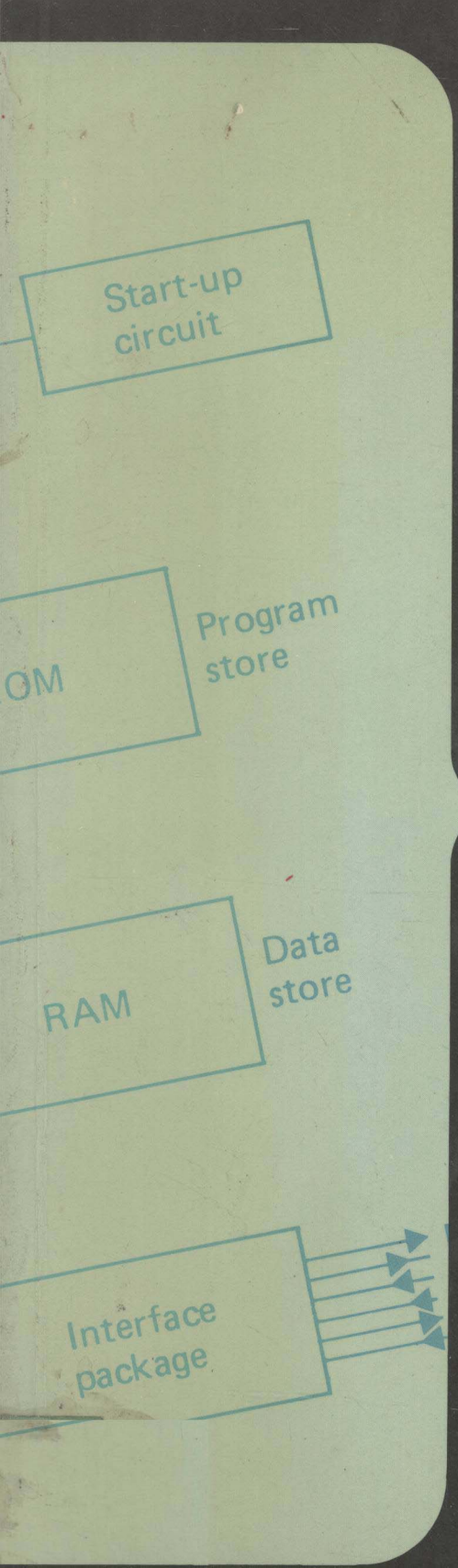


Interfacing to Microprocessors

J. C. Cluley

INTERFACING TO MICROPROCESSORS

J. C. Cluley

Senior Lecturer in Electronic and Electrical Engineering, University of Birmingham

M

First published 1983 by
THE MACMILLAN PRESS LTD
London and Basingstoke
Companies and representatives
throughout the world

Printed and bound in Great Britain
at The Pitman Press, Bath

ISBN 0 333 34061 2 (paper cover)

Preface

The majority of the microprocessors now produced are installed as the controlling element in larger, generally non-electronic systems. Examples of these applications are domestic products such as cookers and washing machines, cash registers, weighing machines and much industrial process control and test equipment. The user is usually not aware that a microprocessor is embedded in the equipment, since it begins operations automatically when power is applied.

Although the microprocessor itself is produced in large quantities to a standard design, the extremely wide range of applications is reflected in the many different ways in which it must be interfaced to the outside world.

The design of these external circuits and their interactions with the microprocessor are the main topics of this book. Although the principles of data input and output with microprocessors are similar to those used with minicomputers, the detailed arrangements differ substantially. In particular, in order to afford the maximum flexibility in use, nearly all microprocessor input and output packages are programmable.

In planning the book I have assumed that the reader has some understanding of the way in which a microprocessor operates and how instructions are retrieved from the program store and executed. This information is well presented in *Understanding Microprocessors* by B. S. Walker (published by the Macmillan Press, London and John Wiley, New York) and I have not attempted to duplicate it. Although the book is mainly concerned with hardware and system design, I have included short program segments to illustrate the way in which interfaces can be controlled.

In chapter 4 I have included a brief survey of transducers. These are not always mentioned in many courses as they are not regarded as electronic devices. They are however essential features of many systems, and the availability of cheap and reliable transducers is often the critical factor in deciding whether a particular microprocessor application is economically justified.

I have given examples of the use of several of the more popular microprocessors, rather than confining the book to only one model, so as to give some indication of the variety of features and design philosophy available. Readers who require more information on the components in any particular family of microprocessors will need to consult the manufacturers' design data.

With the use of microprocessors now extending to a wide range of products, processes and equipment, there is a corresponding need for engineers and scientists in many disciplines to acquire an understanding of microprocessor applications and interfacing. This material is also being incorporated into many courses at universities and polytechnics.

I hope that this book will provide a satisfactory and comprehensive introduction to this developing and important subject.

J. C. CLULEY

Contents

Preface ix

1 Basic Microprocessor Architecture 1

1.1 Introduction 1
1.2 Components of a Microprocessor System 1
1.3 Bus Lines and Bus Signals 3
1.4 Signal Flow During Data Transfers 6
1.5 Input/Output Transfer Methods 7
1.6 Interrupts 9
1.7 Direct Memory Access 11

2 Principles of Data Transfer 13

2.1 Device Addressing 13
2.2 Memory-mapped Input/Output 15
2.3 Input/Output Package Design 17
2.4 Combined Interface Packages 18
2.5 Counter/Timer Packages 19
2.6 Serial Input/Output 19

3 Input/Output Packages 23

3.1 Parallel Input/Output Packages – the M6820 and M6821 23
3.2 Programming the M6821 26
3.3 Flag Testing 29
3.4 Interrupt Handling with the M6821 PIA 31
3.5 The Intel 8255 PPI 32
3.6 Programming the 8255 36
3.7 Serial Interfaces for the M6800 38
3.8 The Intel 8251A USART 42
3.9 M6852 Synchronous Serial Data Adapter (SSDA) 45
3.10 Counter/Timer Packages 45
3.11 DMA Controllers 47
3.12 Priority Interrupt Controllers 49
3.13 Other Input/Output Packages 53

4 Transducers and Signal Conversion 54

4.1 Signal Transformations for Input and Output 54
4.2 Digital Inputs 54
4.3 Digital Transducers 56
4.4 Analogue Sensors 60
4.5 Temperature Measurement 62
4.6 Signal Conditioning 63
4.7 Non-linear Operations 66
4.8 Digital to Analogue Converters 69
4.9 Analogue to Digital Converters 71
4.10 Slew Rate Limitations 75
4.11 Tracking Converters 78
4.12 Digital Coding of Analogue Signals 79
4.13 ADC Interfacing 79
4.14 Other Types of A/D Converter 82
4.15 Protection from Interference 82
4.16 Output Circuits 85
4.17 Motor Drive Circuits 86

5 Single-chip Microprocessors 88

5.1 Applications of Single-chip Systems 88
5.2 Typical One-chip Devices – the Intel 8048 Family 88
5.3 The Intel 8021 and 8022 92
5.4 The Zilog Z8 Family 92
5.5 The Motorola MC6801 94

6 Practical Problems and Applications 96

6.1 Sensing Data from Mechanical Switches 96
6.2 Manual Input Devices 98
6.3 Input from Keyboards 101
6.4 Reverse Scanning 103
6.5 Driving Digital Displays 107
6.6 Waveform Generation 111

7 Microcomputer Buses 115

7.1 The Development of Standard Buses 115
7.2 The S–100 Bus 115
7.3 The IEEE–488 or IEC 625 Bus 120

7.4 The E78 Europa Bus 122
7.5 The IEEE–796 Bus 122
7.6 Other Bus Standards 123
7.7 Serial Data Standards 125

8 System Testing and Development 128

8.1 System Development 128
8.2 ROM Simulators 129
8.3 Board Testing 129
8.4 In-circuit Emulators 130
8.5 Logic State Analysers 131
8.6 Asynchronous Display 132
8.7 Graphical Displays 133
8.8 Signature Analysis 133
8.9 Other Fault-finding Aids 136

9 Interfacing to 16-Bit Microprocessors 137

9.1 16-Bit Bus Organisation 137
9.2 The Intel 8086 Family 137
9.3 The Motorola M68000 138
9.4 The Zilog Z8000 140

Bibliography 142

Appendix A: Pin Connections of the 8080 Family 146
Appendix B: Pin Connections of the 8085 Family 148
Appendix C: Pin Connections of the 6800 Family 150
Appendix D: Pin Connections of the Z80 Family 152

Index 154

1 Basic Microprocessor Architecture

1.1 Introduction

A microprocessor has been defined as 'an integrated circuit which performs the central processing function in a digital computer system'. The essential requirement for successful manufacture is a high volume of production, so that the very large cost of design and mask-making can be spread over typically hundreds of thousands of devices.

The problem of the microprocessor user is to take this standard product and tailor it to the particular requirements of his own environment and application. The main aspects of this operation are the unique program of instructions which determines the action of the microprocessor, and the way in which it is attached or 'interfaced' to the system with which it operates.

In this book we are concerned mainly with the design and arrangement of the interface, and to a lesser extent with programming. The program examples given show how data transfers can be effected, how the state of external devices can be tested and how they can be controlled. Program design and strategy will not be discussed in any detail.

1.2 Components of a Microprocessor System

A microprocessor alone is unable to perform any useful actions. In order to work successfully it must be connected to other units, generally on a shared or 'party line' basis, so as to provide all the functions of a complete computing system. A typical arrangement is shown in figure 1.1, and comprises in addition to the microprocessor the following items.

(a) Program store

This is a read-only device, which has been programmed during manufacture (masked ROM), or by the user (fusible link PROM or erasable PROM). It is supplied with an address, control and timing signals, and will then drive on to the data lines the contents of the location addressed.

(b) Data store

This is writable storage (usually referred to as RAM) used for holding information upon which the computer is operating. It has the same set of connections

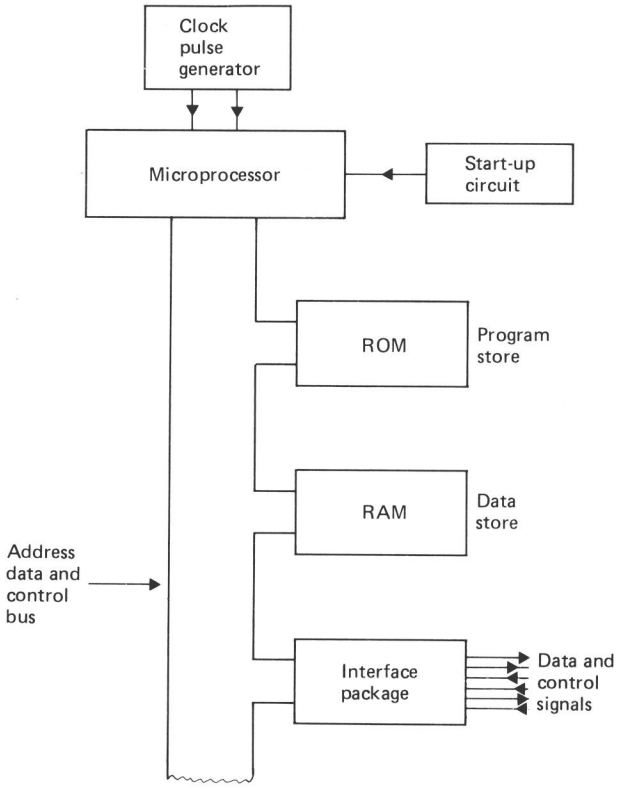


Figure 1.1 The components of a basic microprocessor system

as the program store but in addition needs control signals to order either a read or a write operation.

(c) Peripheral interface packages

Some external devices can be connected directly to the microprocessor, but for an output operation data storage is almost invariably required. In addition, various status flags and timing signals may be required, and it is thus usually much simpler to use the manufacturer's input/output (I/O) interface package than to construct an interface using a number of small-scale integrated circuits. Most applications require parallel data transfers but serial data are needed for feeding data over telephone lines or attaching VDUs and teleprinters. Interface packages are available which incorporate all of the necessary logic for these applications within a single package.

(d) Clock pulse supply

Microprocessors require a continuous train of pulses to synchronise internal and external events, or for some versions two non-overlapping pulse trains. These clock pulses were originally generated by a separate package, their frequency being determined by a quartz resonator.

Later microprocessors included the clock pulse generator circuits, so that the only external component required was the quartz crystal.

(e) Start-up circuit

In order that the microprocessor shall set its internal registers in the proper state when the power is first applied, it is necessary to earth the reset or restart pin for some milliseconds before returning it to its normal +5 V level. This action may be performed either by a trigger circuit which senses the 5 V supply, or by a simple C–R circuit, both external to the microprocessor package.

1.3 Bus Lines and Bus Signals

The connections which link together the processor, the program and data stores and the peripheral units are called a 'bus' and can conveniently be separated into three groups, the data bus, the address bus and the control and timing bus. The bus lines cannot use conventional logic packages because they are required in some cases to operate in different directions at different times.

Thus for a 'write' operation data must flow along the data bus from a processor register to a peripheral device or to the data store. In a 'read' operation the direction of the data flow is reversed, and information from external sources is read into a processor register. In order to allow this bidirectional operation, devices which drive the bus are usually designed to have three output states, logic 1 output (high), logic 0 output (low) and off. In the off condition the circuit presents a high impedance to the bus, so allowing its potential to be determined by some other device connected to the bus. This arrangement is called 'tri-state' drive and is usually provided by two series transistors as shown in figure 1.2(a).

An alternative arrangement sometimes used on lightly loaded lines is the 'open-drain' circuit of figure 1.2(b). Here the single transistor is cut off when the device is not transmitting data, and is only turned on when transmitting a logical 0. A single resistor in the range 2–10 kΩ is connected between the bus line and +5 V to hold the bus at logic 1 level when no driver is energised. This circuit has the disadvantage that its positive-going transition is rather slow, being determined by the product of the resistor value and the total stray capacitance of the bus. The tri-state circuit has a much faster transition since the upper transistor is fully conducting to signal a logic 1 and its resistance is typi-

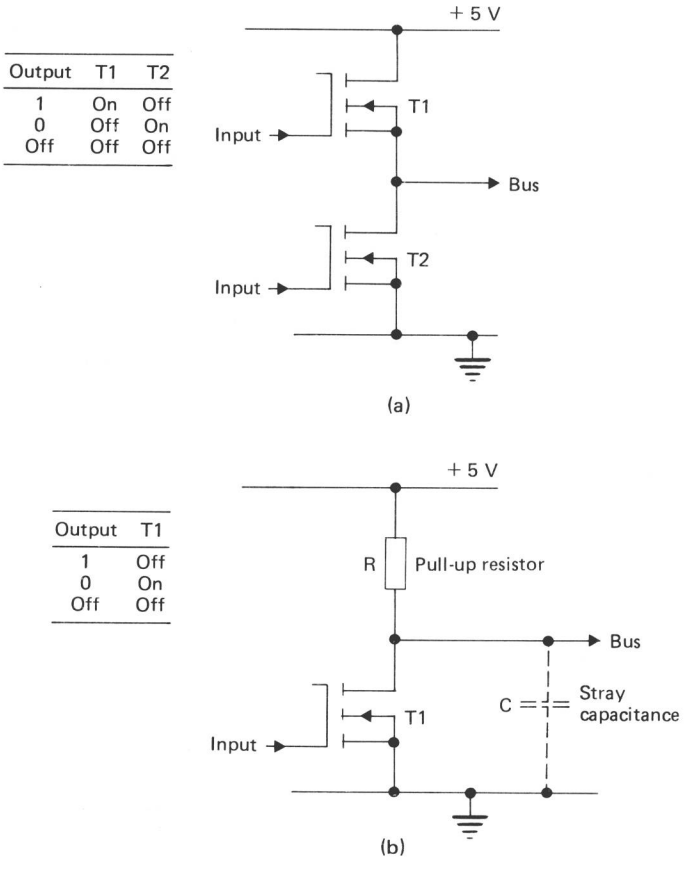


Figure 1.2 (a) Tri-state bus driver; (b) open-drain bus driver

cally only a few hundred ohms. Consequently all bus lines which are liable to substantial capacitance loading and require bidirectional working (the data and address buses particularly) are normally driven by tri-state circuits.

Unlike fast minicomputers, the bus lines are usually confined to one or two printed circuit boards, are relatively quite short, and so do not need to be treated as transmission lines or be properly terminated. This saves a great deal of power and allows transistors with much lower current ratings to be used.

To indicate typical values, we note that the specification of the M6800 microprocessor puts an upper limit of 130 pF on the total capacitance loading of the data bus lines.

Assuming a system with this capacitance and a pull-up resistor of 3.3 kΩ the time constant for a positive-going transition with the open-drain driver circuit of figure 1.2(b) is given by

$$T = CR$$
$$= 130 \times 10^{-12} \times 3300 \text{ s}$$
$$= 0.43 \ \mu s$$

The 10–90 per cent rise time is then

$$T_R = 2.2\,T$$
$$= 0.95 \ \mu s$$

Clearly this is much too long if the microprocessor has a 1 MHz clock rate and must thus handle pulses which are typically 0.5 μs long. The arrangement however is generally used for control lines which are active only on the negative-going transition and have no critical timing on the positive-going (slower) transition. Typical of these is an interrupt request line which is connected only to peripheral devices, not to storage packages, and so has less loading than the data lines.

Note that in this case the important negative-going transition will be much faster, because the resistive component of the time constant is almost entirely the drain-source resistance of the driver transistor, which will be considerably less than 3.3 kΩ.

If the calculation above is repeated for the tri-state circuit, using a typical driver resistance value of 300 Ω, the time constant is eleven times less and the rise and fall times are both 86 ns. This neglects any switching time for the transistor; if some allowance is included for this, the rise and fall times would be in the region of 100 ns.

Taking this value, the mean transistor current during the rise or fall of bus voltage is given by

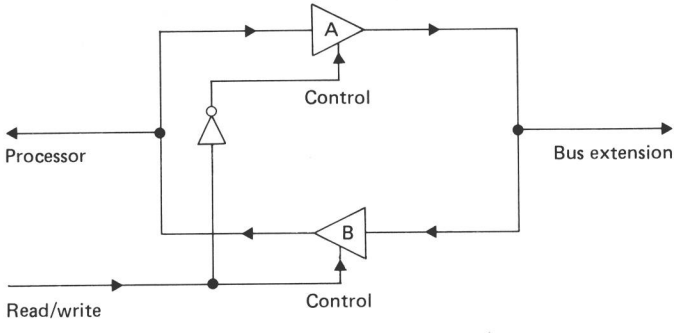
$$i = \frac{CV}{t} = \frac{100 \times 10^{-12} \times 5}{100 \times 10^{-9}} \text{ A}$$

$$= 5 \text{ mA}$$

For a 1 MHz pulse rate each transistor will conduct for only 100 ns in each 1 μs, so giving an average current of only 0.5 mA. These peak and mean currents are well within the capacity of the physically small transistor of the microprocessor and its supporting packages. If however the bus lines are so long that they need correct terminations at both ends, as in some minicomputers, the driver current needed to pull the bus voltage down to 0 V may exceed 50 mA.

Microprocessor systems with a number of store packages and peripheral devices attached to the lines may give a total bus capacitance which exceeds the permitted figure. In this case the bus must be split into sections using buffers or bus-extenders, so that no section has more than the permitted loading.

The type of device used depends upon the nature of the bus line. For unidirectional lines such as the clock and interrupt lines and the address lines, a simple unity-gain amplifier suffices, but for the data lines which have to transfer information in both directions two amplifiers connected back-to-back as shown in figure 1.3 are used. In order to prevent oscillation or latch-up, the amplifiers are switched or gated, and arranged so that only one is operative at a time. The control line is connected to the read/write bus, so that the outgoing (from the microprocessor) amplifier operates for write, and the incoming amplifier for read.



Action	R/W line	A	B	Data direction
Write	Low	On	Off	Out
Read	High	Off	On	In

Figure 1.3 Bidirectional bus driver

1.4 Signal Flow During Data Transfers

We first consider the action needed during the output or write instruction which transmits a byte of data in parallel over the data lines. Since this will be present on the bus for usually less than a microsecond, some form of storage is necessary, as few if any devices can operate correctly with such a brief sample of data. The interface needs to perform the following functions

(a) Detect that the processor is making a data transfer involving this particular interface. This normally requires the recognition of either an address or a device number on the address lines.

(b) Detect that the processor is executing a write transfer. This involves sensing either the read/write line or the write strobe line.

(c) Detect the correct moment when the data are present on the data bus and available for storage. This involves sensing either a clock pulse or the write strobe pulse.

Generally the three signals resulting from (a), (b) and (c) are ANDed together to provide a clock pulse for a set of D-type bistables. The D terminals are permanently connected to the data lines, so that after the clock pulse the data transferred will be stored and available at the Q output of the bistable. It will remain until the next data transfer occurs.

To ensure that the output is in a known state when the power is first applied, the clear terminals of the bistables may be connected to the reset or restart line so that when power is applied to the system the device register is automatically cleared.

Before a data transfer occurs it may be necessary to examine a status flag in the interface to ensure that the device is ready to accept data. Usually a complete byte will be read from a control and status register, and one bit of this will be examined.

For a read operation no storage is generally provided. The functions required are

(a) As for a write operation (address sensing).

(b) Detect that the processor is executing a read instruction, using the read/write line or the read strobe.

(c) Detect when the processor requires the data to be gated on to the data lines. This timing information is derived from a clock pulse or a read strobe.

Again the three signals from (a), (b) and (c) are ANDed together, this time to provide a gating signal which turns on a tri-state driver which puts the data on to the data lines.

Other signals may need gating into the combined load signal, depending upon the processor, for example valid memory address (VMA) or input/output request (IORQ).

These input and output logic circuits can be assembled from small-scale integrated circuits but it is generally simpler to use the manufacturer's parallel I/O package which includes storage for output transfers and data bus drivers, in addition to control registers. These will be described in more detail in chapter 3 where the characteristics of some current I/O packages are discussed.

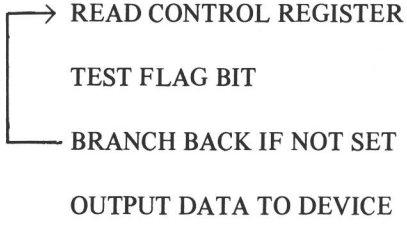
1.5 Input/Output Transfer Methods

Having considered the principles governing the logic of input and output operations, we next examine the program instructions involved, and the processor actions which take place. We can distinguish three different arrangements, generally called program-controlled transfers, interrupt driven transfers and direct memory access (DMA) or autonomous transfer.

Program-controlled transfers are the simplest from both the hardware and software aspects. They occur when the program executes an I/O instruction, consequently the instant of transfer is decided by the program alone. Since

this is not synchronised with the activities of the peripheral device there may be a conflict if the processor, for example, outputs data when the device is not ready to accept them. The data would then be lost.

In order to prevent this, a bistable or 'flag' is incorporated into the interface package, which can be set by the device and read by the processor. If we assume that the device sets the flag to logic 1 when it is ready for data, the program is arranged to test the flag, and if it is not set, to continue testing it. When the device sets the flag the program will branch to an output instruction which transfers the data. For convenience the data transfer is usually arranged to clear the flag ready for the next transaction. The loop of instructions which precedes the transfer is generally called a 'waiting loop' and the sequence is of the form

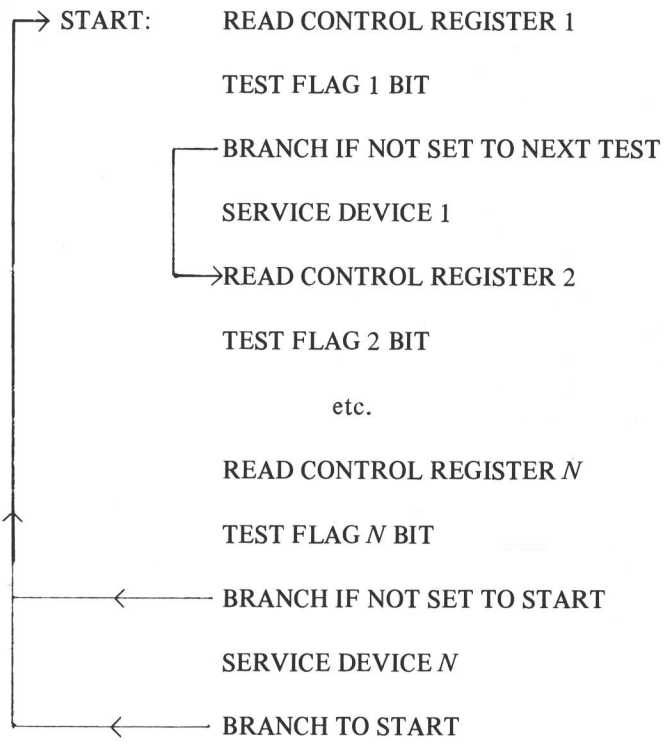


This is sometimes called 'wait and go' operation, and it allows the program execution to be delayed until the device is ready. However, it will not allow program execution to be accelerated if the device is ready before the I/O transfer instruction is reached.

Difficulties may arise if several devices are attached to the processor, since a device which is not ready will hold up the program and prevent any other device from being serviced.

An alternative program strategy to that given above is to test each device and, if it is not ready, branch to test the next device. Each device is tested in turn and only when a device is found which is ready for data transfer does the program branch to effect the transfer. The procedure is called 'polling' and requires the processor to remain in a loop; this may include some other processor action if time permits. The essential requirement is that even in the worst case, when every device is serviced, each one is polled often enough to input or output all the data required, without missing any. Generally the maximum data rate for each device is known, and the critical time is the time between data transfers required by the fastest device. The longest time to execute a loop of the program must be somewhat less than this. For example the maximum data rate for a teleprinter is ten characters per second. Thus any processor which has a teleprinter attached to a parallel data port must poll the port at intervals of less than one-tenth of a second to determine whether a character has been received by the port.

The basic structure of the polling routine is



The instruction 'SERVICE DEVICE I' is usually a JUMP TO SUBROUTINE instruction, so that the program after completing the subroutine will return to test the flag connection to the device next on the list.

1.6 Interrupts

Although polling is a satisfactory process for a set of attached devices with roughly similar data rates, it cannot be used with fast devices if many other interfaces are also active. In such a case the fast devices need rapid service and there is no time to test all the other devices first. The solution is to use a program interrupt.

This means that the device requiring attention can cause the processor to leave the program that it is currently executing, and transfer to the device service routine. At the end of the routine the processor must resume its execution of the previous program.