C2000 *Teaching Materials*

GETTING STARTED WITH THE TMS320F24x PROCESSOR

Tutorial 5: Using Input and Output Ports

New Instructions Introduced

```
AND
OR
```

Introduction

Input and output ports form an important interface between a processing device such as the TMS320F24x and the outside world. Computer users will be familiar with both serial (mouse and keyboard) ports and parallel (printer) ports.

The TMS320F24x has both serial and parallel ports. In this tutorial we shall use the parallel ports.

Logical Operations

In order to configure the parallel ports of the TMS320F24x, we need to modify specific bits of data words. For these we use logical operators.

Readers may already be familiar with the C language operators | (logical OR) and & (logical AND). For both of these there is an equivalent TMS320F24x assembly language operator.

Logical AND

The first logical operation we shall look at is the logical AND. The truth table is shown in Table 5-1:

Table 5-1. Logical AND

	1st Value	2nd Value	Result	
	0	0	0	
	0	1	0	
	1	0	0	
	1	1	1	

In order for the Result to be 1, both the 1st Value AND the 2nd Value must be 1.

As an example, let us perform the logical AND of the values 695Ch and 00FFh. We apply the truth table for the logical AND to each bit of the two values, as shown in Table 5-2:

Table 5-2. Logical AND

	Decimal	Hexadecimal	Binary
1st Value	26972	695Ch	0110 1001 0101 1100
2nd Value	00255	00FFh	0000 0000 1111 1111
Result	00092	005Ch	0000 0000 0101 1100

To use the analogy of C code, if the first value is stored in `variable`, we could write the operation shown in Table 5-2 as:

Example 5-1.

`unsigned int variable = 0x695C;`	
`variable &= 0x00FF;`	`// AND with variable.`

Let us now implement the operation shown in Table 5-2 using TMS320F24x assembly language. Assuming that `variable` is stored at data memory address 255h:

Example 5-2.

	.setsect	".text", 8800h
start:	CLRC CNF	; CNF = 0. Map block B0 into ; data memory.
	LDP #4	; Data page 4. Gain access to ; data memory addresses 200h to ; 27Fh.
	SPLK #695Ch,55h	; Store the value 695Ch at data ; memory address 200h + 55h = ; 255h.
	LACC 55h	; Load the accumulator with the ; contents of data memory address ; 20h + 255h = 255h.
	AND #00FFh	; Perform a logical AND of the ; accumulator with 00FFh. The ; accumulator now contains the ; value 0000005Ch.
	SACL 55h	; Save the right-most 16 bits of ; the result at data memory ; address 200h + 55h = 255h. ; The accumulator still contains ; 0000005Ch.
	B *start*	; Go round again.

In this case, the instruction AND (Logical AND with accumulator) takes a single operand, which is an immediate value in the range 0 to 65535 decimal (0 to FFFFh). Note that the symbol # must precede the value 0 to FFFFh.

In Example 5-2 it can be seen that the instruction `AND` with the immediate value FFh leaves the low (right-most) byte unaffected. However, all the other bits of the accumulator are cleared to zero. This is a way to get rid of unwanted bits.

Logical OR

The truth table for the logical OR is shown in Table 5-3:

Table 5-3. Logical OR

	1st Value	2nd Value	Result	
	0	0	0	
	0	1	1	
	1	0	1	
	1	1	1	

If either the 1st Value or the 2nd Value is a 1, then the result will be 1. The above truth table is applied a total of 16 times, once to each bit of a data word.

We can perform a logical OR of 695Ch and 00FFh as shown in Table 3-4:

Table 5-4. Logical OR

	Decimal	Hexadecimal	Binary
1st Value	26972	695Ch	0110 1001 0101 1100
2nd Value	00255	00FFh	0000 0000 1111 1111
Result	27135	69FFh	0110 1001 1111 1111

Instead of clearing specific bits of the accumulator to 0, as is the case with the logical AND, the logical OR sets specific bits to 1. We can therefore use the instruction `OR` (logical OR) to set one or more bits of the accumulator.

We can implement Table 5-4 using the analogy of C code:

Example 5-3.

`unsigned int` *`variable`* `= 0x695C;`	

`variable` `|= 0xFF;`	`// Logical OR` *`variable`*
	`// with FFh.`

To carry out the operation in Example 5-3 using TMS320F24x assembly language and assuming that `variable` is stored at data memory address 233h we write:

Example 5-4.

	`.setsect`	`".text", 8800h`

`start``:`	`CLRC CNF`	`; CNF = 0. Map block B0 into`
		`; data memory.`

	`LDP #4h`	`; Data page 4. Gain access to`
		`; data memory addresses 200h to`
		`; 27Fh.`

	`SPLK #695Ch,33h`	`; Store the value 695C at data`
		`; memory address 200h + 33h =`
		`; 233h.`

	`LACC 33h`	`; Load the accumulator with the`
		`; contents of data memory`
		`; address 200h + 33h = 233h.`

	`OR #00FFh`	`; Perform logical OR of the`
		`; accumulator with FFh. The`
		`; accumulator now contains`
		`; 000069FFh.`

	`SACL 33h`	`; Save the result of the logical`
		`; OR at data memory address 200h`
		`; + 33h = 233h. The accumulator`
		`; still contains 000069FFh.`

	`B` *`start`*	`; Go round again.`

The immediate value used as the operand can lie in the range 0 to 65535 decimal (0h to FFFFh). The logical OR is used to set specific bits to 1, but leaves other bits unaffected.

Using Ports

Having familiarized ourselves with the logical AND and OR operators, we are now ready to configure the ports on the TMS320F24x.

The number of input / output ports varies between devices in the TMS320F24x family. For example, the TMS320F243 has four input /output ports referred to as Port A, Port B, Port C and Port D.

Some port pins have dual functionality. For example, two pins of Port A are used for the serial interface to the GO-DSP debugger. This means that on the TMS320F243 DSK, two pins of Port A are not available for general purpose.

To avoid inadvertently reconfiguring Port A, we shall work only on Port B, Port C and Port D.

Data and Direction Registers

Port B is a representative input / output port. It has eight physical pins that can be configured as inputs or outputs, on an individual basis. The individual pins are as referred to as IOPB0 to IOPB7 where IOPB means *input output port* B and are numbered 0 to 7.

Port B is configured by writing a value to the I/O Port B Data and Direction Register (abbreviated to PBDATDIR).

The I/O Port B Data and Direction Register (PBDATDIR) has dual functionality. The high byte (bits 15 to 8) controls the *direction* of the data transfer, that is, whether each pin is to be used as an input or an output. The low byte (bits 7 to 0) contains the *data* to be transferred.

Figure 5-1. Data and Direction Register

Direction Bits								Data Bits							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DIR 7	DIR 6	DIR 5	DIR 4	DIR 3	DIR 2	DIR 1	DIR 0	D7	D6	D5	D4	D3	D2	D1	D0

At power up, all the I/O Ports power up as *inputs*. Pins assigned as inputs can be safely connected to ground or the power supply. If all the I/O Ports powered up as outputs, then some pins could be trying to drive into a power rail, which could cause damage to the silicon of the device.

Configuring Port B as an Output Port

To make Port B an output port, we must carry out two operations. The first is to configure the *direction* bits of PBDATDIR and the second is to configure the *data* bits.

To set the *direction* bits of PBDATDIR to output we must write the value FFh to the high byte (bits 15 to 8). Whatever we write to the low byte (bits 7 to 0) will be the *data* that appears at the physical pins IOPB7 to IOPB0.

PBDATADIR is located at address 709Ah in data memory. To write the value 0x36 to IOPB we would write:

Example 5-5.

	`.setsect ".text",`	`8800h`

start:	`LDP #225`	`; Data page 225. Gain access to`
		`; data memory address 7080h to`
		`; 70FFh.`

	`SPLK #0FF36h, 1Ah`	`; Configure Port B at data memory`
		`; address 7080h + 1Ah = 709Ah as`
		`; an output port and write 36h to`
		`; it.`

	`B` *start*	`; Go round again.`

Configuring Port B as an Input Port

To configure Port B as an input port, we must again carry out two operations. The first is to configure the *direction* bits to input. To do this we write 00 to the high byte (bits 15 to 8) of the register PBDATDIR. The value written to the low byte (bits 7 to 0) has no effect.

Example 5-6.

	`.setsect ".text",`	`8800h`

start:	`LDP #225`	`; Data page 225. Gain access to`
		`; data memory address 7080h to`
		`; 70FFh.`

	`SPLK #0000h, 1Ah`	`; Write to register PBDATDIR.`
		`; Configure Port B as an input`
		`; port.`

loop:	`LACC 1Ah`	`; Read in Port B to accumulator.`
	`B` *loop*	`; Go round again.`

When we read PBDATDIR, the low byte (bits 7 to 0) contains the values at the port pins IOPB7 to IOPB0. The high byte (bits 15 to 8) will always contain zero.

A More General Routine for Writing to Port B

In the course of a program, we might configure the direction of a particular port once only. However, we are likely to write values to the port on a regular basis to interface to the outside world.

There is a practical problem. Suppose we are copying a value from the accumulator to Port B, which has previously been configured as an output port. When we write to PBDATDIR, the high byte (bits 15 to 8) must always contain FFh, otherwise the direction of the port would be inadvertently changed to input.

We need a way to be able to write data values to a port, but without affecting the *direction* bits. For this we use the logical operators AND and OR.

Let us assume that the value stored at data memory address 303h is to be transferred to the output pins of Port B. We copy this value from data memory address 303h to the accumulator, and then from the accumulator to PBDATDIR.

Example 5-7.

	.setsect ".text",	8800h
start:	LDP #225	; Data page 225. Gain access to ; data memory address 7080h to ; 70FFh.
	SPLK #5555h, 1Ah	; For test purposes. Write a ; known value to PBDATDIR.
loop:	LACC 1Ah	; Read present value of PBDATDIR.
	AND #0FF00h	; Clear the data bits to zero. ; Leave the direction bits ; unchanged.
	LDP #6	; Data page 6. Gain access to ; data memory addresses 300h to ; 37Fh.
	OR 3h	; Perform a logical OR of the ; contents of data memory ; address 300h + 3h = 303h with ; the accumulator.
	LDP #225	; Data page 225. Gain access to ; data memory address 7080h to ; 70FFh.
	SACL 1Ah	; Write new value to PBDATDIR.
	B *loop*	; Go round again.

Here we have loaded PBDATDIR into the accumulator, performed a logical OR with the new output value and then sent the value to the output port PORTB. In this way the high byte of PBDATDIR, which controls the direction of data transfer remains unchanged.

There is an assumption in Example 5-7; that the bits 15-8 of data memory address 303h are zero. When this is the case, they have no effect on the direction bits of PBDATDIR.

Hardware Testing of Port B

One way to test that Port B is operating correctly as an output port is to write a series of different values to it and then measure the physical levels on the port pins using an oscilloscope or a frequency meter.

One way to do this is shown in Example 5-8.

Example 5-8.

```

unsigned int x;

while (1)
{
   x++;
   PBDATDIR = (x | 0xFF00);
}
```

The code in Example 5-8 continually writes a different value to PBDATDIR. This means that one or more of the pins of Port B changes each time a value is written to PBDATDIR.

Let us now implement Example 5-8 using assembly language. We shall assume that the variable *x* is stored at data memory address 300h.

Example 5-9.

	.setsect ".text",	8800h
start:	CLRC CNF	; CNF = 0. Map block B0 into ; data memory for general ; purpose use.
	LDP #6	; Data page 6. Gain access to ; data memory addresses 300h to ; 37Fh.
	LACC 0h	; Read value from 300h.
	ADD #1	; Increment value.
	SACL 0h	; Write back incremented value to ; 300h. The accumulator still ; contains the incremented value.
	LDP #225	; Data page 225. Gain access to ; data memory address 7080h to ; 70FFh.
	OR #0FF00h	; Set all direction bits to ; outputs.
	SACL 1Ah	; Write new value to PBDATDIR.
	B *start*	; Go round again.

The code in Example 5-9 takes the value at data memory address 300h then copies it to the accumulator where it is incremented. The direction bits (bits 15 to 8) are set to FFh every time a value is written to PBDATDIR.

Mixed Input and Output Ports

The easiest way to allocate input and output pins is on a port-by-port basis. For example, we might use Port A and Port B for inputs and Port C and Port D as outputs.

However, in a real-life situation, we might need 9 inputs and 11 outputs. This means we have to set up certain ports to have both input and output pins.

Suppose we wish to configure Port C to have four input pins and four output pins. We shall allocate the four input pins to IOPC3, IOPC2, IOPC1 and IOPC0. Similarly, we shall allocate the four output pins to IOPC7, IOPC6, IOPC5 and IOPC4. This means that PCDATDIR must be configured as follows:

Figure 5-2. Configuration of PCDATADIR

Direction Bits								Data Bits							
Output Control				Input Control				Output Data				Input Data			
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	1	1	1	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0

When reading PCDATDIR, the *direction* bits 15 to 8 will always read back as 1111000 = F0h. When writing to PCDATDIR, the *direction* bits (bits 15 to 8) must always be F0h.

Reading a Mixed Input / Output Port

To read the input part of Port C, we take the value in PCDATDIR (at data memory address 709Ch) then clear all the bits to zero other than D3, D2, D1 and D0. We then make a copy of the variable in data memory, which we can use later in our program.

Example 5-10.

	.setsect ".text",	8800h
start:	LDP #225	; Data page 225. Gain access to ; data memory addresses 7080h to ; 70FFh.
	SPLK #0F0F0h, 1Ch	; Test purposes. Write at known ; value to PCDATDIR at data ; memory address 7080h + 1Ch = ; 709Ch.
loop:	LACC 1Ch	; Load contents of PCDATDIR at ; data memory address 7080h + 1Ch ; = 709Ch into the accumulator.
	AND #000Fh	; Clear all bits except bit 3 to ; bit 0. The accumulator now ; contains the input data only.
	LDP #6	; Data page 6. Gain access to ; data memory addresses 300h to ; 37Fh.
	SACL 20h	; Save at 300h + 20h = 320h for ; later use.
	LDP #225	; Data page 255. Page used by ; PCDATDIR.
	B *loop*	; Go round again.

Without the instruction `AND #000Fh`, bits 15 to 12 would always be read back as 1111b. For example, if the input pins were 0101b, then we would incorrectly read 0F05h into the accumulator.

Writing a Value to a Mixed Input / Output Port

Suppose we wish to write a 4-bit value in the range 0 to 15 to IOPC7, IOPC6, IOPC5, IOPC4 as per Figure 5-2.

To carry out this operation is more complex than previous operations. When we write to PCDATDIR, we must be careful to keep the *direction* bits (bits 15 to 8) as F0h, otherwise we could inadvertently change inputs to outputs. Second, the *data* value, which lies in the range 0 to 15, must be shifted four places to the left to align it with bits 7 to 4 of PCDATDIR.

In C code we could write:

Example 5-11.

unsigned int *y* = 0x05;		
PCDATADIR = ((*y* << 4) & 0x00F0)	0xF000h);		

In Example 5-11, *y* is first shifted four places to the left. A logical AND is then performed to make all the other bits zero. Then a logical OR is performed to add the four *direction* bits.

Assuming that *y*, which is to be written to Port C is at data memory address 301h:

Example 5-12.

	.setsect ".text",	8800h
start:	LDP #225	; Data page 225. Gain access to ; data memory addresses 7080h to ; 70FFh.
	SPLK #0F0F0h, 1Ch	; Configure PCDATDIR as a mixed ; input and output port.
loop:	LDP #6	; Data page 6. Gain access to ; data memory addresses 300h to ; 37Fh.
	SPLK #05h, 01h	; Test purposes. Make variable *y* ; at address 300h + 1h = 301h a ; known value.
	LACC 01h, 4	; Load accumulator with value of ; *y* shifted 4 bits to the left.
	AND #00F0h	; Clear all data bits except 7 to ; 4.
	OR #0F000h	; Set direction bits to output ; for bits 7 to 4.
	LDP #225	; Data page 225. Gain access to ; data memory addresses 7080h to ; 70FFh.
	SACL 1Ch	; Send value to PCDATDIR at data ; memory address 7080h + 1Ch = ; 709Ch.
	B *loop*	; Go round again.

Note use of the shift used with the instruction LACC (load accumulator). This allows us to shift the value specified by the first operand then load it into the accumulator.

Strictly speaking, in Example 5-12, the instruction `AND #00F0h` is redundant. This is provided that the variable x lies in the range 0 to 15 (0 to 000Fh).

However, what happens if the value of x used is the out-of-range value 00FFh? When shifted four places to the left it will become 0FF0h. The effect upon the direction bits is shown in Figure 5-3.

Figure 5-3. Configuration of PCDATADIR with Incorrect Data

Direction								**Data**							
Output Control				Input Control				Output Data				Input Data			
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	1	1	1	1	1	1	1	1	1	1	1	D3	D2	D1	D0

When the value FFh is loaded with shift then each of bits 11 to 8 will be set to 1. This will turn what is intended to be an *input* port into an *output* port.

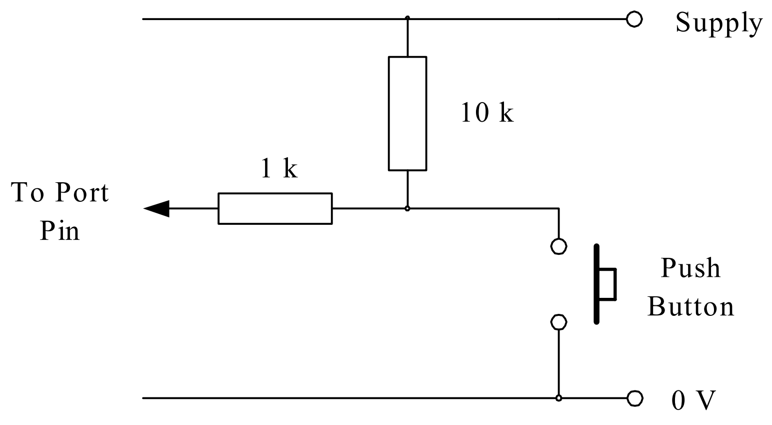
TMS320F243 DSK EXPERIMENTS

Equipment Required

TMS320F243 DSK
Frequency meter or oscilloscope
Analog voltmeter or multi-meter with range 0-5V D.C. or 0-10V D.C.
Circuit as shown in Figure 5-3 to short input pins to ground.

A suggested configuration for a port pin is shown in Figure 5-3. This limits the input and output current. For the TM320F243 the supply can be in the range 3.3V to 5.0V. However, for 3.3V devices such as the TMS320LF2407, the supply must be 3.3V; higher voltages such as 5V will cause damage.

Figure 5-3. Port Configuration



Experiment 5-1.

Objective: To Generate waveforms at Input/Output port B.

Enter the code in Example 5-9 into a text editor, save it, assemble it and run it on the TMS320F243 DSK. Measure the output frequency on pins IOPB0 to IOPB7 using a frequency meter or an oscilloscope. Use connector P2 pins 19 or 20 as the ground reference.

The values measured should be:

TMS320F243 DSK	Port Pin	Frequency
Connector P2, Pin 11	IOPB0	102.4 kHz
Connector P2, Pin 12	IOPB1	51.2 kHz
Connector P2, Pin 13	IOPB2	25.6 kHz
Connector P2, Pin 14	IOPB3	12.8 kHz
Connector P2, Pin 15	IOPB4	6.4 kHz

Connector P2, Pin 16	IOPB5	3.2 kHz
Connector P2, Pin 17	IOPB6	1.6 kHz
Connector P2, Pin 18	IOPB7	0.8 kHz

Experiment 5-2.

Objective: To test the inputs of a mixed input/output port.

Enter the code in Example 5-10 into a text editor, save it, assemble it and load the code into the TMS320F243 DSK. Step through the code, making a note of the values in the accumulator and data memory address 420h.

Short out IOPC0 (Connector P2, pin 21) to ground (Connector P2, pin 19 or 20). Step through the code and make a note of the values in the accumulator and data memory address 420h. Bit 0 of both of these should be 0.

Repeat for:

IOPC1 (Connector P2, pin 22), which should make bit 1 of the accumulator 0.
IOPC2 (Connector P2, pin 23), which should make bit 2 of the accumulator 0.
IOPC3 (Connector P2, pin 24), which should make bit 3 of the accumulator 0.

Experiment 5-3.

Objective: To Use Input / Output Port C as an Analog-to-Digital Converter

Enter the code in Example 5-12 into a text editor, save it, assemble it and run it on the TMS320F243DSK. Using an analog voltmeter, measure the values on each port of pin IOPC0 to IOPC3. The values on the output Port C should be:

DSK	Port Pin	Value
Connector P2, Pin 21	IOPC0	4.9 V (logic ‘1’)
Connector P2, Pin 22	IOPC1	0.1V (logic ‘0’)
Connector P2, Pin 23	IOPC2	4.9V (logic ‘1’)
Connector P2, Pin 24	IOPC3	0.1 V (logic ‘0’)

What effect is there upon the XF LED?

Experiment 5-4.

Objective: To Read / Write a Data Direction Register

Enter the code in Example 5-6 into a text editor. Save it, assemble it and run it on the TMS320F243 DSK. This writes the 0000h to PBDATDIR. What value is read back? Why is this?

Design Problem 5-1.

Write code to set up Port C as an input port and Port D as an output port. The program should execute the loop repeatedly, copying the value from the input of Port C to the output of Port D. The data and direction register for Port D is PDDATDIR and is situated at data memory address 709Eh.

Self-Test Questions**.......................................** *Click Here To View* Answers!

1.	What is meant by the term *port* when applied to TMS320F24x hardware?

2.	After the following two instructions have been executed, what will be the contents of the accumulator?
	`LACC #1234h`
	`AND #F00Fh`

3.	After the following two instructions have been executed, what will be the contents of the accumulator?
	`LACC #5678h`
	`OR #0FF0h`

4.	Write 0000h to PBDATDIR then read it back. Why has the value changed?
5.	If we configure Port A of the TMS320F243 DSK as an output port, the debugger stops working. Why is this?
6.	The abbreviation IOPB means: a) Inner and outer pointer block b) Inclusive OR with PB c) Input Output Port B d) Input Output pointer base
7.	What is meant by *direction* when applied to a port?
8.	What is meant by *data* when applied to a port?
9.	Why do input / output ports power up as inputs?
10.	What happens if we inadvertently change the direction bits of PBDATDIR?
11.	In a design, why might we need to have mixed input and output pins on a particular Port?
12.	After the instruction `LACC #01h, 4` what value does the accumulator contain? a) 01h b) 04h c) 08h d) 10h e) 14h

References

TMS320F/C24x DSP Controllers. CPU and Instruction Set. Reference Number: SPRU160.

TMS320F/C240 DSP Controllers. Peripheral Library and Specific Devices. Reference Number: SPRU161

TMS320F243, TMS320F241 DSP Controllers. Reference Number SPRS064.

CLICK TO VIEW
Tutorials
1 2 3 4 5 6 7 8 9 10

Click here to view.......... Route Map