I/O Ports 6

CHAPTER 6
I/O PORTS

I/O ports provide a mechanism to transfer information between the device and the surrounding system circuitry. They can read system status, monitor system operation, output device status, configure system options, generate control signals, provide serial communication, and so on. Their usefulness in an application is limited only by the number of I/O pins available and the imagination of the engineer.

6.1. FUNCTIONAL OVERVIEW

The 8XC196KC/KD has five 8-bit I/O ports. Each I/O port has pins that operate as either input-only (input), output-only (output), quasi-bidirectional (QBD), open-drain bidirectional, or a combination of the four. Almost all of the I/O port pins perform an alternate function. For example, one of the I/O port pins can become a PWM output, while another can be used as an analog input.

Every I/O port pin has a default operation (e.g., output). However, the operation of the pin may change when its alternate function is selected. One such change may be that a quasi-bidirectional pin becomes an output-only pin when the alternate function is selected. In any case, the pin has a basic set of operating characteristics associated with a given configuration (e.g., output). Table 6-1 lists the five I/O ports and their default and alternate functions.

The fifth column of Table 6-1 indicates which SFR determines whether the pin operates as a port or its alternate function. Some port pins operate both as a port and the alternate function at the same time (e.g., Port 0 and the analog inputs). In this case, the fifth column of Table 6-1 is left empty.

In addition to the standard I/O ports, the High-Speed Input module and the High-Speed Output module provide extra I/O functions if the timer-related features of these pins are not needed.

Table 6-1. Port Pin Functions

Port Pin(s)	Port Type	Alternate Function	Alternate Type	Control SFR
P0	Input	ACH0–ACH7	Input	
P1.0, P1.1, P1.2	Quasi-Bidirectional	None		
P1.3, P1.4	Quasi-Bidirectional	PWM1, PWM2	Output	IOC3
P1.5, P1.6	Quasi-Bidirectional	BREQ#, HLDA#	Output	WSR
P1.7	Quasi-Bidirectional	HOLD	Input	WSR
P2.0	Output	TXD	Output	IOC1
P2.1	Input	RXD	Input or Output	SPCON
P2.2	Input	EXTINT	Input	IOC1
P2.3	Input	T2CLK	Input	IOC0
P2.4	Input	T2RST	Input	IOC0
P2.5	Output	PWM0	Output	IOC1
P2.6	Quasi-Bidirectional	T2UP-DN	Input	IOC2
P2.7	Quasi-Bidirectional	T2CAPTURE	Quasi-Bidirectional	
P3, P4	Open-Drain Bidirectional	AD0–AD15	Bidirectional	

6.1.1. Input Port Pins

Any port pin defined as an input can only be read. Any write operation to the pin is ignored (or is not possible). The major circuits of an input pin consist of an input sample latch and a read buffer (see Figure 6-1).

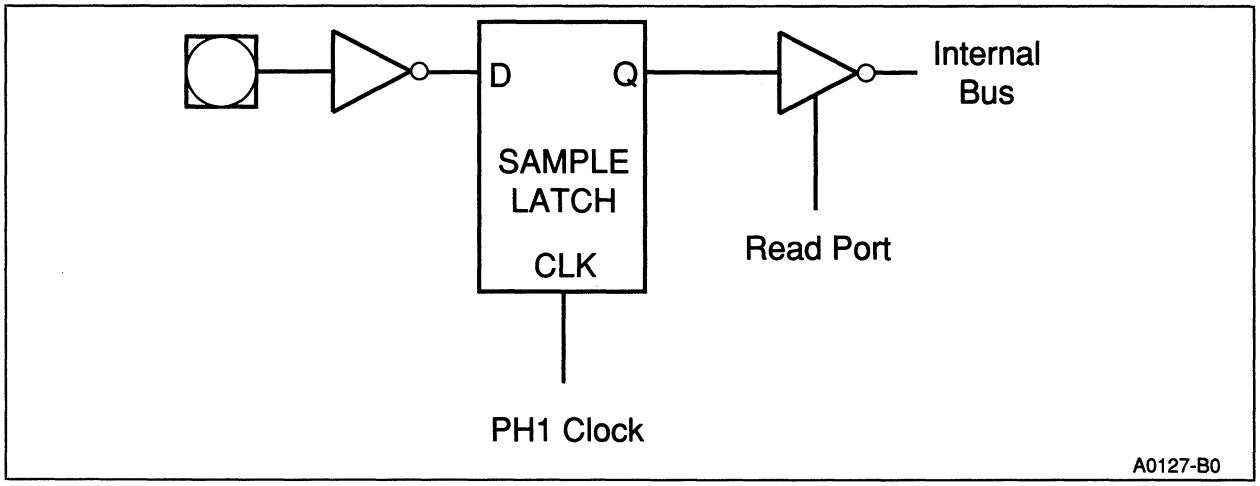

Figure 6-1. Input Port Pin

The port pin is sampled one state time before the read buffer is enabled. Sampling occurs during Phase 1 (while CLKOUT is low) and resolves the value (high or low) of the pin before it is presented to the internal bus. If the pin value changes during the sample time, the new value may or may not be recorded during the read.

Input port pins have no output drivers. The input leakage of these pins is very low, as is their capacitive loading.

6.1.2. Output Port Pins

An output pin consists of port latch and pin driver logic (see Figure 6-2).

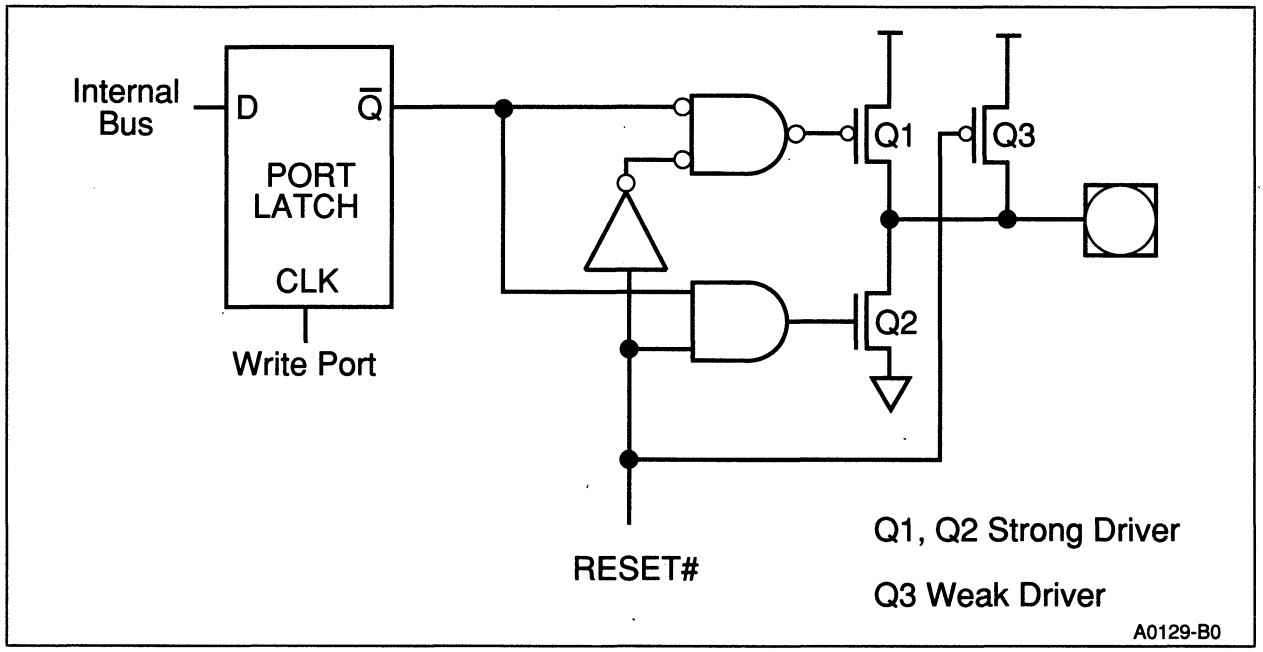

Figure 6-2. Output Port Pin

The new value of the output pin transitions during Phase 1 (while CLKOUT is low). The pin value remains stable until another write operation changes it (or a device reset occurs). Output port pins do not have sample and read buffer circuitry. When an output pin is read, its value is undefined and should be masked.

Since there is no mechanism to read the value of the port pin, it is possible only to set or clear an output pin. If it is necessary to complement the value of the pin, an image of the pin must be kept in memory. This memory image is manipulated and then used to set or clear the pin.

6.1.3. Quasi-Bidirectional Port Pins

Quasi-bidirectional pins can be used as input and/or output pins (without the need for direction control logic). These pins output a strong low value or a weak high value. The weak high value can be externally overridden, providing an input function.

Figure 6-3 illustrates the major circuits that make up a quasi-bidirectional port pin. Like an input pin, it has a sample latch and read buffer. Like an output pin, it has a data latch. It is the output driver structure that makes a quasi-bidirectional pin unique.

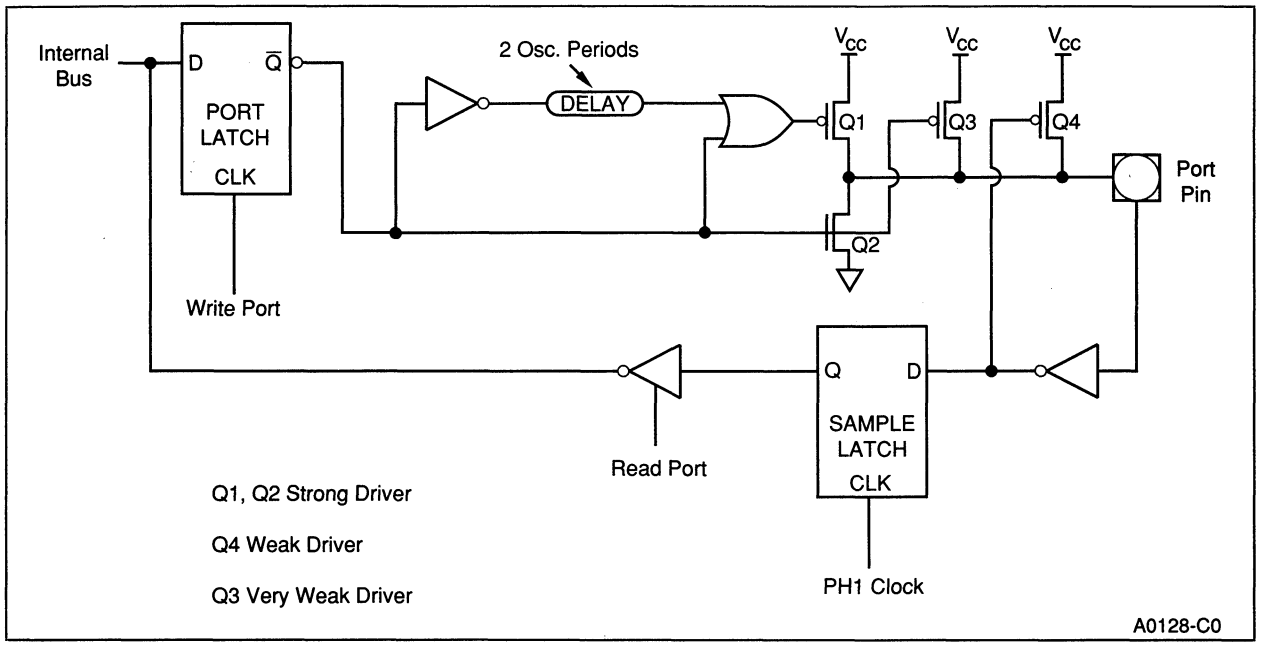

Figure 6-3. Quasi-Bidirectional Port Pins

The output of the pin can have only two static values: strongly driven low or weakly driven high. (If the output is transitioning from low to high, it is strongly driven high for a brief time; this is discussed later.) While the pin is driving low, it is not possible to overcome the driver (i.e., it cannot be driven high externally without exceeding device specifications).

Writing a one to the port pin disables the strong low driver (Q2) and enables a very weak high driver (Q3). To get the pin to transition high quickly, a strong high driver (Q1) is enabled for one state time and then disabled (leaving only Q3 active). Unless the port pin is heavily loaded, Q3 is capable of keeping the pin at a logic one state.

Because a quasi-bidirectional port pin has read circuitry, it is possible to read the output value of the pin directly (unlike the output port pins). However, since the port pin can be externally overridden with a zero, reading the port pin could falsely indicate that it was written as a zero.

The ability to overdrive the weak output driver is what gives the quasi-bidirectional port pin its input capability. To reduce the amount of current that flows when the pin is externally pulled low, the weak output driver (Q4) is turned off when a valid zero (low) is detected (approximately 2 volts).

Figure 6-4 illustrates the transfer characteristic for the port pin when Q3 and Q4 are enabled. As the pin voltage is lowered from V_{CC}, the current sourced by the pin increases until the threshold current, I_{TL}, is reached. At this point, Q4 is turned off and only a very weak pull-up (Q3) is present (I_{IL} in the data sheet).

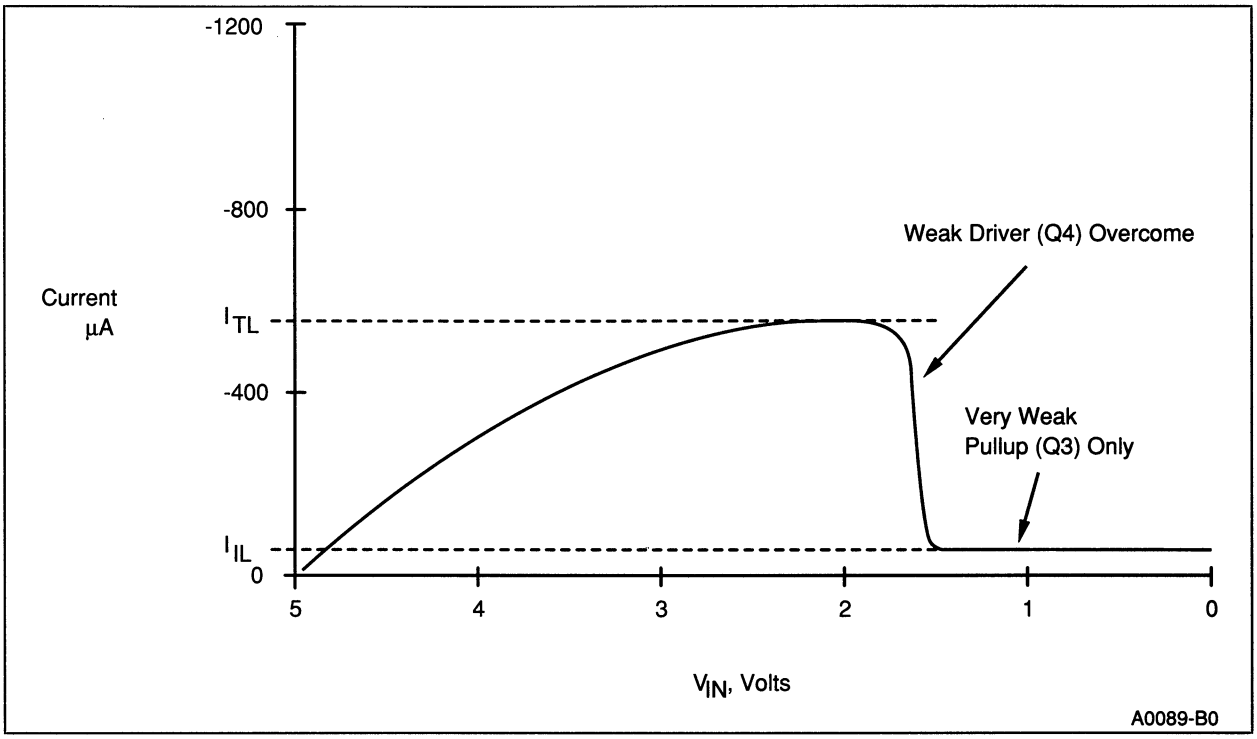

Figure 6-4. Quasi-Bidirectional Input Transfer Characteristic

The output of the quasi-bidirectional port pin also contains a pull-up driver (Q3) that is enabled whenever a one has been written to the output latch. This pull-up driver supplies a very small amount of current that causes a floating pin to return to a high (one) value. The current is specified as I_{IL} in the data sheet.

If some pins of a port are to be used as inputs and some are to be used as outputs, the programmer should be careful when writing to the port. Specifically, care should be exercised when using XOR opcodes or any opcode that is a read-modify-write instruction. It is possible for a quasi-bidirectional pin to be written as a one but read back as a zero, if an external device (i.e., a transistor base) is pulling the pin below V_{IH}.

The timing characteristics of the quasi-bidirectional port pins are similar to those of the input and output port pins. When the port pin is read, its value is sampled during Phase 1 (CLKOUT low) one state before the read buffers are enabled. A write to the output latch changes the pin value during Phase 1.

6.1.4. Open-Drain Bidirectional Port Pins

Open-drain bidirectional port pins operate like quasi-bidirectional port pins, except that there are no strong or weak high drivers. Figure 6-5 shows the basic circuits of the open-drain bidirectional port pin.

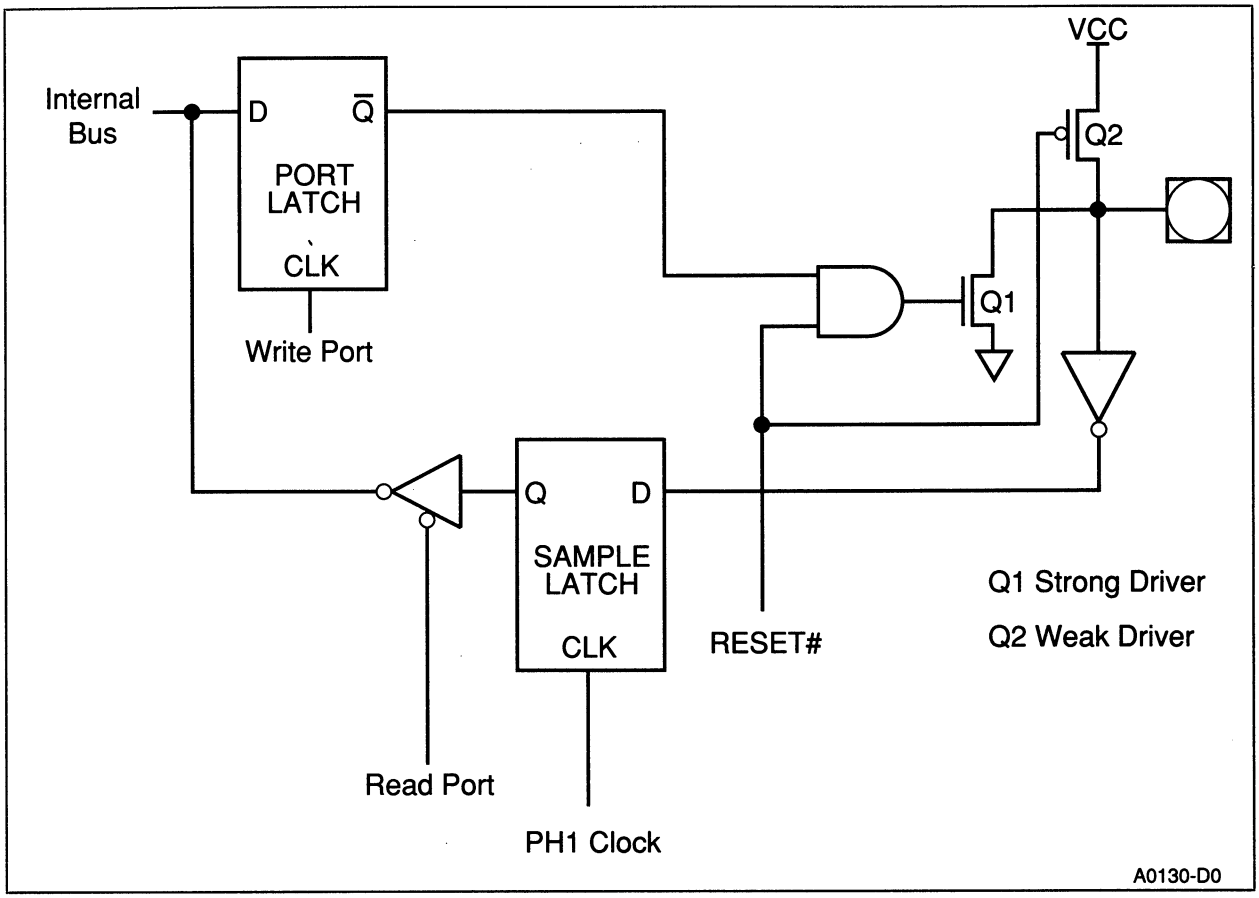

Figure 6-5. Open-Drain Bidirectional Port Pin

Writing a zero to the port latch enables the strong low driver (Q1). While Q1 is enabled, it is not possible to overdrive the zero externally without exceeding device specifications. A one written to the port latch disables Q1 and leaves the port pin floating. Externally, the port pin can be driven high with a pull-up resistor or logic gate.

6.2. PROGRAMMING THE I/O PORTS

Each of the five I/O ports has an associated Special Function Register (SFR) to read or write the port pins. Some of the I/O ports have an additional SFR that reconfigures the port pin to support an alternate function. Not all port SFRs are both readable and writeable (e.g., the Port 0 SFR is only readable). Also, not all SFRs exist in the same horizontal window (e.g., the SFR control registers are written in HWindow 0 but read back in HWindow 15).

IOPORT0, IOPORT1, and IOPORT2 are the SFRs used to read Port 0, read/write Port 1, and read/write Port 2, respectively. These registers have a similar format, shown in Figure 6-6. A complete description of each register can be found in Appendix C. Table 6-2 summarizes what happens when an I/O port is read or written and what conditions may exist to cause an unexpected result.

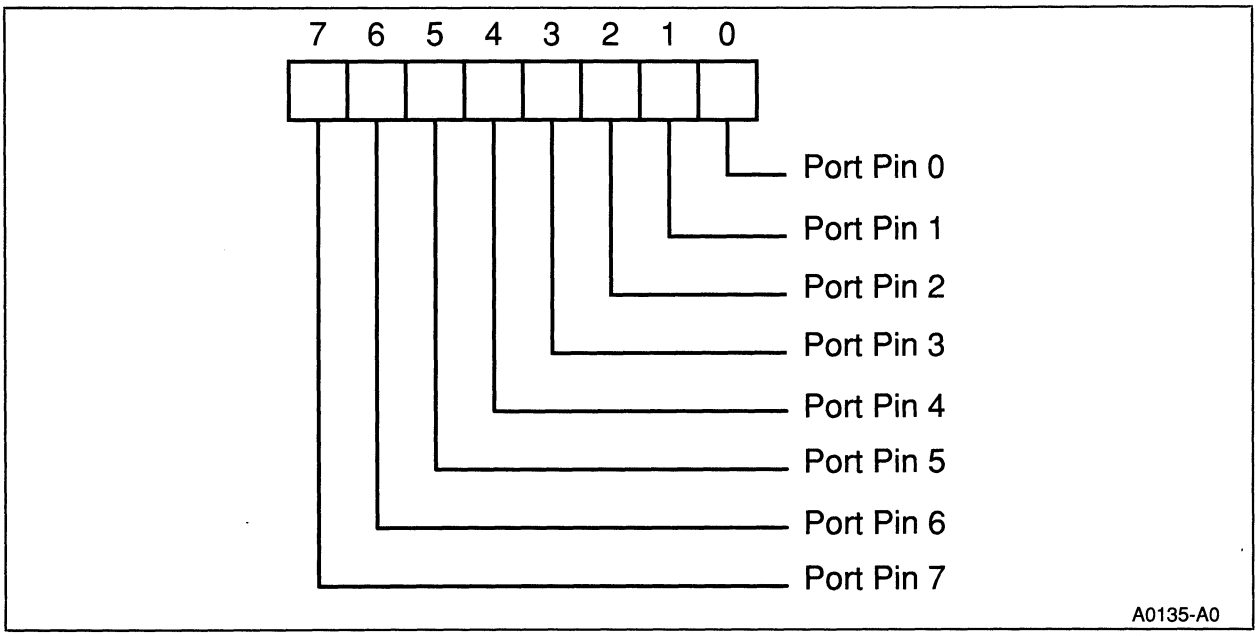

Figure 6-6. I/O Port SFR

Table 6-2. Summary of Port Read/Write Operation

Port	Accessed By	Read Operation	Write Operation
0	IOPORT0	Reads the current value of the port pins. The port should not be read while an A/D conversion is in progress.	Not possible. Writing to IOPORT0 does not affect Port 0, but does affect the baud rate controller.
1	IOPORT1	Reads the current value of the port pins. A port pin may have been written a value of one but return a value of zero because it is being overridden externally.	Writes change the state of the port pin latch. Changing the port pin latch has no effect on a port pin if its alternate function has been selected.
2	IOPORT2	Reads the current value of the port pins, except those pins that are output only. Output-only pins return an unspecified value and should be masked by software.	Writes change the state of the port pin latch, except those pins that are input only. Input-only port pins have no pin latches, so the value written is ignored.
3,4	1FFEH	Reads the current value of the port pins. A port pin may have been written a value of one but return a value of zero because it is being overridden externally. Ports 3 and 4 are always accessed together as a word read operation.	Changes the state of the port pin latch. Any external access of the bus controller overrides the value of the port pin latch to perform the bus operation. Ports 3 and 4 are always accessed together as a word write operation.

After a valid device reset, all of the I/O ports except Port 3 and Port 4 behave as port functions. When EA# is low during the rising edge of RESET#, Port 3 and Port 4 automatically support the system address/data bus and cannot be used as I/O ports unless they are reconstructed externally. All or portions of Port 0, Port 1, and Port 2 are optionally configured to enable alternate pin functions.

The SFR used to configure the ports depends on which I/O port pin is being reconfigured and is not discussed in this chapter. Instead, the chapter describing the operation of the peripheral that makes use of the I/O port pin contains the information needed to reconfigure the pin's function. Refer to Table 6-1 on page 6-2 to determine which SFR controls the alternate function of an I/O port pin (or pins).

6.2.1. Port Input

Reading an I/O port first samples the value of the port pin before it is stored internally. Because each port read samples a new value, it not possible to "latch" the port value simply by reading it once. If the port value must be operated on multiple times, a temporary memory location must be used to hold the initial value read. For example, the following operation reads the current value of Port 1 and stores the value in a memory location labeled P1TEMP:

```
LDB P1TEMP, ioport1            ;Read P1 and store in P1TEMP
```

Certain pins of Port 2 are configured as outputs only and do not contain any input circuitry. After the port is read, any SFR bit location containing an output-only pin should be masked before the input value is operated on.

To use a quasi-bidirectional port pin as an input, it is first necessary to ensure that the strong low driver is turned off. Writing a one to the port pin disables this strong driver. For example, the code below configures the lower half of Port 1 to be used as inputs:

```
LDB IOPORT1,#00001111B         ;Write 1s to Port 1 input pins
```

6.2.1.1. PORT 0 CONSIDERATIONS

Port 0 pins are special in that they may individually be used as digital inputs and analog inputs at the same time. A Port 0 pin being used as a digital input acts as a high-impedance input. However, Port 0 pins being used as analog inputs are required to provide current for a short time to charge the internal sample capacitor. This means that the input characteristics of a pin momentarily change if a conversion is being done on that pin. In either case, if Port 0 is to be used as analog or digital I/O, it is necessary to provide power to this port through the V_{REF} pin and the ANGND pin.

The A/D converter is sensitive to noise, and one way to induce noise into the converter is to read Port 0. It is strongly recommended that no port read operation occur while an A/D conversion is in progress. Port 0 has been designed such that the sample latch is not clocked unless a port read is being executed.

6.2.2. Port Output

Writing to an I/O port sets or clears its associated port latch. The output of the port latch is then used by the output driver. A write operation affects only output-only or bidirectional (quasi-bidirectional or open-drain) pins. A write to an input-only pin performs no operation (i.e., no latch is set or cleared).

There are many ways to write to an I/O port. The port can be written directly, using the following instructions:

```
LDB ioport1,#10000001B        ;Set bits 0 and 7, clear bits 1-6
                              ; or
LDB intrega,#81H              ;Value to be written
STB intrega,ioport1           ;Output value to port
```

Typically, a port is directly written to only for initialization or when all of the port pins need to change (or can change) at once. More often, only a single pin needs to be set, cleared, or toggled. To perform single-bit operations, a read-modify-write operation is performed on the port value, as follows:

```
LDB AL, ioport1               ;Read current port value
ORB AL, #10000001B            ;Set bits 0,7 without affecting bit 1-6
LDB ioport1, AL               ;Write the new port value
```

The above operation could all be done in one instruction:

```
ORB ioport1,#10000001B        ;reads ioport1, modifies it,
                              ;writes ioport1 back
```

The following operation could be used to complement a port pin value:

```
XORB ioport1,#1000000B        ;Complement P1.7
```

In all of the above operations, the present value of the port is read, modified, and then written back. However, there are some instances in which the above operations would yield less than desirable results. One example involves Port 2, which has two output-only pins. Since output-only pins have no read buffer logic, the port's current value cannot be read. Obviously, a read-modify-write operation does not make sense.

In another example, suppose Port 1 consisted of both inputs and outputs. Since all the pins of Port 1 are quasi-bidirectional, any of the pins used as inputs are required to be written as a one. This presents a problem when the port is read, modified, and then written back. If any of the input pins have a zero value being driven externally, performing a read-modify-write operation would write a zero back to the input port pin. Writing a zero to a quasi-bidirectional pin enables the strong low driver and causes a short-circuit condition when a one is driven externally.

To solve the example problems mentioned above, a mirror image of the port is maintained in memory. The operation (set, clear, toggle) to be performed on the port pins occurs on the memory image, which is then written to the port directly. To illustrate, the following operation is used to toggle P1.7 and assumes P1IMAGE can contain only a zero for those outputs driving a zero:.

```
XORB P1IMAGE,#10000000B        ;Complement P1.7 Image
LDB ioport1, P1IMAGE           ;Write new value to Port 1
```

There are other ways to get around the problem, such as making sure that the OR operation contains a one for any bidirectional pins used as inputs. However, it is important to remember that mixing inputs and outputs on the same port requires special attention when modifying an output value..

6.2.3. Port 3 and Port 4 Access

Accessing Port 3 and 4 as I/O is easily done from the internal Port 3 and Port 4 register (IOPORT34) at location 1FFEH. Note that Ports 3 and 4 can be accessed only as a word. It is not possible to read or write Port 3 alone or Port 4 alone. To use Ports 3 and 4 as I/O ports, the device must be set for internal execution (see "Port 3 and Port 4 Considerations" on page 6-11).

Since the LD and ST instructions require the use of internal registers, it may be necessary to first move the port information into the lower register file before using the data. If the data is already internal, the LD is unnecessary. For instance, to write a word value to Ports 3 and 4:

```
LD intreg,portdata             ;Register <- data
                               ;Not needed if already internal
ST intreg,1FFEH                ;Register -> Port 3 and 4
```

To input from Ports 3 and 4 requires that ones first be written to the port registers to set up the input port configuration circuit. This will put the ports in a high-impedance mode. Note that the ports are reset to this input condition. If zeroes have been written to the port, then ones must be rewritten to pins that are to be inputs. Reading Port 3 and 4 from a previously written zero condition is as follows:

```
LD intrega,#0FFFFH          ;Setup port change mode pattern
ST intrega,1FFEH            ;Register -> Port 3 and 4
                            ;LD & ST not needed if
                            ; previously written as ones
LD intregb,1FFEH            ;Register <- Port 3 and 4
```

Note that while the format of the LD and ST instructions are similar, the source and destination directions change.

6.2.3.1. PORT 3 AND PORT 4 CONSIDERATIONS

The pins of Ports 3 and 4 have two I/O functions. They function as either an open-drain bidirectional port, a bidirectional system address/data bus (with strong low **and** high pin driver outputs), or both. How Ports 3 and 4 function depends on the state of the EA# pin during the last reset and on the need to access external program/data memory.

The EA# pin not only determines the operation of the Port 3 and 4 pins but also affects what happens when the ports are accessed. If EA# is low during a device reset, Ports 3 and 4 are used only to support the system address/data bus. A read or write to the Ports 3 and 4 address, 1FFEH, is interpreted as an external bus access, not a port access. If required, Ports 3 and 4 can be "reconstructed" by the circuit shown in Figure 6-7. This circuit uses standard latches and drivers to imitate the open-drain I/O function. A bus read/write access to address 1FFEH must be decoded to enable the latches properly.

When EA# is sampled high during a device reset, Ports 3 and 4 have two functions: I/O and system address/data bus. Normally the ports function as open-drain I/O pins. A zero written to the port latch at address 1FFEH enables a strong low driver, while a one written to the port latch disables the strong low driver and floats the pin. Writing a one to the port pins enables them to be used as inputs. The read or write to the port is interpreted as an internal access, so no external bus cycle is executed. Ports 3 and 4 remain open-drain I/O ports at all times unless an external bus cycle is executed.

Any access to an address not interpreted as an internal address results in the execution of an external bus cycle. To perform the bus cycle, Ports 3 and 4 temporarily switch function to support the system address/data bus. This means that the port pins have strong high and low drivers during address and data write portions of the bus cycle, and that they float during data read portions of the bus cycle. After the bus cycle completes, the port pins switch back to their I/O port function. The state of the port pin after the switch back to the I/O function depends on the value last written to the port latch.

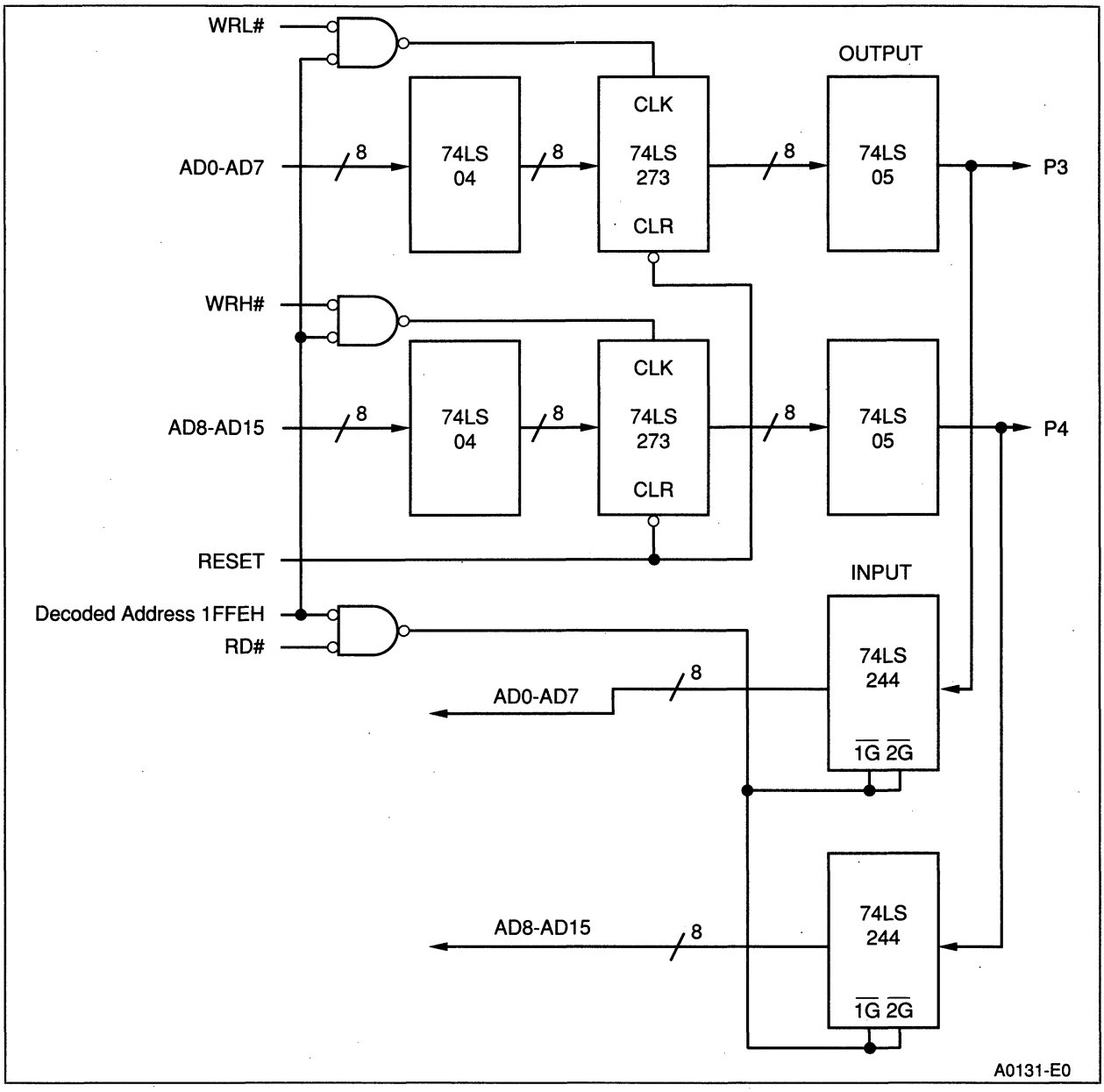

Figure 6-7. Ports 3 and 4 Reconstruction

In applications that use the system address/data bus, the value programmed into the port latch has some significance. The port latch value appears on the bus during idle times (i.e., during those times that Ports 3 and 4 are not the system address/data bus). Writing a zero to Ports 3 and 4 results in the system address/data bus being driven to a strong low when idle, while writing a one to Ports 3 and 4 causes the system address/data bus to float.

Which way the port latches should be written depends on the application. Writing zero means that the bus are driven low when idle, while writing one means the bus may float when idle. A floating bus could cause excessive current draw, but is easily fixed by placing pull-up resistors on the system address/data bus. Having the bus drive low removes the need to have resistors, but the design needs to ensure that no other device can drive the bus at the same time (possibly causing a short-circuit condition).

When Ports 3 and 4 are used exclusively as the system address/data bus, terminating the bus with pull-up resistors causes 0FFFFH to be read off the bus when an instruction fetch is made to an unimplemented memory location. Fetching a 0FFH operand causes the device to reset. Assuming that an instruction fetch to unimplemented memory is an indication of a system or software failure, having the device reset itself is good and prevents further device operation.

6.3. HARDWARE CONNECTION HINTS FOR QBD PORTS

When using the quasi-bidirectional ports as inputs tied to switches, series resistors may be needed if the ports are written to internally after the part is initialized. For example, writing a zero to a port pin that is externally tied to V_{CC} through a switch causes a short-circuit (high current) situation. The amount of current sourced to ground from each pin can exceed 20 mA. Sourcing this much current could make it possible to exceed the specification limits for the pin or port.

This potential problem can be solved in software or hardware. In software, never write a zero to a pin being used as an input. In hardware, a 1-KΩ resistor in series with each pin will limit current to a reasonable value without impeding the ability to override the high-impedance pull-up. The problem is less severe when the inputs are tied to electronic devices (e.g., TTL or CMOS gates) instead of switches, since the electronic devices tend not to sink excessive amounts of current.

Writing to a quasi-bidirectional port with electronic devices attached to the pins requires special attention. Consider trying to toggle P1.1 as an output:

```
XORB ioport1,#00000010B        ;Complement P1.1
```

A problem can occur when the instruction executes. Even though P1.1 is driven high by the 8XC196KC/KD, it could possibly be held low externally. This typically happens when the port pin drives the base of an *npn* transistor, which in turn drives whatever needs to be toggled in the outside world. The base of the transistor will clamp the port pin to the transistor's V_{BE} above ground, typically 0.7V. The 8XC196KC/KD will input this value as a zero, even if a one has been written to the port pin. When this happens, the XORB instruction will always write a one to the port pin's SFR, and the pin will not toggle.

The problem can best be solved by the external driver design. Using a series resistor between the port pin and the base of the transistor often works, by raising the voltage present on the port pin. A software solution is to keep a byte in RAM as an image of the data to be output to the port. Any time the software wants to modify the data on the port, it can modify the image byte and copy it to the port.

If a switch is used on a long line connected to a quasi-bidirectional pin, a pull-up resistor is recommended, to reduce the possibility of noise glitches and to decrease the rise time of the line. On extremely long lines that are handling slow signals, a capacitor may be helpful in addition to the resistor to reduce noise.