FEATURE

Introducing Microprocessors Part 6

This month we look at I/O, methods for input to and output from a microprocessor system.

MIKE TOOLEY

The general learning objective for part six is that readers should be able to describe the internal architecture and facilities provided by a typical programmable parallel I/O device. The specific objectives for Part Six are as follows:

4.1 I/O Methods

4.1.1 Distinguish between memory mapped and port I/O techniques.
4.1.2 Draw and interpret the block diagram of a simple memory mapped I/O and state the function of each block.
4.1.3 Draw and interpret the block diagram of a simple port I/O and state the function of each block.

4.2 Programmable Parallel I/O Devices

4.2.1 Describe and distinguish between serial and parallel data transfer.
4.2.2 Describe the reasons for using parallel I/O devices and explain why they need to be programmed.
4.2.3 Draw and interpret a block diagram to show the simplified internal architecture of a representative programmable parallel I/O device.
4.2.4 State the function of each of the principal internal elements of a representative programable parallel I/O device.

Input and Output

All microprocessor based systems require means of inputting and outputting data. The input/output (I/O) provision in a microprocessor based system will obviously be dictated by the application for which it is intended. As an example, a microprocessor based central heating controller might have as its input a small keypad together with one or more temperature sensors interfaced to the system by some additional signal conditioning circuitry. The output of the central heating controller might comprise a simple status display using light emitting diodes together with relay outputs for controlling a boiler and a central heating pump.

The I/O provision in a personal computer would be vastly different. User input would be provided via a conventional QWERTY keyboard and joystick port while outputs would be provided for a TV or monitor (VDU) and also for a printer using the popular Centronics parallel interface. In addition, an RS-232C serial I/O port may be provided in order to facilitate data exchange with other microcomputers or with a modem.

Despite the obvious difference in the I/O provision of the two systems, it is eminently possible for them to use identical I/O devices (at least as far as the parallel I/O provision is concerned).

Parallel versus serial I/O

The personal computer mentioned earlier has provision for both parallel (Centronics) and serial (RS-232C) I/O. Parallel I/O involves transferring data one byte at a time between the microcomputer and peripheral along multiple wires (usually eight plus a common ground connection). Serial I/O, on the other hand, involves transferring one bit after another along a pair of lines (one of which is usually a ground connection).

In order to transmit a byte (or group of bytes) the serial method of I/O must involve a sequence or stream of bits. The stream of bits will continue until all of the bytes concerned have been transmitted and additional bits may be added to the stream in order to facilitate decoding and provide a means of error detection.

Since data present on a microprocessor data bus exists in parallel form, it should be apparent that a means of parallel-to- serial and serial-to-parallel conversion will be required in order to implement a serial data link between microcomputers and peripherals.

Memory mapped versus port I/O

In the last part we briefly mentioned that I/O can be "mapped" into the address space of a microprocessor based system. In such cases, the processor does not distinguish between memory and I/O when it performs its read and write operations; the processor treats I/O devices in much the same way as RAM and ROM.

Some processors (notably the 8085 and Z80) can make a distinction between memory and I/O devices and have control signals which are used to qualify their read and write operations. In order to make use of this facility, a number of software instructions are provided which deal exclusively with input (read) and output (write) operations to I/O devices. This type of I/O is usually described as "port I/O".

In the case of the 8085, a single control line is used to inform the system whether

the current read or write cycle relates to I/O or whether it is directed at memory. Not surprisingly, this line is marked IO/M; the line is taken high to denote an I/O operation and low to signal a memory read or write.

In the case of the Z80, two separate control lines are provided. Both of these lines are active-low (*ie*, asserted when taken to logic 0.) The Z80's MREQ (memory request) signal is asserted (taken low) when the processor is performing a memory read or write whereas its IOREQ (input/output request) signal is asserted (taken low) when the processor is performing an equivalent operation to a peripheral I/O device.

The architecture of a representative microcomputer using memory- mapped I/O is shown in Fig. 6.1. Note that the six most significant addresses (A10 to A15) are fed to an address decoder, the outputs of which are used to drive the active-low chip select (CSC) lines of the ROM, RAM, and I/O devices. The I/O device, for example, is selected (enabled) whenever CS2 goes low. The address decoding is, of course, arranged so that only one of the chip select lines goes low at any time. The action of the address decoder can be explained using Table 6.1.

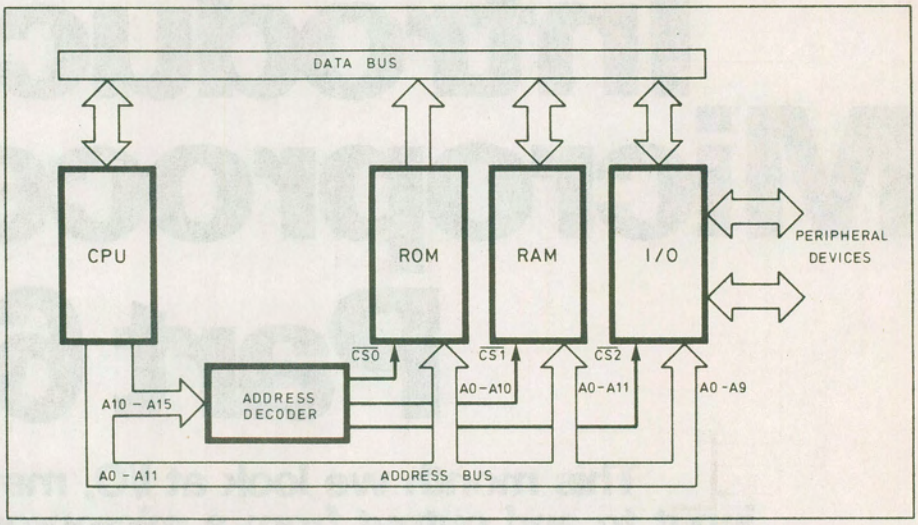

Fig. 6.1. Architecture of a representative microcomputer using memory-mapped I/O.

A15	A14	A13	A12	A11	A10	CS2	CS1	CS0	Block selected	Address range (hex.)
0	0	0	0	X	X	1	0	1	RAM	0000–0FFF
1	0	0	0	0	0	0	1	1	I/O	8000–8400
1	1	1	1	1	X	1	1	0	ROM	F800 FFFF

Table 6.1. Truth table for the address decoder in Fig. 6.1.

Problem 6.1

Refer to Fig. 6.1 and Table 6.1.

(a) What is the capacity of the ROM?

(b) How many I/O addresses are provided for?

(c) How much RAM space is provided?

The architecture of a representative microcomputer using port I/O is shown in Fig. 6.1. This system is a little more complex than its memory mapped counterpart and it is important to note that the processors MREQ and IOREQ control signals are fed to the address decoders and are used in the production of the ROM, RAM and I/O block chip selected signals (CS0, CS1, and CS2 respectively). The upper address decoder logic is arranged so that CS0 and CS1 can only be asserted when MREQ is taken low, (*ie*, when the processor is performing a memory read or write). Note that address lines A10 to A15 are still used by the upper decoder to distinguish between ROM and RAM.

The lower address decoder logic is arranged so that CS2 can only be asserted when the IORQ line is taken low. The internal registers of the I/O device correspond to a set of four unique addresses determined by the state of the two least significant address lines (A0 and A1).

To illustrate the difference between memory mapped and port I/O as far as software is concerned, consider the simple problem of reading a byte from one I/O address and transferring it to another. Let's assume that the memory mapped system has input and output addresses of 8001H and 8003H while the corresponding addresses for the port based system are 01H and 03H. Typical assembly language routines for 6502 (memory mapped) and Z80 (port I/O) processors would take the form:

6502 (memory mapped)

```
LDA $8001 ; Load accumulator from input
from input
STA $8003 ; and transfer to the output
```

Z80 (port I/O)

```
IN A, (01H) ; Read the input port and
OUT (03H), A ; transfer to the output port
```

Readers should compare the foregoing fragments of code noting how the "load from memory" (LDA) and "store in memory" instructions of the 6502 are replaced by the IN and OUT instructions of the Z80. Note also the difference in conventions for expressing hexadecimal numbers (the leading $ and trailing H) and that the port addresses for the Z80 are contained within brackets.

Problem 6.2

Refer to Fig. 6.2 and Table 6.2 What operation is being carried out when:

(a) MREQ is low, IOREQ is high, and A10 to A15 are all low

(b) MREQ is low, IOREQ is high, and A10 to A15 are all high

(c) MREQ is high, IOREQ is low, and A0 to A7 are all low?

Parallel I/O devices

Microcomputer I/O is greatly simplified with the use of one or more sophisticated VLSI devices, the operational characteristics of which can be established by writing data to one, or more, internal registers. This property is the key to making devices suitable for a wide range of applications and provides the microprocessor system designer with a great deal of flexibility: the I/O configuration of a system may be modified using nothing more than a short sequence of software instructions.

VLSI parallel I/O devices enjoy a variety of names depending upon their manufacturer. Despite this, parallel I/O devices are remarkably similar in internal architecture and operation with only a few subtle differences distinguishing one device from the next (see Data Card No. 6 for details).

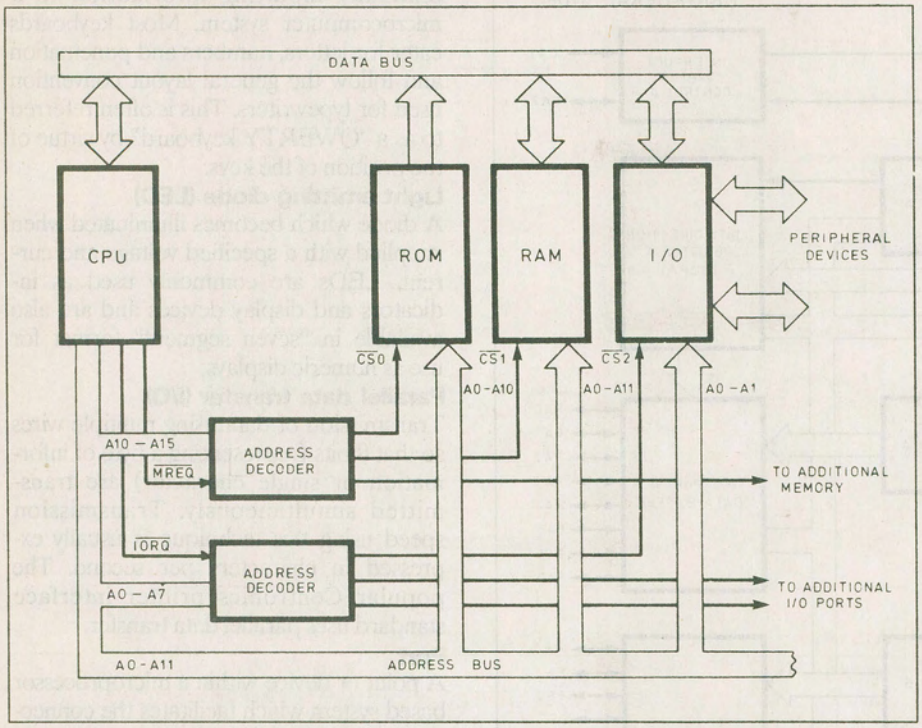

Fig. 6.2. Architecture of a representative microcomputer using port I/O.

A15	A14	A13	A12	A11	A10	MREQ	IOREQ	CS2	CS1	CS0	Block selected	Address range (hex.)
0	0	0	0	X	X	0	1	1	0	1	RAM	0000–0FFF
1	1	1	1	1	X	0	1	1	1	0	ROM	F800–FFFF

X=don't care

A7	A6	A5	A4	A3	A2	A1	A0	MREQ	IOREQ	CS2	CS1	CS0	Block selected	Address range (hex.)
0	0	0	0	0	0	X	X	1	0	0	1	1	I/O	00–01

X=don't care

Table 6.2. Truth tables for the address decoders in Fig. 6.2.

The internal architecture of a representative parallel I/O device is shown in Fig. 6.3. Despite the complexity of this diagram, parallel I/O is really quite straightforward. When used for output, the I/O device must latch data from the system data bus into a byte-wide output register. This register will preserve the data written to it so that it can be presented via a buffer, to the outside world. When used for input, the I/O device must contain an octal tri-state buffer which, when enabled by the appropriate read instruction, will place the data received from the peripheral onto the system bus.

In common with most programmable parallel I/O devices, the chip shown in Fig. 6.3 provides two independent 8-bit I/O ports (labelled A and B). Each port has an Output Data Direction Register (DDRA and DDRB), and a Control Register (CRA and CRB). Interrupt Status Control circuitry is also provided for "handshaking", the aptly named process by which control signals are exchanged between the microcomputer and peripheral devices.

The function of the signals shown in Fig. 3 may be summarized:

CPU Side

D0 to D7 System data bus.

CS Active-low chip select line. This line is asserted whenever the CPU wishes to read or write to the I/O device.

RS0 and RS1

These Register Select lines used to distinguish the internal registers of the I/O device. Note that since the two Register Select lines are connected to two of the address bus lines (usually A0 and A1), the device will occupy four memory locations.

R/W Read/Write

This is the standard CPU control signal.

IRQA and IRQB.

These two lines are used to provide interrupt request signals for the CPU. Each line is associated with a different port.

RESET Active lowsystem reset line.

When asserted, this signal places the internal registers of the I/O device in a known state.

Peripheral Side

PA0 to PA7 Port A I/O lines

0 corresponds to the least significant bit (LSB) while 7 corresponds to the most significant bit (MSB).

CA1 and CA2 Handshaking lines for port A

CA1 is an interrupt input while CA2 can be used as both an interrupt input and peripheral control output.

PB0 to PB7 Port B I/O lines

0 corresponds to the least significant bit (LSB) while 7 corresponds to the most significant bit (MSB).

CB1 and CB2 Handshaking lines for Port B

CB1 is an interrupt input while CB2 can be used as both an interrupt input and peripheral control output.

As mentioned earlier, programmable devices can be configured under software control. Several options are normally provided including;

(a) making all eight lines of a designated port inputs

(b) making all eight lines of a designated port outputs

or (c) individually configuring port lines as either inputs or outputs.

This process is carried out by sending (writing) a Mode Setting Word to the Control Register. A subsequent word may also be written in order to define the direction of lines within a port and this byte will be placed in the corresponding Data Direction Register.

The bit positions in each Data Direction Register correspond to similarly numbered peripheral lines in the port concerned. A logic 0 placed in a particular position will define the corresponding peripheral line as an input, and vice versa.

The Register Model of the programmable I/O device is shown in Fig. 6.4. Note that this model dispenses with much of the detail shown in Fig. 6.3 and simply treats the I/O device as two groups of three registers. We shall examine the process of programming I/O devices in much greater detail in part Eight.

Next Month

We shall be dealing with methods for interfacing microprocessor based systems with such commonplace devices as LEDs, relays, and switches.

Glossary

Keyboard

Group of push button switches used for

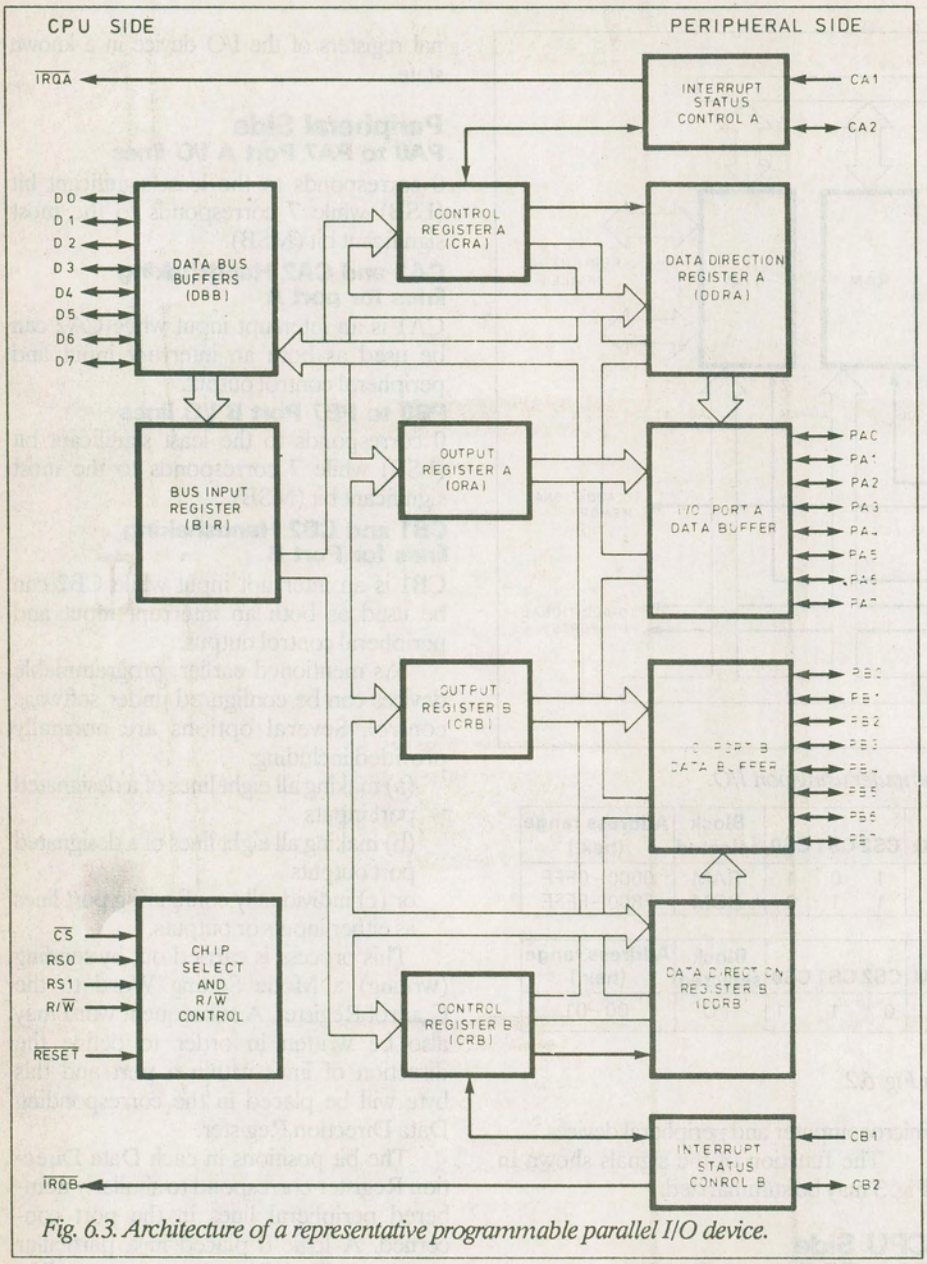

Fig. 6.3. Architecture of a representative programmable parallel I/O device.

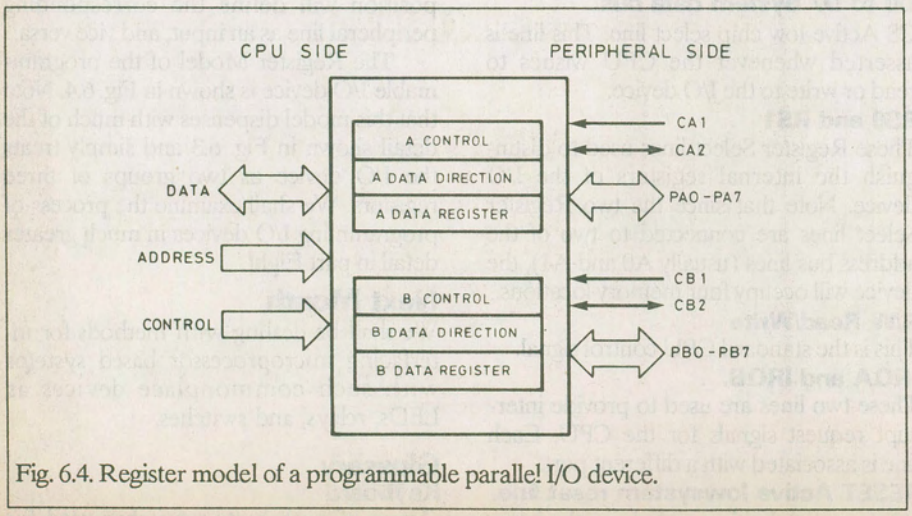

Fig. 6.4. Register model of a programmable parallel I/O device.

manually inputting information to a microcomputer system. Most keyboards cater for letters, numbers and punctuation and follow the general layout convention used for typewriters. This is often referred to as a "QWERTY keyboard" by virtue of the position of the keys.

Light emitting diode (LED)
A diode which becomes illuminated when supplied with a specified voltage and current. LEDs are commonly used as indicators and display devices and are also available in "seven segment" format for use as numeric displays.

Parallel data transfer (I/O)
Transmission of data using multiple wires so that 8 bits (representing a byte of information or single character) are transmitted simultaneously. Transmission speed using this technique is usually expressed in characters per second. The popular Centronics printer interface standard uses parallel data transfer.

Port
A point or device within a microprocessor based system which facilitates the connection of external (peripheral) devices so that they may communicate (exchange information) with the system. The configuration of a port is often to be determined by software instructions sent to the programmable I/O device which is used to implement the port.

Programmable I/O device (PIO)
An input/output device (invariably single VLSI chip) which can be programmed by the user to provide an interface with external devices and components (e.g. relays, switches, keyboards, etc.)

Relay
A single or multiple switch which is usually operated by electromagnetism. Relays provide a high degree of electrical isolation between a microprocessor and the circuit which it is being used to control. Relays are also capable of switching currents greatly in excess of those available from a microprocessor system.

Serial data transfer
Data transmission using a single wire (plus ground) so that each bit of a character is transmitted in turn. Transmission speed is usually expressed in bits per second.

Answer to Problems

6.1 (a) 2K bytes
(b) 1K bytes
(c) 4K bytes
6.2 (a) Read or write operation to RAM
(b) Read operation from ROM
(c) Read or write operation to I/O ■