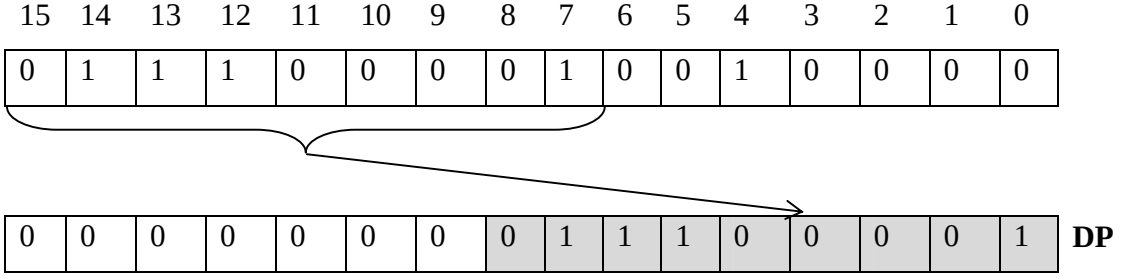
DIGITAL I/O PORTS

RELAYS & SWITCHES INTERFACING

Let us now discuss the programming involved in configuring a pin for digital I/O. As an example, let us configure bit 0 of PORT A as an input pin.

The first task is to set the corresponding bit in the mux control register to 0 in order to configure the pin for I/O. This is bit 0 of the MCRA register. The MCRA register is located at address 7090h. Thus, the value of the data pointer will be:



DP = 00E1h (defined as DP_PF2 in header file `f2407.h`)

The file `f2407.h` defines all the registers in the TMS320LF2407. Thus by including this file, the MCRA register address can be referred to as MCRA in the program. Following is the code segment for setting the MCRA register:

```
.include        "f2407.h"

LDP   #DP_PF2   ;Point to appropriate Data Page
LACC  #0h       ;Pins IOPA0-IOPA3 and IOPB0-IOPB7
SACL  MCRA      ;are configured as I/O pins
```

The next job is to set the pin as input in the PADATDIR register, which is done as follows-

```
LACC  #0h
SACL  PADATDIR  ;Pins IOPA0-IOPA3 are configured as inputs
LACC  PADATDIR  ;Bit 0 of accumulator gives the status of the
                ;IOPA0 pin.
```

Example Program 1:

Write a program to output 1010b to pins IOPA3-0.

Solution:

Registers involved -

MCRA - Bits 3-0

PADATDIR - Bits 3-0 (data)

Bits 11-8 (direction)

```
LDP   #DP_PF2   ;Point to appropriate Data Page
LACC  #0h       ;Pins IOPA0-IOPA3 are
SACL  MCRA      ;configured as I/O pins
LACC  #0f0ah    ;Pins IOPA3-IOPA0 are configured as outputs and
SACL  PATDATDIR ;IOPA3, IOPA1 are set=1 & IOPA2, IOPA0 are set=0.
```

Example Program 2:

Write a program to do the following-

Read data from pins IOPA3-0 of port A and output the data to pins IOB3-0 of port B.

Solution:

Registers involved :

MCRA – Bits 11-8 (IOPB3-0)

MCRB – Bits 2, 3, 4, and 5 (IOPC2, 3, 4, and 5), which allow +5V tolerant inputs

PCDATDIR - Bits 2, 3, 4, and 5 (data for IOPC2, 3, 4, and 5)

Bits 10, 11, 12, and 13 (direction for IOPC2, 3, 4, and 5)

PBDATDIR - Bits 3-0 (data for IOPB3-0)

Bits 11-8 (direction for IOPB3-0)

```
LDP   #DP_PF2   ;Point to appropriate Data Page
LACC  #0h       ;Pins IOPC2,3,4,5 and IOPB0-IOPB7 are
SACL  MCRA      ;configured as I/O pins
SACL  MCRB
LACC  #0h
SACL  PCDATDIR  ;Pins IOPC2, 3, 4, and 5 are configured as inputs
```

```
LACC  PCDATDIR  ;Read contents of IOPC2,3,4,5 into accumulator
AND   #003Ch    ;Mask the bits IOPC2,3,4,5
OR    #0F00h    ;Set IOPB3-0 as outputs and
SACL  PBDATDIR  ;write data read from IOPC2,3,4,5 to IOPB3-0.
```

Interfacing switches to TMS320LF2407 EVM ports

Switches are perhaps the simplest digital input devices that can be interfaced to the DSP. Figure 3.8 shows a 4 position DIP switches connected to the digital port A of the TMS320LF2407.

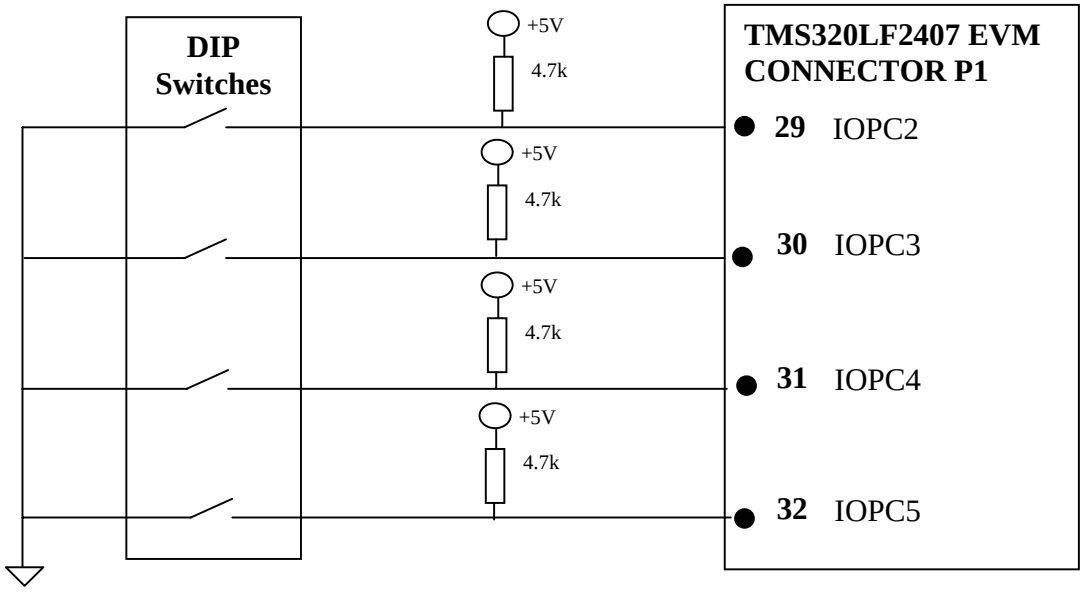


Figure 3.8 *DIP switches interface circuit*

When a switch is closed, a logic HIGH voltage (+5V) is applied to the corresponding input port; when it is opened the voltage at the input port is pulled to logic LOW by the 4.7k resistor.

Example programs for controlling relays and switches from the DSP EVM

The following are example programs that use the general purpose I/O port to turn on 4 AC light bulbs using relays, and input digital data from a 2 position DIP switches. The relays and their interface circuits are connected to port B pin IOPB0, IOPB1, IOPB2, and IOPB3, while the DIP switches uses port A pins IOPB0, IOPB1, IOPB2 and IOPB3. The interface circuits are shown in Figure 3.9.

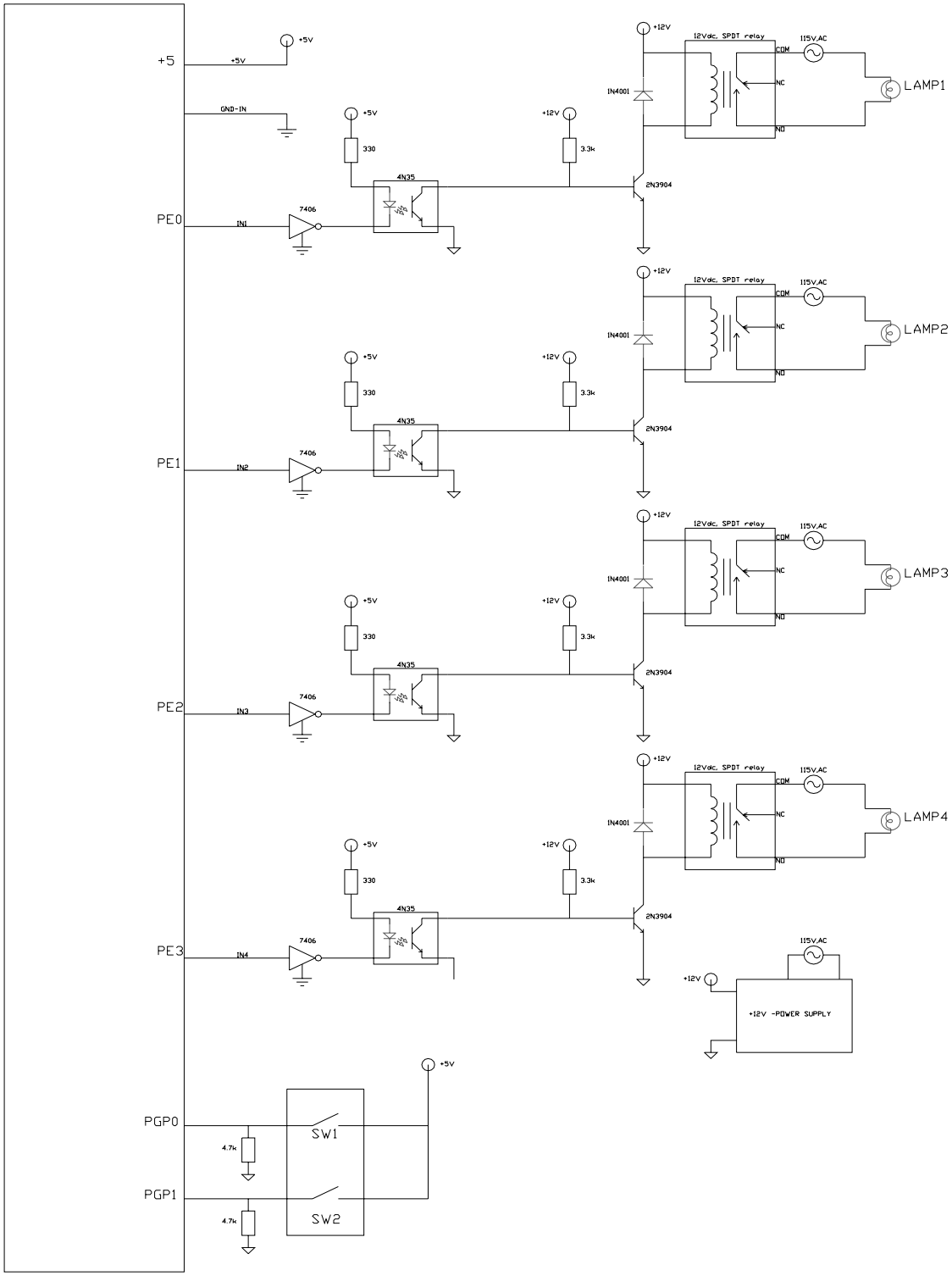


Figure 3.9 *Interface circuit for example programs*

Example Program 3

Write a program that reads the status of the DIP switches and accordingly turns on the lamps; i.e. if SW1 = 1, LAMP1 = OFF; SW2=1, LAMP2=OFF etc.

Let us connect the DP switches 1-4 to IOPC2, 3, 4, and 5 and LAMPS1-4 to IOPB0-3. Thus IOPC2, 3, 4, and 5 will be configured as inputs and IOPB3-0 will be configured as outputs.

Registers involved – the same as **Example Program 2**

```
.include f2407.h

LDP   #DP_PF2   ;Point to appropriate Data Page
LACC  #0h       ;Pins IOPC2,3,4,5 and IOPB0-IOPB7
SACL  MCRA      ;are configured as I/O pins
SACL  MCRB
SACL  PCDATDIR  ;Pins IOPC2,3,4,5 are configured as inputs
LACC  PCDATDIR  ;Read status of DIP switches into accumulator
AND   #003Ch    ;Mask the bits IOPC2,3,4,5
OR    #0F00h    ;Set IOPB3-0 as outputs and
SACL  PBDATDIR  ;write data read from DIP switches to the lamps
```

Example Program 4

Write a program that reads the status of the DIP switches 1 and 2 and turns on/off the lamps as per the following logic.

Inputs		Outputs			
SW2	SW1	LAMP4	LAMP3	LAMP2	LAMP1
0	0	OFF	OFF	OFF	ON
0	1	OFF	OFF	ON	OFF
1	0	OFF	ON	OFF	OFF
1	1	ON	OFF	OFF	OFF

Let the switches 1 and 2 be connected to IOPC2 and IOPC3 respectively. The LAMPS1-4 be connected to IOPB0-3. The logic table therefore becomes-

IOPC3-2	**IOPB3-0**
00b	1110b
01b	1101b
10b	1011b
11b	0111b

Note: Logic 0 turns a lamp ON. Logic 1 turns the lamp OFF

Registers involved:

MCRA - Bits 9-8 (IOPB3-0)

MCRB - Bits 3-2 (IOPC3-2)

PBDATDIR - Bits 3-0 (data for IOPB3-0)

Bits 11-8 (direction for IOPB3-0)

PCDATDIR - Bits 3-2 (data for IOPC3-2)

Bits 11-10 (direction for IOPC3-2)

Here we will introduce the branch instruction. The TMS320C240x has 2 branch instructions: unconditional branch and unconditional branch.

B *pma*[, *ind*[, **AR***n*]]

Where-

pma: 16-bit program memory address

n: value from 0 to 7 designating the next auxiliary register

ind: one of the 7 indirect addressing options

The program control is unconditionally transferred to the specified pma which can be either a symbolic or numeric address.

BCND *pma, cond1* [cond2] [,….]

Where-

pma: 16-bit program memory address

Cond	Condition
EQ	ACC = 0
NEQ	ACC ≠ 0
LT	ACC < 0
LEQ	ACC ≤ 0
GT	ACC > 0
GEQ	ACC ≥ 0
NC	C = 0
C	C = 1
NOV	OV = 0
OV	OV = 1
BIO	BIO low
NTC	TC = 0
TC	TC = 1
UNC	Unconditionally

The program control is unconditionally transferred to the specified pma when all the required conditions are met.

The basic logic that will be followed in this program is:

If DIP switch status = 0, then branch to location which turns on lamp 1, exit
Else if DIP switch status=2, then branch to location which turns on lamp 3, exit
Else if DIP switch status=1, then branch to location which turns on lamp 2, exit
Else, turn on lamp 4, exit.

The main part of the code is as below (the initialization and PHANTOM ISR sections should be added for practical operations)-

```
       .bss invar,1     ;input variable
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Setup shared I/O pins
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #DP_PF2                 ;set data page

        SPLK    #0000000000000000b,MCRA ;group A pins all set to be I/O
pins
*                |||||||||||||||
*                FEDCBA9876543210
* bit 15        0:      0=IOPB7,     1=TCLKINA
* bit 14        0:      0=IOPB6,     1=TDIRA
* bit 13        0:      0=IOPB5,     1=T2PWM/T2CMP
* bit 12        0:      0=IOPB4,     1=T1PWM/T1CMP
* bit 11        0:      0=IOPB3,     1=PWM6
* bit 10        0:      0=IOPB2,     1=PWM5
* bit 9         0:      0=IOPB1,     1=PWM4
* bit 8         0:      0=IOPB0,     1=PWM3
* bit 7         0:      0=IOPA7,     1=PWM2
* bit 6         0:      0=IOPA6,     1=PWM1
* bit 5         0:      0=IOPA5,     1=CAP3
* bit 4         0:      0=IOPA4,     1=CAP2/QEP2
* bit 3         0:      0=IOPA3,     1=CAP1/QEP1
* bit 2         0:      0=IOPA2,     1=XINT1
* bit 1         0:      0=IOPA1,     1=SCIRXD
* bit 0         0:      0=IOPA0,     1=SCITXD

        SPLK    #1111111000000000b,MCRB ;group B pins all set to be I/O
pins
*                |||||||||||||||
*                FEDCBA9876543210
* bit 15        1:      0=reserved,  1=TMS2 (always write as 1)
* bit 14        1:      0=reserved,  1=TMS  (always write as 1)
* bit 13        1:      0=reserved,  1=TD0  (always write as 1)
* bit 12        1:      0=reserved,  1=TDI  (always write as 1)
* bit 11        1:      0=reserved,  1=TCK  (always write as 1)
* bit 10        1:      0=reserved,  1=EMU1 (always write as 1)
* bit 9         1:      0=reserved,  1=EMU0 (always write as 1)
* bit 8         0:      0=IOPD0,     1=XINT2/ADCSOC
* bit 7         0:      0=IOPC7,     1=CANRX
```

```
* bit 6          0:      0=IOPC6,     1=CANTX
* bit 5          0:      0=IOPC5,     1=SPISTE
* bit 4          0:      0=IOPC4,     1=SPICLK
* bit 3          0:      0=IOPC3,     1=SPISOMI
* bit 2          0:      0=IOPC2,     1=SPISIMO
* bit 1          0:      0=IOPC1,     1=BIO*
* bit 0          0:      0=IOPC0,     1=W/R*

        LDP     #DP_PF2      ;set data page
        LACC    #0h             ;ACC = 0000h
        SACL    PCDATDIR   ;store result to PCDATDIR,
                           ;setting IOPC2-3 to be inputs
        NOP             ;Give some time interval for PCDATDIR operation
       LACC PCDATDIR    ;Read status of DIP switches into accumulator
       AND    #000Ch      ;Mask the bits IOPC2-3
       SFR             ;Shift right once
       SFR                    ;Shift right again
       LDP    #invar
       SACL invar       ;Store the status of switches in memory location
       BCND addr1,EQ    ;If switch status = 0, goto addr1
       AND    #01h      ;check if switch status = 2
       BCND addr2,EQ    ;If switch status = 2, goto addr2
       LACC invar
       AND    #02h      ;check if switch status = 1
       BCND addr3,EQ    ;If switch status = 1, goto addr3
       LACC #7h         ;Load valid lamp data for switch status=3 in ACC
       B    addr4
addr1: LACC #0eh      ;Load valid lamp data for switch status=0 in ACC
       B    addr4
addr2: LACC #0bh      ;Load valid lamp data for switch status=2 in ACC
       B    addr4
addr3: LACC #0dh      ;Load valid lamp data for switch status=1 in ACC
       B    addr4
addr4: OR   #0F00h      ;Set IOPB3-0 as outputs and
       LDP    #DP_PF2
       SACL PBDATDIR    ;write appropriate data to tun on/off the lamps

END    B  END
```

Example Program 5

Write a program that turns on the four lamps sequentially from lamp 1 until lamp 4 one by one. Use software delay to keep the lamp on for approximately one second..

Solution:

In this program we will introduce the concept of a sub-routine. A delay subroutine is written to implement a delay of approximately 1 sec. The code segment for the delay subroutine is -

```
mS_DELAY:   LDP   #0h                  ;DP-->0000h-007Fh
            LACC  #6000                ;Load RPT value to GPR0
            SACL  RPT_NUM
            SPLK  #5000,mSEC

mS_LOOP:    LDP   #0h                  ;DP-->0000h-007Fh
            RPT   RPT_NUM              ;6000 cycles = 0.2 ms
            NOP
            LACC  mSEC                 ;Load mSEC or count in ACC
            SUB   #1                   ;Decrement ACC
            SACL  mSEC                 ;Update mSEC
            BCND  mS_LOOP,NEQ          ;Repeat DELAY_LOOP
            RET                        ;Return from DELAY SUBROUTINE
```

The approximate delay can be computed as follows -

Since the CPU clock is 30MHz, each cycle is 33ns. The NOP instruction takes one cycle for execution i.e. it takes 33ns. The RPT instruction repeats the next instruction RPT_NUM times. Thus, the NOP is executed 6000 times which amounts to a delay of 33ns × 6000 = 200μs. As per the ms_LOOP, this is executed mSEC i.e. 5000 times which gives a total delay of 200μs × 5000 = 1 sec.

The main part of the code for the entire program is as follows (the initialization and PHANTOM ISR sections should be added for practical operations)-

```
      .bss mSEC,1
      .bss RPT_NUM,1
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;Setup shared I/O pins
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LDP     #DP_PF2                 ;set data page

        SPLK    #0000000000000000b,MCRA ;group A pins set to be I/O pins
*                |||||||||||||||
*                FEDCBA9876543210
* bit 15         0:      0=IOPB7,     1=TCLKINA
* bit 14         0:      0=IOPB6,     1=TDIRA
* bit 13         0:      0=IOPB5,     1=T2PWM/T2CMP
* bit 12         0:      0=IOPB4,     1=T1PWM/T1CMP
* bit 11         0:      0=IOPB3,     1=PWM6
* bit 10         0:      0=IOPB2,     1=PWM5
* bit 9          0:      0=IOPB1,     1=PWM4
* bit 8          0:      0=IOPB0,     1=PWM3
* bit 7          0:      0=IOPA7,     1=PWM2
* bit 6          0:      0=IOPA6,     1=PWM1
* bit 5          0:      0=IOPA5,     1=CAP3
* bit 4          0:      0=IOPA4,     1=CAP2/QEP2
* bit 3          0:      0=IOPA3,     1=CAP1/QEP1
* bit 2          0:      0=IOPA2,     1=XINT1
* bit 1          0:      0=IOPA1,     1=SCIRXD
* bit 0          0:      0=IOPA0,     1=SCITXD


LAMP1: LDP    #DP_PF2     ;Point to appropriate Data Page
     LACC #0F0eh      ;Configure IOPB3-0 as outputs and turn
     SACL PBDATDIR    ;LAMP1 ON.
     CALL mS_DELAY    ;1 sec delay

     LDP    #DP_PF2     ;point to appropriate data page.
                    ;since the delay sub-routine changes the value
                    ;of DP
     LACC #0F0dh      ;Configure IOPB3-0 as outputs and turn
     SACL PBDATDIR    ;LAMP2 ON.
     CALL mS_DELAY    ;1 sec delay
```

```
     LDP    #DP_PF2     ;point to appropriate data page.
     LACC #0F0bh      ;Configure IOPB3-0 as outputs and turn
     SACL PBDATDIR    ;LAMP3 ON.
     CALL mS_DELAY    ;1 sec delay

     LDP    #DP_PF2     ;point to appropriate data page.
     LACC #0F07h      ;Configure IOPB3-0 as outputs and turn
     SACL PBDATDIR    ;LAMP4 ON.
     CALL mS_DELAY    ;1 sec delay

     B    LAMP1       ;jump unconditionally
;=====================================================================
; Routine Name:  mS_DELAY
;
; Description:   Produces a delay of approximately 1 sec.
;=====================================================================
mS_DELAY:
     LDP    #RPT_NUM  ;DP-->0000h-007Fh
     LACC #6000     ;Load RPT value to GPR0
     SACL RPT_NUM
     SPLK #5000,mSEC

mS_LOOP:
     LDP    #RPT_NUM  ;DP-->0000h-007Fh
     RPT    RPT_NUM   ;6000 cycles = 0.2 ms
     NOP
     LACC mSEC    ;Load mSEC or count in ACC
     SUB    #1      ;Decrement ACC
     SACL mSEC    ;Update mSEC
     BCND mS_LOOP,NEQ ;Repeat DELAY_LOOP
     RET            ;Return from DELAY SUBROUTINE
```

LABORATORY EXPERIMENT 2

DIGITAL INTERFACING: RELAYS AND SWITCHES

Objectives

The objective of this lab session is to familiarize the students with the DSP digital I/O interfacing. In this session, the students will write and test assembly language programs that use TMS320LF2407 general-purpose I/O ports to turn on AC light bulbs using relay circuits and read digital data from DIP switches.

Discussion

General purpose digital I/O ports in the TMS320LF2407

The Relays Interface Circuit

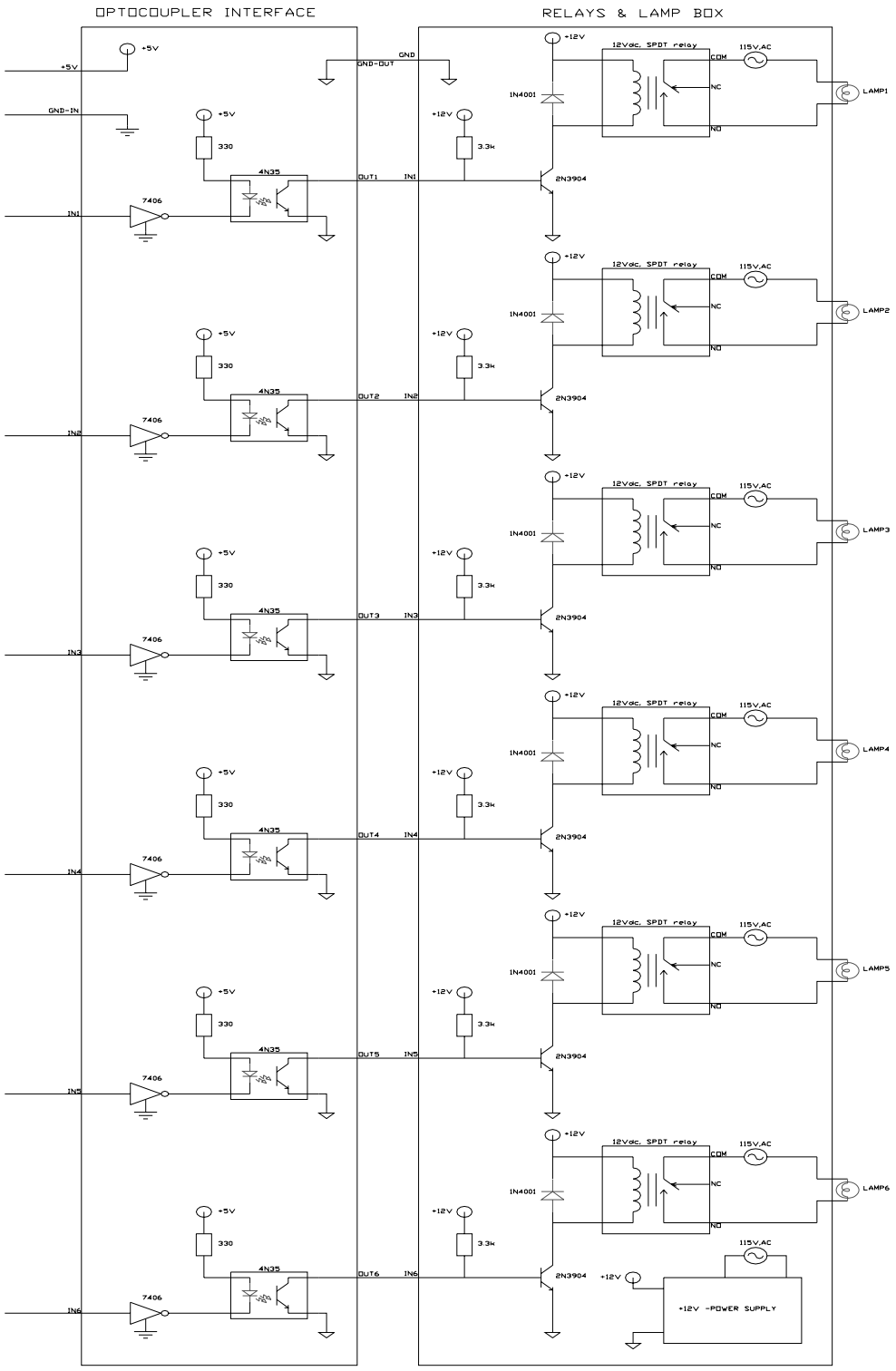
In the lab experiment you will utilize the general-purpose I/O ports of the TMS320LF2407 EVM to turn on or off AC light bulbs using relays. The interface circuit is shown in Lab Figure 3.1. The circuit is built of two modules i.e. the ***optocoupler interface circuit board*** and the ***relays & lamps box***.

The ***optocoupler interface circuit board*** contains a hex inverter buffers (7406) and six optocouplers (4N35). The optocouplers are used to separate the power supplies of the microcontroller and the relays. This separation is necessary in order to avoid the large load current fluctuation in the relays circuit being coupled back to the logic power supply.

The ***relays & lamps box*** encloses the relays and lamps circuitry. The relays used here are 12 Vdc, 30 mA SPDT relays with contact ratings of 10A, 125Vac. The lamps are 25 watts, 115 Vac colored light bulbs. These lamps are connected to the NO (normally open) terminal of the relay switches. The box is powered from the 115Vac source, which is used to supply the lamps and a 12Vdc-power supply.

Note that the overall net effect of the interface circuit shown in Lab Figure 3.1 is an inverting logic. This means that a logic 0 (LOW) at the input of the optocoupler

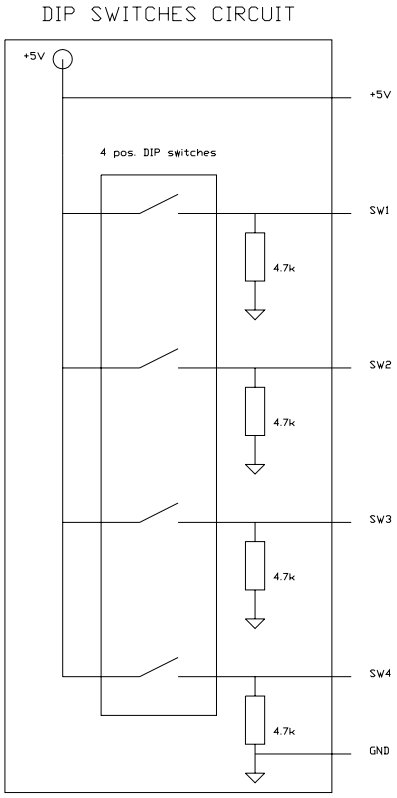
interface circuit will energize the relay, while a logic 1 (HIGH) cause the relay to be de-energized.



Lab Figure 3.1 *Relays interface circuit*

DIP switches interface circuit

Lab Figure 3.2 shows the circuit that will be used to interface a 4 position DIP switches to the input port of the DSP EVM.



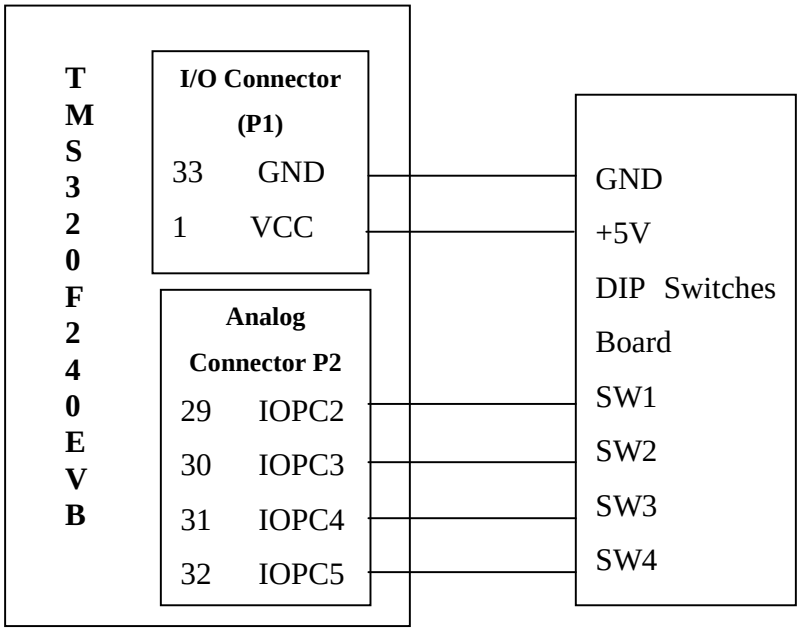
Lab Figure 3.2 DIP Switches Interface Circuit Board

Closing a switch causes the voltage at the corresponding terminal (SW1-SW4) to be pulled to logic HIGH, while opening a switch causes the terminal to be at logic LOW.

Procedure

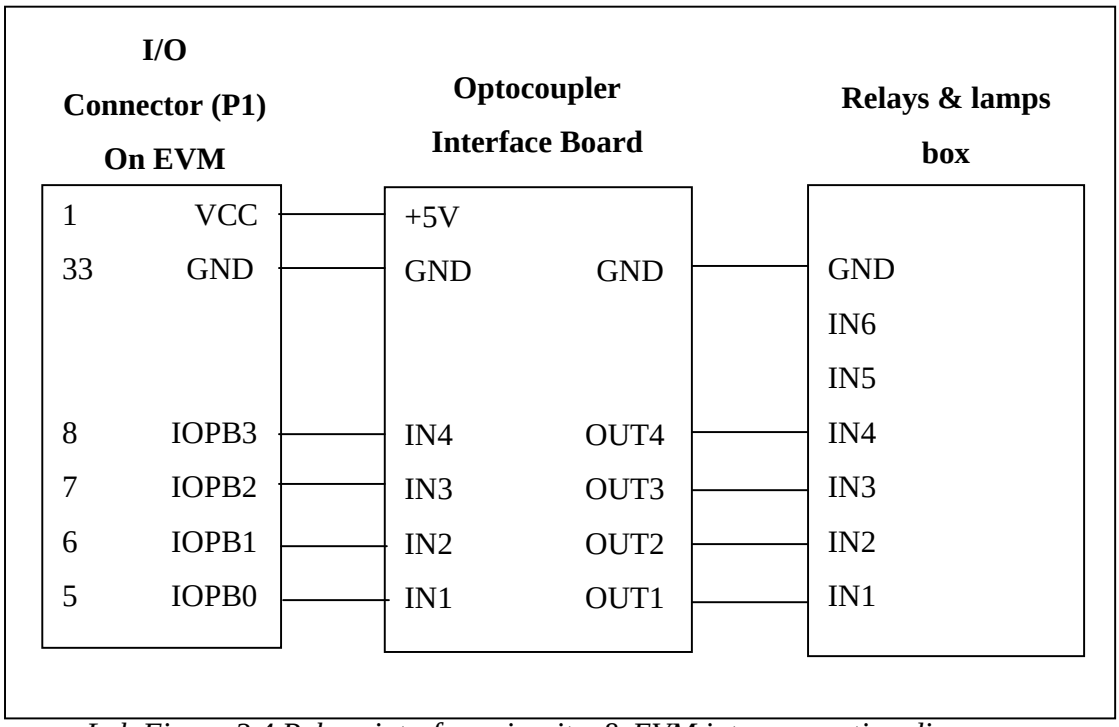
Setup

1. Setup the EVM and PC as discussed in LAB1.
2. Connect the DIP switches circuit board to the EVM terminal Blocks as shown in Lab Figure 3.3. This will connect the DIP switches to the Port C pins IOPC5, 4, 3, and 2 to SW4-1. Note: The EVM connector details are attached at the end of this lab session.



Lab Figure 3.3 DIP switches board & EVM interconnection diagram

3. Connect the relays interface circuit (optocoupler circuit board and relays & lamps box) as shown in Lab Figure 3.4. These interconnections cause the relays to be interfaced to the port B pins IOPB5-0.



Lab Figure 3.4 Relays interface circuits & EVM interconnection diagram

4. Show all the interconnections you made in step 1, 2 and 3 to the TA. If the TA agrees, you can proceed to the next step.
5. Turn on the PC and wait until the MS-Windows has started.
6. Power on the EVM power supply and run the EVM Testing program in PC to check the proper operation of the EVM and its connection to PC. If the test fails, turn off the EVM power supply, quit MS-Windows, and turn off the PC. Check the cable connections from the EVM to the PC and repeat step 5 and 6. If the problem persists, report to the lab TA. If the test succeeds, you are ready to use the EVM and proceed to the next procedure.
7. Power on the relays and lamps box by connecting its power cord to the AC outlet. If no problem exists, you should see all the lamps are on. Otherwise, check all the connections you have made. If the problem persists, report to the TA.

Laboratory Assignments

1. Write a program to that reads data from the DIP switches, inverts the data and outputs it to the lamps i.e. if SW1=0, LAMP1=OFF etc. Connect the DIP switches to IOPA3-0. Connect lamps 1-4 to IOPB0-3.

2. Write a program that will read the status of the switches and turn on the lamps as per the following logic.

Inputs		Outputs					
SW2	SW1	LAMP6	LAMP5	LAMP4	LAMP3	LAMP 2	LAMP1
0	0	OFF	OFF	OFF	OFF	ON	ON
0	1	OFF	OFF	OFF	ON	ON	OFF
1	0	OFF	ON	ON	OFF	OFF	OFF
1	1	ON	ON	OFF	OFF	OFF	OFF

3. Write a program that turns on the six lamps **one by one** according to the following sequence:

   1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 2, 3, 4,5, 6, 5, 4, 3, 2, 1, ...

   which repeats 'forever'. Use software delay to keep each lamp on for approximately two seconds.

4. Write a program that changes the pattern of the lamps every about five seconds, from pattern #1 until pattern #4, then repeats from pattern #1 and continues forever. The patterns are:

Pattern	LAMP6	LAMP5	LAMP4	LAMP3	LAMP2	LAMP1
#1	ON	ON	OFF	ON	OFF	ON
#2	OFF	ON	OFF	OFF	ON	ON
#3	ON	OFF	ON	OFF	OFF	OFF
#4	ON	ON	OFF	OFF	ON	ON

Table 2: P1 I/O

Pin #	Signal	Pin #	Signal
1	VCC, +5 Volts	2	VCC, +5 Volts
3	PWM1/IOPA6	4	PWM2/IOPA7
5	PWM3/IOPB0	6	PWM4/IOPB1
7	PWM5/IOPB2	8	PWM6/IOPB3
9	PWM7/IOPE1	10	PWM8/IOPE2
11	PWM9/IOPE3	12	T1PWM/T1CMP/IOPB4
13	T2PWM/T2CMP/IOPB5	14	T3PWM/T3CMP/IOPF2
15	* TDIRA/IOPB6	16	* TCLKINA/IOPB7
17	GND	18	GND
19	BOOTEN-/XF	20	* BIO/IOPC1
21	* CAP1/QEP1/IOPA3	22	* CAP2/QEP2/IOPA4
23	* CAP3/IOPA5	24	* CAP4/QEP3/IOPE7
25	RESERVED	26	* PDPINTA-
27	SCITXD/IOPA0	28	* SCIRXD/IOPA1
29	* SPISIMO/IOPC2	30	* SPISOMI/IOPC3
31	* SPICLK/IOPC4	32	* SPISTE/IOPC5
33	GND	34	GND

* Signal is interfaced through a quick switch to allow 5 volt tolerant inputs.