PUSH BUTTON

Revision 04.02.11

_____

Class

_____

Instructor / Professor

© 2003 Axiom Manufacturing
2813 Industrial Lane - Garland, TX 75041

Phone: 972-926-9303
http://www.axman.com
email: sales@axman.com

CONTENTS

1 REQUIREMENTS

1.1 Hardware

To complete this lab, **the following hardware is required**:

- Axiom CME11E9 EVBU Development Kit
- PC running Windows OS
- Push Button Lab Kit which includes:
  - (1) Push Button
  - (3) 10k ohm resistor ¼ w
  - (3) Jumper wires

1.2 Software

The CME11E9 EVBU board used in this experiment comes with all the software needed to complete this project.

There are many additional utilities included on the boards support CD that can make developing your own projects easier. The CD contains example source code, documentation and experiments for all Axiom development boards. You can also download the latest versions of the software and documentation free from our web site at: www.axman.com.

Also included is an integrated development environment, called AxIDE, for communicating with the board (via the serial port) and for reading and writing its flash memory. To complete this Lab, you should have this program installed on a PC running Microsoft Windows (95/98/2000/XP).

NOTE: This lab does not teach you how to use the AxIDE terminal interface or the Buffalo Monitor program to modify memory and upload programs. It assumes you're already familiar with these procedures. Refer to your board manual for details on installing and using this software, including a tutorial for using AxIDE.

CAUTION

Devices used in this lab are static sensitive and easily damaged by mishandling. Use caution when installing wires and devices onto the board to prevent bending the leads.

Experiments should be laid out in an orderly fashion. Start your lab time with the bench clean and free of metal objects and leave the lab area in a clean condition by picking up loose parts, wires and small objects.

2 GETTING STARTED

This lab project will show you how to add a push button as an input device to the microcontroller on your Axiom development board. In this example one normally open (NO) push button is used. This push button contains an SPST switch.

A push button is a switch that makes or breaks electrical contact when a button is pressed. In this example, a resistor is added to the circuit which will normally pull-up the input pin on the microcontroller. This resistor is pulled low when the push button is pressed, so by reading the level on the input pin you can determine if the switch is pressed or released.

Push buttons are common input devices for many products.

3 LAB PROCEDURE

This lab is arranged in a series of simple steps. Each step should be completed before moving on to the next one, which builds on prior ones. Repeat each step as many times as necessary to become familiar with it. You will find it easier to complete more complex experiments after mastering the simple ones.

As an aid to keeping track of location it's a good idea to mark each step as it's completed, since the experiment will fail if anything is skipped.

3.1 Description

This example uses PORTA channel PA0 on the HC11 microcontroller at address $1000. This is an input only port but is also an input capture port. Reading the input register will return the level on the pin or push button.

This lab demonstrates how to manually read the level of the push button by reading the port. The CME-11E9 EVBU has a 10k pull-down resistor on PA0. This lab uses three 10k resistors as pull-ups for the push-button switch. This is equal to a 3.3k resistor and able to pull the line high on PA0 even with the 10k pull-down resistor on PA0.

More advanced projects could use the input compare function of PA0 to allow the microcontroller to be interrupted when a key is pressed.

3.2 Detailed Steps

This section describes how to build the Push Button project and test it with the monitor running on the CME11E9 EVBU. In the next section, you'll see how to write a simple program that responds when someone presses the push button.

> **NOTE**: To complete these steps you must be familiar with modifying register contents on your board. For example, to read PORTA register with the Buffalo monitor, type `MM 1000` at the monitor prompt and press <enter>. The monitor will display the value read from address $1000. Type Help for more commands.
>
> You can use a different monitor or debugger if you prefer, such as the GNU GDB.

1. Verify power is NOT applied to development board.

2. Install the push button and resistor on the breadboard area as shown in **Figure 1**.

3. Install the jumper wires on the board as follows:

MCU PORT	**Breadboard**
GND	GND
VCC	VCC
PA0	OUT

4. Install the MODA jumper on the EVBU board and remove jumpers MODB and MEM_EN, to configure the board for single chip operation, then apply power to the board.

5. Read PORTA. It should have a binary value of "xxxxxxx1". In other words, bit 0 should be high when the push button is not depressed. Bits 1 – 7 are "don't care".

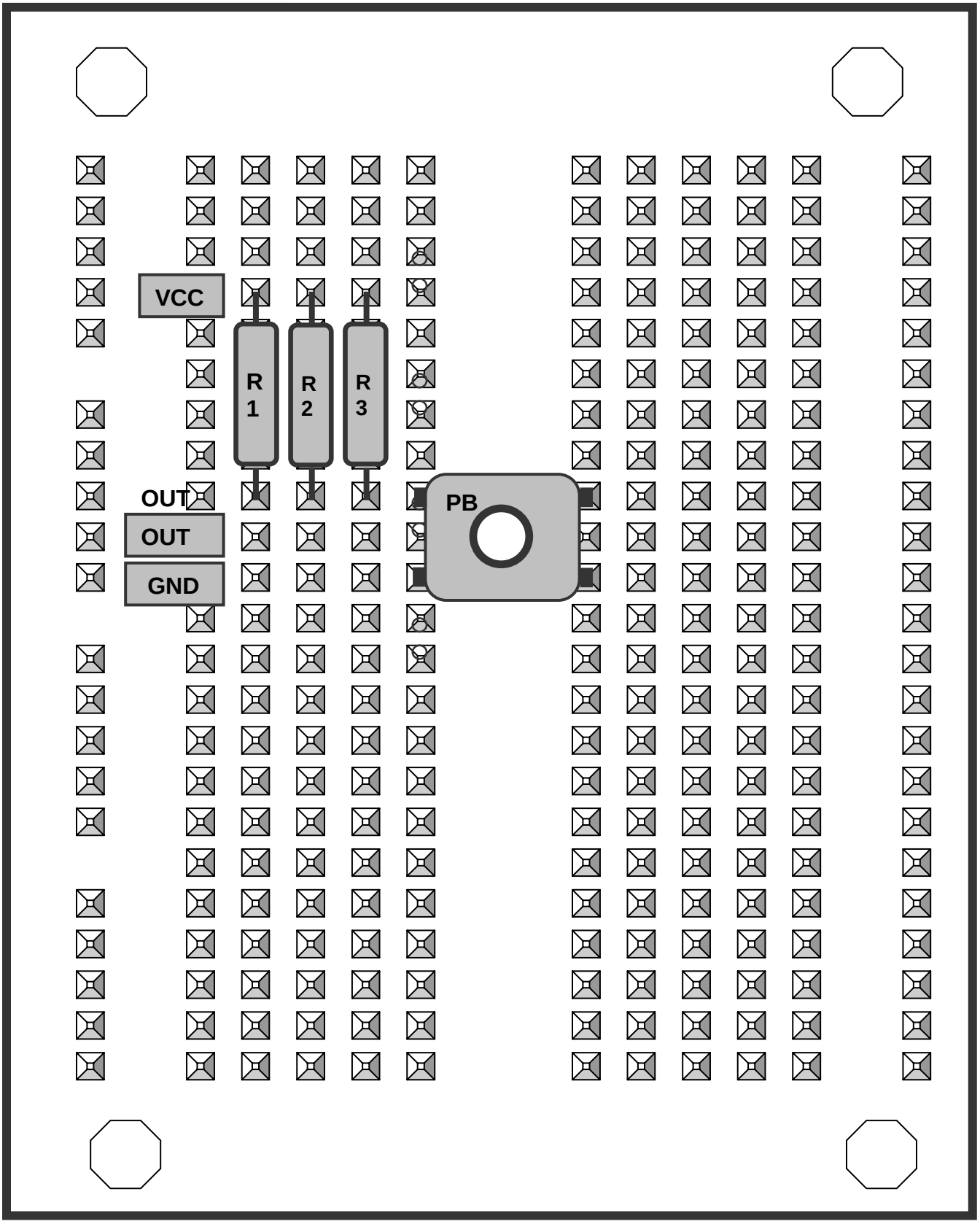
6. Press and hold the push button while reading PORTA. It should have a binary value of "xxxxxxx0". Bit 0 should be low while the push button is pressed.

You can see how easy a push button can be added to a microcontroller. By using a single I/O line you can read the position of the button. Several push buttons can be added in the same manner. A common design is an array of push buttons for input, such as a keypad.

Push buttons come in all types: normally open, normally closed, momentary, push on, push off, illuminated and many others. They serve a useful and common function in microcontroller projects.

An important thing to remember about this type of input device is that it is mechanical. When the push button is pressed the electrical contacts can "bounce" open and closed many times. This can result in fake button presses and releases. To compensate, software commonly uses a "de-bounce" routine which counts the number of times a high or low input is read continuously. Only after a set number of continuous input readings (high or low) is the input considered valid. This set number (or de-bounce time) is unique to each circuit and is usually determined by testing.

Figure 1



4 Push Button PROGRAM

The previous section described how to read a push button manually using a debugger. While this method is useful for testing and experimenting, once hardware is working you'll want to write a software program to read the push button.

This section describes how to write such a program in assembly language. This is a simple program requiring the microcontroller to continually monitor the push button. A more advanced program might use an interrupt driven pin instead.

The input is not de-bounced in this example, but should be in a finished product. A simple delay routine is used instead, to allow the line to settle. This is not as reliable as a de-bounce counter however, so you may see fake presses or releases in this test.

The source code is listed at the end of this section. Both source code and assembled executable can also be downloaded from the Axiom web site: **www.axman.com**.

If viewing this on your PC, you can copy and paste the source code below into a text editor (such as notepad) then save and assemble it using AxIDE. Refer to the owner's manual of your board for instructions on creating software and running programs for your development board.

4.1 Program Description

This program monitors the level on PA0. When the button is pressed a message is displayed. After a short delay the button is monitored until it is released and a push-button released message is displayed.

Finally, the program jumps back to the beginning and repeats until you press reset or remove power.

4.2 Running the Program

1. Upload the assembled program PSHB1D.S19 into the RAM on your board. This program starts at address $0100, which is internal memory on the CME-11E9 EVBU. The source code for this program is shown below.

2. Execute the program by typing `call 100` in the terminal and pressing <enter>.

3. Press and hold the button and verify the pressed message is displayed on the terminal.

4. Release the button and verify the released message is displayed.

4.3 Push Button Program Source Code

```
* HC11 Push Button Lab Experiment
*
PORTA       equ     $1000       ; PORT A input resister
OUTCRLF     equ     $FFC4       ; Buffalo routine CR/LF
OUTSTRG     equ     $FFC7       ; Output a String
*
            org     $0100       ; start address

Loop
            ldaa     #$01       ; mask bit
            bita     PORTA      ; Push Button Pressed?
            beq      Press      ; report Pressed
            bra      Loop

* Push Button is pressed
Press
            jsr      OUTCRLF    ; new line
            ldx      #Mes_Pr    ; Pressed Message
            jsr      OUTSTRG    ; Output Message
            jsr      Delay
Press_L
            ldaa     #$01       ; Mask Bit
            bita     PORTA      ; Push Button Pressed?
            beq      Press_L    ; wait for released

* Push Button is NOT pressed
            ldx      #Mes_RL    ; Released Message
            jsr      OUTSTRG    ; Output Message
            bra      Loop

Delay
            ldy      #$2000
DelayA
            dey
            bne      DelayA
            rts

* Messages
Mes_Pr
            fcc     ' Push Button Pressed Detected'
            fcb     4
Mes_RL
            fcc     ' Push Button Released Detected'
            fcb     4
```

5 QUIZ

Answer the following questions based on the example presented in this lab.

1. In this example the push button contains?

A. NC Switch **B**. NO Switch **C**. DPDT Switch **D**. Transistor

2. Running program PSHB1D, what happens when the push button is pressed?

A. Message **B**. Halts **C**. Runs **D**. Counts

3. A push button should be?

A. Lit **B**. Toggled **C**. De-bounced **D**. Sealed

4. Is PA0?

A. Input Capture **B**. Analog **C**. Output Compare **D**. All of these

5. How many push buttons can be used on a microcontroller?

A. 1 **B**. Input Port Count **C**. Output Port Count **D**. Serial Port Count

BONUS QUESTION: What type port is best for a push button?

A. Data Port **B**. Analog Port **C**. Digital Port **D**. Input Capture

6 SCHEMATIC