

PUSH BUTTON

Revision 04.02.11

Class

Instructor / Professor

LICENSE

You may use, copy, modify and distribute this document freely as long as you include this license and the Axiom Manufacturing copyright at the bottom of this page. You can download the latest version of this document from the Axiom website: **www.axman.com**

CONTENTS

1	REQUIREMENTS.....	3
1.1	HARDWARE	3
1.2	SOFTWARE	3
2	GETTING STARTED.....	4
3	LAB PROCEDURE	4
3.1	DESCRIPTION.....	4
3.2	DETAILED STEPS.....	4
4	Push Button PROGRAM.....	7
4.1	PROGRAM DESCRIPTION	7
4.2	RUNNING THE PROGRAM.....	7
4.3	PUSH BUTTON PROGRAM SOURCE CODE.....	8
5	QUIZ.....	10
6	SCHEMATIC	11

1 REQUIREMENTS

1.1 Hardware

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

- Axiom CML-12S256 Development Kit
- PC running Windows OS
- Push Button Lab Kit which includes:
 - (1) Push Button
 - (1) 10k ohm resistor ¼ w
 - (3) Jumper wires

1.2 Software

The CML12S256 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 MON12 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 PORT J on the DP256 microcontroller. This is a multiple functional, bi-directional port. Bit 0 is configured as an input using the direction register (DDRJ) at address \$026A. You can see in the DP256 microcontroller reference manual that port J data register (PTJ) is located at address \$0268 and port J input register (PTIJ) is located at \$0269.

Reading the input register will return the level on the pin or push button. In more advanced projects the key-wake-up function of this port could be used allowing the microcontroller to be interrupted automatically 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 CML12S256. 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 write \$00 to the DDRJ register with the MON12 monitor, type MM 026A at the monitor prompt and press <enter>. Then type 00 <enter> then . to end modify mode. 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 jumper wires on the board as follows:

MCU PORT -----Breadboard

GND-----GND

+5V-----+5V

PJ0-----OUT

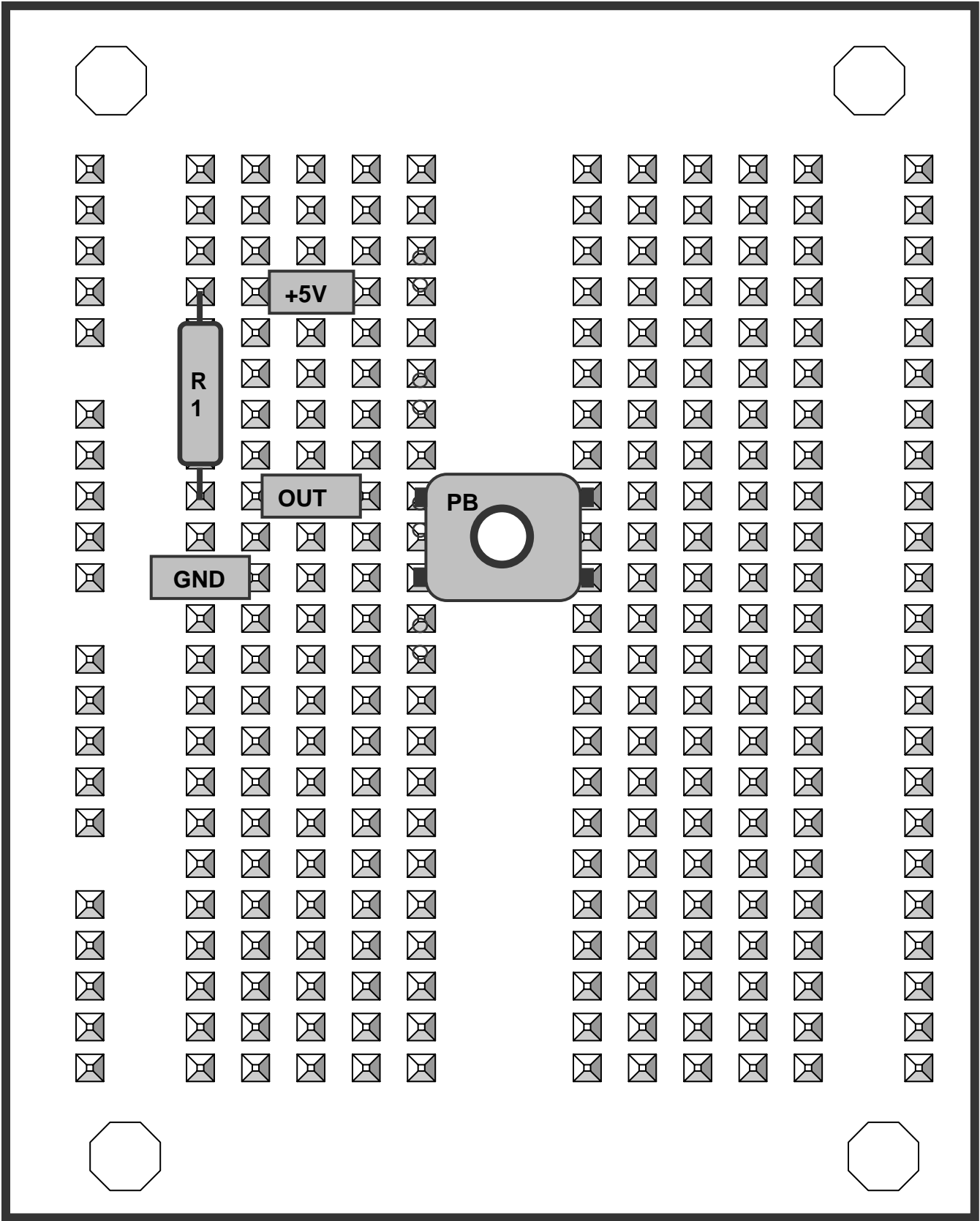
4. Configure the CML12S256 board to start the MON12 debugger by setting the jumpers to their default positions (No_Auto, MEM_EN, JP1 and JP2 – ON. MODC and ECS - OFF)
5. Write \$00 to the DDRJ register. This configures PORT J bit 0 as input.
6. Read PTIJ. It should have a binary value of “xxxxxxx1”. In other words, bit 0 should be high when the push button is not depressed. Bits 1 – 7 are “don’t care”.
7. Press and hold the push button while reading PTIJ. 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 the 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

The program first sets the direction register of PORT J bit 0 as an input using the constant mask Bit0.

Next, Bit0 is used to read the state of the push button. When the button is pressed, a message is displayed. After a short delay, the button is monitored until it is released then another 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 PSHB2D.S19 into the RAM on your board. This program starts at address \$1000, which is internal memory on the CML12S256. The source code for this program is shown below.
2. Execute the program by typing `call 1000` 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

```

; PshB2D.asm          Push Button Example for the HC12
;
; Register Equates
SC0SR1:    equ        $00CC          ; Status Register
SC0DRL:    equ        $00CF          ; Data Register
PTJ:       equ        $0268          ; Port J data register
PTIJ:      equ        $0269          ; Port J Input register
DDRJ:      equ        $026A          ; Port J direction
Bit0:      equ        $01            ; mask for Bit 0
EOM:       equ        $00            ; End of Message

                org        $1000          ; Start of Program
                bclr       DDRJ,Bit0      ; Set Bit

Loop:        brclr       PTIJ,Bit0,Press ; report pressed
                bra        Loop

;
; Report Button Pressed
Press:
                jsr        CrLf           ; new line
                ldx        #Mes_Pr       ; Pressed Message
                jsr        OutStrg        ; Output Message
                jsr        Delay          ; Delay for de-bounce

Press_L:      brclr       PTIJ,Bit0,Press_L ; Loop until released
                jsr        Delay          ; Delay for de-bounce
                ldx        #Mes_Rl       ; Released Message
                jsr        OutStrg        ; Output Message
                bra        Loop          ; return to beginning

;
; Output A to SCIO
Output:
                brclr       SC0SR1,#$80,Output ; wait for Xtrans to empty
                staa        SC0DRL        ; send character
                rts

;
; send car & line feed
CrLf:
                psha        ; save A
                ldaa        #$0a         ; send car
                bsr        Output
                bsr        Delay
                ldaa        #$0d         ; line feed
                bsr        Output
                bsr        Delay
                pula        ; restore A
                rts

;
; Output string of ASCII bytes starting at x until end of text ($00).
OutStrg:
                jsr        CrLf           ; output carriage-return

OutStrg0:
OutStrg1:
                ldaa        0,X           ; read char into a
                cmpa        #EOM          ; is this end of text?
                beq        OutStrg3      ; jump if yes
                jsr        Output        ; output character
                inx            ; increment pointer
                bra        OutStrg1      ; loop

OutStrg3:
                rts

```



```

; Delay Routine
Delay
    ldy    #$1000
DelayA:
    dey
    bne    DelayA
    rts

;
; Messages
Mes_Pr:
    fcc    ' Push Button Pressed Detected'
    fcb    0
Mes_Rl:
    fcc    ' Push Button Released Detected'
    fcb    0

```

5 QUIZ

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

1. In this example, the push button contains a?
A. NC Switch B. NO Switch C. DPDT Switch D. Transistor
2. Running program PSHB2D, what happens when the push button is pressed?
A. Message B. Halts C. Runs D. Counts
3. A push button must be?
A. Lit B. Toggled C. De-bounced D. Sealed
4. Is PORT J?
A. Input B. Output C. Key-Wake-Up 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. Key-Wake-Up

6 SCHEMATIC

