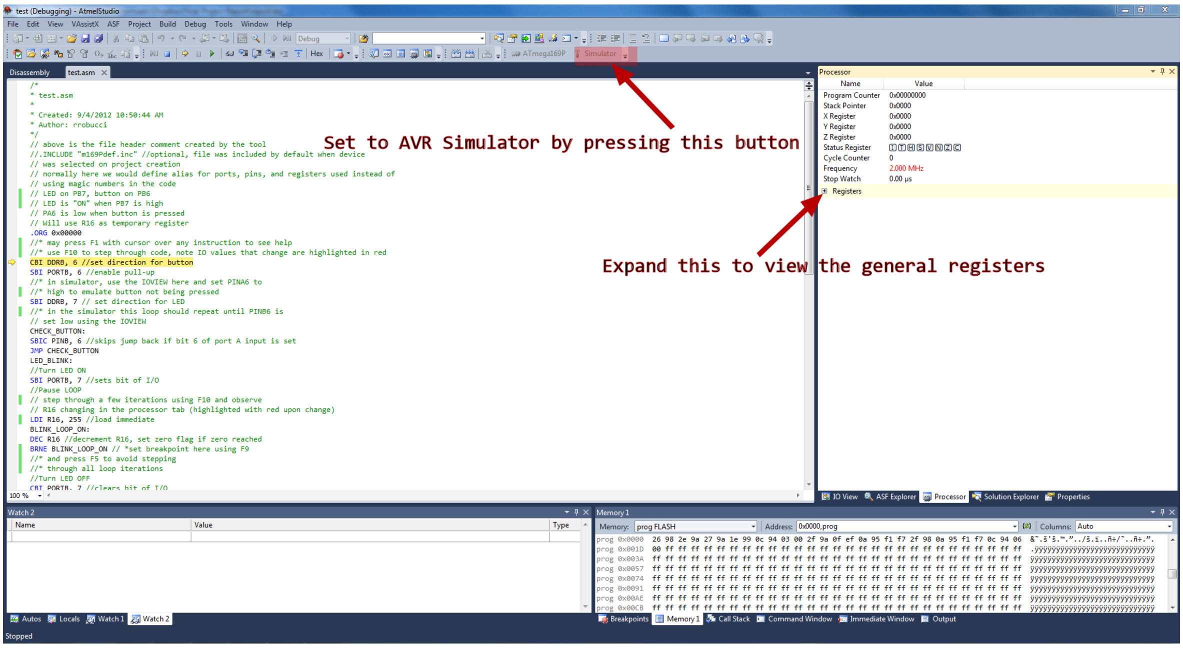
2.1 The Atmel Studio

Students who wish to work from their laptops may install the Atmel Studio version 6.2 package which is available for free download from here.

Projects in Atmel Studio are developed using the Atmel Software Framework (ASF).

2.2 Steps to create and debug projects

1. Create a new assembly project, selecting ATmega169P
2. Type in the code on the next page



test (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Hex
ATmega169P
Simulator
Disassembly
test.asm
100 %
Watch 2
Name
Value
Type
Autos Locals Watch 1 Watch 2
Stopped
IO View ASF Explorer Processor Solution Explorer Properties
Memory 1
Breakpoints Memory 1 Call Stack Command Window Immediate Window Output

Set to AVR Simulator by pressing this button

Expand this to view the general registers

```
/*
 * test.asm
 *
 * Created: 9/4/2012 10:50:44 AM
 * Author: rrobucci
 */
// above is the file header comment created by the tool
//.INCLUDE "m169Pdef.inc" //optional, file was included by default when device
// was selected on project creation
// normally here we would define alias for ports, pins, and registers used instead of
// using magic numbers in the code
// LED on PB7, button on PB6
// LED is "ON" when PB7 is high
// PA6 is low when button is pressed
// Will use R16 as temporary register
.ORG 0x00000
//* may press F1 with cursor over any instruction to see help
//* use F10 to step through code, note IO values that change are highlighted in red
CBI DDRB, 6 //set direction for button
SBI PORTB, 6 //enable pull-up
//* in simulator, use the IOVIEW here and set PINA6 to
//* high to emulate button not being pressed
SBI DDRB, 7 // set direction for LED
//* in the simulator this loop should repeat until PINB6 is
// set low using the IOVIEW
CHECK_BUTTON:
SBIC PINB, 6 //skips jump back if bit 6 of port A input is set
JMP CHECK_BUTTON
LED_BLINK:
//Turn LED ON
SBI PORTB, 7 //sets bit of I/O
//Pause LOOP
// step through a few iterations using F10 and observe
// R16 changing in the processor tab (highlighted with red upon change)
LDI R16, 255 //load immediate
BLINK_LOOP_ON:
DEC R16 //decrement R16, set zero flag if zero reached
BRNE BLINK_LOOP_ON // *set breakpoint here using F9
//* and press F5 to avoid stepping
//* through all loop iterations
//Turn LED OFF
CBI PORTB, 7 //clears bit of I/O
```

Processor

Name	Value
Program Counter	0x00000000
Stack Pointer	0x0000
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	0
Frequency	2.000 MHz
Stop Watch	0.00 µs
⊞ Registers	

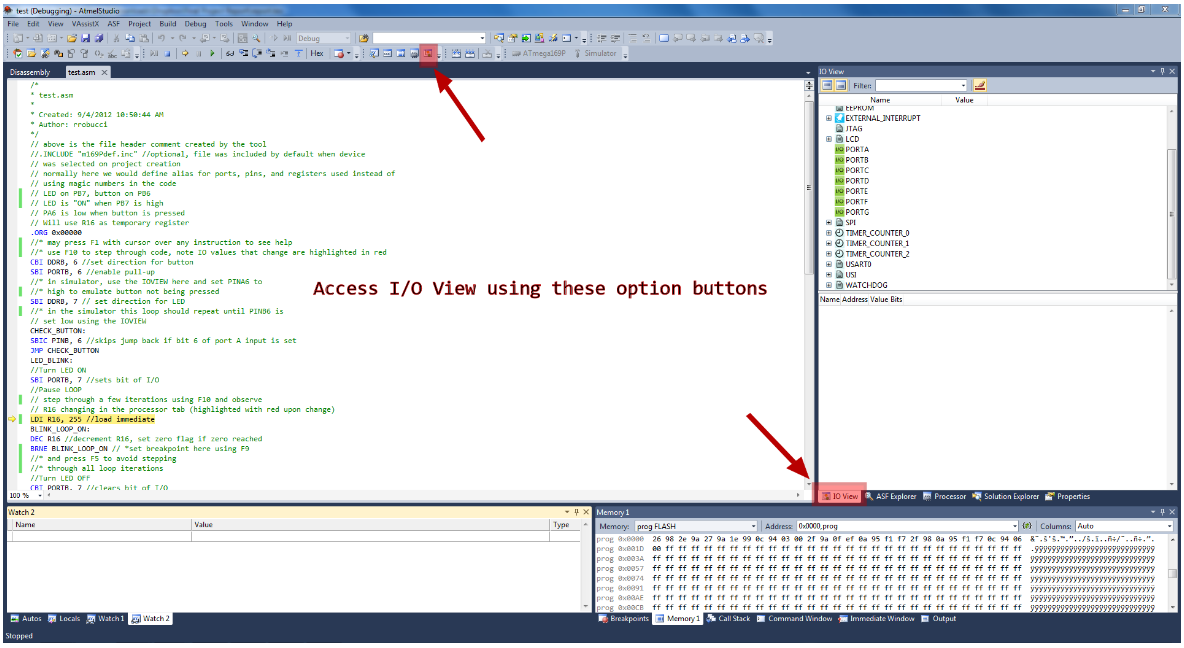
Memory: prog FLASH Address: 0x0000,prog Columns: Auto

```
prog 0x0000  26 98 2e 9a 27 9a 1e 99 0c 94 03 00 2f 9a 0f ef 0a 95 f1 f7 2f 98 0a 95 f1 f7 0c 94 06  &˜.š'š.™.”../š.ï..ñ÷/˜..ñ÷.”.
prog 0x001D  00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x003A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0057  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0074  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0091  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00AE  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00CB  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

3. Start Debugging by selecting Debug − > Start Debugging and Break

4. The Debugger stops on first executable line

5. On the right hand side , select processor tab, and change the frequency to 2.0 MHz



Access I/O View using these option buttons

```
/*
* test.asm
*
* Created: 9/4/2012 10:50:44 AM
* Author: rrobucci
*/
// above is the file header comment created by the tool
//.INCLUDE "m169Pdef.inc" //optional, file was included by default when device
// was selected on project creation
// normally here we would define alias for ports, pins, and registers used instead of
// using magic numbers in the code
// LED on PB7, button on PB6
// LED is "ON" when PB7 is high
// PA6 is low when button is pressed
// Will use R16 as temporary register
.ORG 0x00000
//* may press F1 with cursor over any instruction to see help
//* use F10 to step through code, note IO values that change are highlighted in red
CBI DDRB, 6 //set direction for button
SBI PORTB, 6 //enable pull-up
//* in simulator, use the IOVIEW here and set PINA6 to
//* high to emulate button not being pressed
SBI DDRB, 7 // set direction for LED
//* in the simulator this loop should repeat until PINB6 is
// set low using the IOVIEW
CHECK_BUTTON:
SBIC PINB, 6 //skips jump back if bit 6 of port A input is set
JMP CHECK_BUTTON
LED_BLINK:
//Turn LED ON
SBI PORTB, 7 //sets bit of I/O
//Pause LOOP
// step through a few iterations using F10 and observe
// R16 changing in the processor tab (highlighted with red upon change)
LDI R16, 255 //load immediate
BLINK_LOOP_ON:
DEC R16 //decrement R16, set zero flag if zero reached
BRNE BLINK_LOOP_ON // *set breakpoint here using F9
//* and press F5 to avoid stepping
//* through all loop iterations
//Turn LED OFF
CBI PORTB, 7 //clears bit of I/O
```

Memory: prog FLASH Address: 0x0000,prog Columns: Auto

```
prog 0x0000  26 98 2e 9a 27 9a 1e 99 0c 94 03 00 2f 9a 0f ef 0a 95 f1 f7 2f 98 0a 95 f1 f7 0c 94 06  &˜.š'š.™.”../š.ï..ñ÷/˜..ñ÷.”.
prog 0x001D  00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x003A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0057  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0074  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0091  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00AE  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00CB  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

3 Demonstration

3.1 Code Listing

Complete the initial setup for the experiment using the supplied AVR Dragon programmer and AVR Butterfly code.

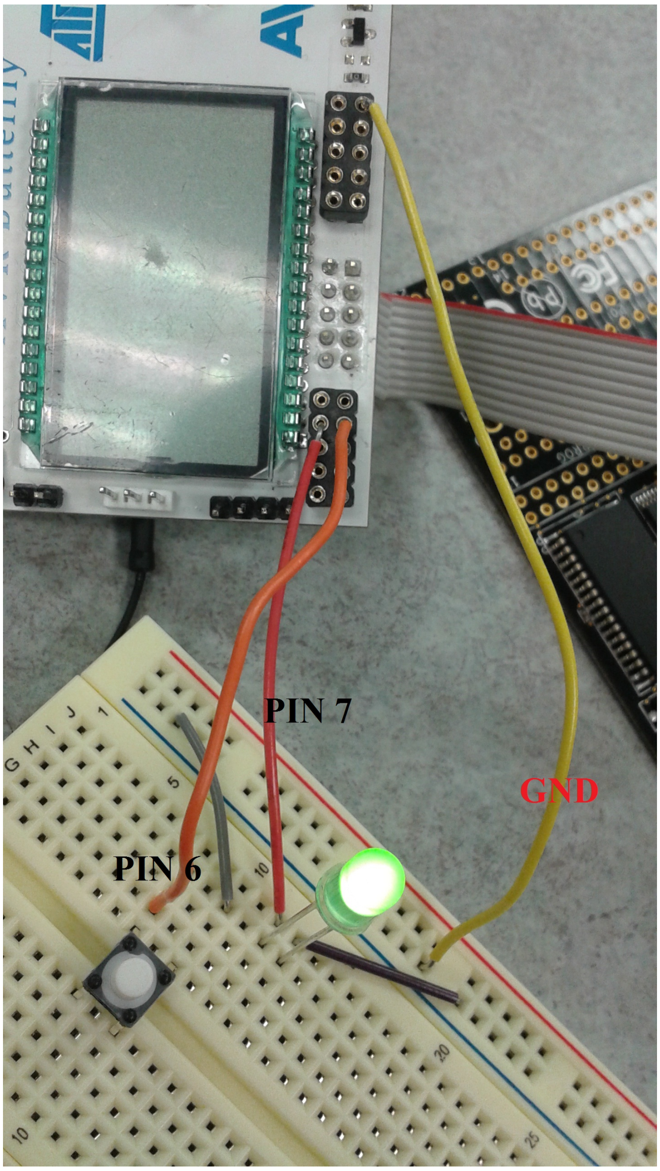
You may use the following code during the discussion session.

```
/* test.asm
* Created: 9/4/2012 10:50:44 AM
* Author: rrobucci
*/
// above is the file header comment created by the tool
//.INCLUDE "m169Pdef.inc" //optional, file was included by
// default when device was selected on project creation
// normally here we would define alias for ports, pins, and
// registers used instead of using magic numbers in the code
// LED on PB7, button on PB6
// LED is "ON" when PB7 is high
// PB6 is low when button is pressed
// Will use R16 as temporary register
.ORG 0x00000
//* may press F1 with cursor over any instruction to see help
//* use F10 to step through code, note IO values that change
//* are highlighted in red
CBI DDRB, 6 //set direction for button
SBI PORTB, 6 //enable pull-up
//* in simulator, use the IOVIEW here and set PINB6 to
//* high to emulate button not being pressed
```

```
SBI DDRB, 7 // set direction for LED
//* in the simulator this loop should repeat until PINB6 is
// set low using the IOVIEW
CHECK_BUTTON:
//skips jump back if bit 6 of port B input is set
SBIC PINB, 6
JMP CHECK_BUTTON
LED_BLINK:
//Turn LED ON
SBI PORTB, 7 //sets bit of I/O
//Pause LOOP
// step through a few iterations using F10 and observe
// R16 changing in the processor tab (highlighted with red
// upon change)
LDI R16, 255 //load immediate
BLINK_LOOP_ON:
DEC R16 //decrement R16, set zero flag if zero reached
BRNE BLINK_LOOP_ON // *set breakpoint here using F9
//* and press F5 to avoid stepping through all loop
// iterations
//Turn LED OFF
CBI PORTB, 7 //clears bit of I/O
LDI R16,127 //Pause
BLINK_LOOP_OFF:
DEC R16
BRNE BLINK_LOOP_OFF
JMP LED_BLINK//* using F9, set breakpoints on the SBI and
// CBI code lines above and repeatedly
// press F5 to see the LED pin toggle
```

3.2 Pin Connections

Figure below shows pin connections between AVR Butterfly and Led Circuit corresponding to code in 3.1



REFERENCES

[1] (2013, August) Atmega169p / atmega169pv datasheet. [Online]. Available: http://www.atmel.com/Images/doc8018.pdf

[2] (2013, August) What is a pull-up resistor ? Sparkfun. [Online]. Available: https://learn.sparkfun.com/tutorials/pull-up-resistors/what-is-a-pull-up-resistor