apricot

Technical Reference Manual

apricot

Technical Reference Manual

ACKNOWLEDGEMENTS

ACT wish to acknowledge the invaluable assistance afforded by its consultants in bringing Project Apricot to fruition.

QED Project Design,
Clarence House,
Newport, Gwent.

CAPA,
Aberdeen House,
Highbury Grove,
Islington,
London N2.

PREFACE

The Technical Reference Manual for the ACT Computer Apricot is divided into three sections and a number of appendices, as detailed below. This manual is intended for programmers and engineers involved in hardware and software design for Apricot.

1.SYSTEM OVERVIEW

This section provides an overall description of the Apricot computer and is divided into three chapters:

1-1 HARDWARE, describes the hardware of Apricot in its basic configuration, and forms the introduction to the three elements which constitute the microcomputer;the System Unit,the Keyboard Unit and the Display Unit.

1-2 SOFTWARE, provides a brief description of the operating system and its interface to the associated BIOS. An introduction to the software modules which constitute the BIOS is also provided.

1-3 OPTIONS, forms the introduction to the hardware options available for the microcomputer, at the time of printing the manual.

2.HARDWARE DETAIL

This section contains detailed descriptions of all the hardware aspects of the microcomputer and is divided into a number of chapters, as detailed overleaf. Each of the three elements of Apricot are discussed in detail with major circuit elements (e.g. Disk Drives, Power supply etc.) also having separate descriptions.

3.SOFTWARE DETAIL

This section contains a detailed description of all software aspects of the BIOS and is also divided into a number of chapters. The first provides a detailed description of the BIOS as a whole with subsequent chapters detailing the individual hardware drivers.

APPENDICES.

A number of appendices are included in this manual which provide general hardware reference information and also associated software information of specific use to systems/application programmers.

CONTENTS

SECTION 1.SYSTEM OVERVIEW

1-1 HARDWARE

1-2 SOFTWARE

1-3 OPTIONS

2

CONTENTS

SECTION 2.HARWARE DETAIL

2-1 SYSTEM UNIT

2-2 SYSTEM BOARD

2-3 INTERRUPT CONTROLLER

2-4 TIMER

2-5 CRT CONTROL

2-6 FLOPPY DISK INTERFACE

CONTENTS

2-7 SERIAL INTERFACE

2-8 PARALLEL INTERFACE

2-9 SOUND GENERATION

2-10 EXPANSION SLOTS

2-11 DISK DRIVE

CONTENTS

2-12 KEYBOARD

2-13 DISPLAY UNIT

SECTION 3.SOFTWARE DETAIL

3-1 PROGRAMMERS GUIDE TO THE BIOS

3-2 SCREEN DRIVER

CONTENTS

3-3 MICROSCREEN DRIVER

3-4 KEYBOARD DRIVER

3-5 CONTROL DEVICE DRIVER

3-6 DISK INPUT/OUTPUT SYSTEM

3-7 BOOT ROM

3-8 CALCULATOR

3-9 SOUND GENERATION

3-10 AUXILIARY DEVICE/SERIAL PORT

CONTENTS

3-11 GRAPHICS SYSTEM EXTENSION SYSTEM

APPENDICES

SECTION 1:
SYSTEM OVERVIEW

HARDWARE

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The Apricot is the generic name given to a family of 16-bit microcomputers, possessing a diverse range of hardware facilities to suit the needs and requirements of the user in a wide variety of applications. Each member of the family is produced starting from an Apricot in its basic operational configuration. This is then transformed into a different member of the family by the addition of options, as required. A more detailed treatment of the options is provided in a subsequent chapter.

All Apricot computers are composed of three separate elements; a System Unit, a Keyboard Unit and a Display Unit as illustrated on Figure 1.

The System Unit is the area of the computer, which houses the majority of the processing elements, the system memory components and the interface circuitry for peripheral equipment. The Unit also contains the following items:

(a) At least one micro floppy disk drive.
(b) A power supply module.
(c) A loudspeaker.
(d) A cooling fan.

HARDWARE

The Keyboard Unit consists of, a standard layout typewriter keyboard, a calculator keypad, a number of programmable function keys with associated LED indicators, a set of "fixed" function keys, and a 2 line by 40 character MicroScreen. The Keyboard Unit also contains processing circuitry which provides the interface between the keys and the System Unit.

The standard Display Unit is a high resolution 9 inch monochrome display which fits into a shallow recess on the top of the System Unit. The Display Unit tilts, swivels and slides, enabling the operator to position the screen to obtain the optimum viewing angle. An anti-reflective mesh is incorporated on the screen to minimise the possibility of eye fatigue.

In its basic configuration, the Apricot Computer incorporates the following:

(a) An Intel 8086 16-bit processor.
(b) An Intel 8089 16-bit input/output processor.
(c) 256 Kbytes of dynamic RAM.
(d) A single 3.5 inch MicroFloppy disk drive.
(e) A Centronics parallel port for connecting a printer.
(f) An RS232C serial interface for data communications.
(g) A port for connecting a "mouse" device.
(h) A sound generator to provide audio feedback.
(i) Two internal expansion slots to cater for RAM expansion and connecting other optional facilities.
(j) A port for connecting a light pen plus associated ancillary electronics.

The basic configuration of the microcomputer can be expanded by the addition of two further options, in addition to the optional facilities available via the internal expansion slots. These two further options are as follows:

(a) A second 3.5 inch MicroFloppy disk drive.
(b) A numeric data processor (Intel 8087) to increase the processing capabilities of the microcomputer in mathematical and scientific applications.

Connections for the printer, data communications equipment and mains input are located at the rear of the System Unit, together with connectors for the Keyboard and Display Units. The expansion slots and the connection for the light pen and associated circuitry are located within the System Unit. The connection for the "mouse" device is located at the rear of the Keyboard Unit.

SYSTEM UNIT

All the processing and control circuitry within the System Unit is contained on a single printed circuit board, designated the System Board, which is fitted horizontally inside the System Unit housing. The major elements of the System Board are illustrated on the block diagram, Figure 2a/2b. The System Board incorporates the following items:

(a) The system processing elements.
(b) The Boot PROMs.
(c) 256 Kbytes of DRAM.
(d) A floppy disk controller.
(e) A parallel printer interface.
(f) A dual channel serial interface, one channel for the Keyboard and the other for the RS232C connector.
(g) A display controller.
(h) Two expansion slots.

The processing elements on the System Board are arranged in a multiprocessing configuration which includes two Intel 16-bit microprocessors; an 8086 central processing unit (CPU) and an 8089 input/output processor (IOP). The two microprocessors are connected in parallel and share a common bus structure. This configuration is termed (by Intel), a local multiprocessing configuration. The number of microprocessors within the local multiprocessing configuration is increased to three if the optional Intel 8087 numeric data processor (NDP) is fitted.

In the local multiprocessing configuration, only one of the microprocessors has access to the shared address, control and data buses at any one time. The input/output processor is employed to perform the bulk of routine data transfers, under the direction of the central processing unit.

HARDWARE

High speed DMA data transfers, from a selected source (memory or peripheral) to a selected destination (memory or peripheral), are performed by the IOP. The IOP is able to transfer data whilst the central processing unit is engaged in executing tasks which do not require use of the buses.

The 8087 numeric data processor extends the processing capabilities of the Apricot computer, by performing arithmetic and comparison operations on numeric types which can vary from 16 to 80 bits. It is also able to execute numerous transcendental functions (e.g.tangent and logarithmic functions).

The Apricot computer uses 3.5 inch Sony disk drives and employs the IBM system 34 format in the double density mode, with 512 bytes per sector (and 9 sectors per track). The total storage capacity of each disk is 315 Kbytes of formatted data. The floppy disk controller incorporates the necessary capability to control a second floppy disk drive without any modification to the hardware.

The serial interface provides two separate channels; one channel for communication between the System Board and the Keyboard Unit; the second channel for communication between the System Board and data communications equipment. The serial link to the keyboard is a full duplex asynchronous communications channel, which transmits and receives data at a fixed baud rate.

The RS232C channel can be programmed to operate in both asynchronous and synchronous modes, with the programmer having independent control over transmit and receive baud rates. The rates can be generated by either an internal programmable timer or determined by the external data communications equipment. Both byte and bit oriented synchronous modes are available. These include the byte oriented modes, Monosync and Bisync, and the bit oriented modes, SDLC and HDLC.

The two expansion slots are identical and allow the basic configuration of the computer to be expanded by the addition of optional expansion cards. The expansion bus consists of the processing system buses (address, control

and data buses) plus additional control and timing signals. These include interrupt request lines and control lines for DMA transfers.

Regulated supply voltages, for the System Board, Keyboard Unit, Monitor Unit, floppy disk drive(s) and any optional boards connected to the expansion slots, are supplied from the power supply module located within the System Unit. A switch on the rear of the System Unit switches the mains supply through to the power supply module. An LED in the centre of the Keyboard acts as a power supply present indicator.

KEYBOARD UNIT

The Keyboard Unit provides the normal interface between the user and the microcomputer. The keys of the Keyboard are divided into a number of functional groups, as follows:

(a) The standard typewriter, cursor control and editing keys occupying the majority of the keyboard space.
(b) The calculator keypad located at the far right of the keyboard.
(c) A group of general function keys (grey keys located above the typewriter keys).
(d) A group of "soft" function keys. (The six membrane keys located directly below the MicroScreen).

The system reset button is located on the right hand side of the Keyboard and has a one second delay to prevent the accidental resetting of the system.

In addition to the keys, the Keyboard Unit has a number of display features, which are the six LEDs and the MicroScreen. These are controlled by software from the System Unit, which transmits the display data to the Keyboard via the serial link.

The six LEDs are used to signify whether the six soft function keys are available for operation. The MicroScreen is a two line by forty character liquid crystal display unit, with each character consisting of a five by seven dot matrix.

HARDWARE

A thumbwheel control on the right hand side of the Keyboard Unit adjusts the viewing angle of the MicroScreen.

The MicroScreen can be used in a variety of applications; e.g. for displaying:

(a) A real time clock,consisting of the time, the day and the date.
(b) Labels for the six 'soft' function keys.
(c) A copy of the data as it appears on the CRT display.
(d) System prompts.
(e) Operands and results during calculator operations.

The electronic circuitry of the Keyboard Unit is contained on a single printed circuit board fitted horizontally within the Keyboard Unit housing. A block diagram of the keyboard circuitry is shown on Figure 2a/2b.

All the keys are of a capacitive type with keyboard scanning performed using a single chip microcomputer. The microcomputer translates the selected key switch into a scan code, which is transferred to the System Unit via the serial link. The use of scan codes enables application programs to redefine the code (or sequences of codes) generated by the system, as required.

All communication between the Keyboard and the System Unit is handled by the microcomputer within the Keyboard. In addition to transmitting the keyboard scan codes to the System Unit, the microcomputer also provides the System Unit with the following serial data:

(a) Current time and date information on request from the System Unit.
(b) Control codes generated by the movement of a "mouse" device, when connected to the Keyboard mouse port.

The time and date information is generated by a real time clock/calendar chip on the Keyboard circuit board. A battery (standard PP3) back-up supply is connected to the clock/calendar to allow it to continue functioning when the mains supply is switched off.

DISPLAY UNIT

The standard Display Unit is a high resolution 9 inch green on black display. A long persistence phosphor (P39) is used to reduce operator eye fatigue by minimising display flicker. A control next to the recessed carrying handle adjusts the brightness of the display.

The Display Unit is connected to the System Unit by means of a six way cable, which carries the video data, synchronising pulses and +12V supply.

The Apricot computer is capable of displaying bit mapped images with a resolution of 800 pixels horizontally by 400 pixels vertically. In the character mode, a text display area of 80 characters by 25 lines is available, using a character cell of 10 pixels by 16 pixels. The video attributes in the character mode are:

(a) Reverse
(b) Highlight
(c) Underline
(d) Strike-through

HARDWARE

1. Monitor
2. System Unit
3. Keyboard
4. Microscreen tm
5. Touch Sensitive Keys
6. Fixed Function Keys
7. Door
8. Brightness Control

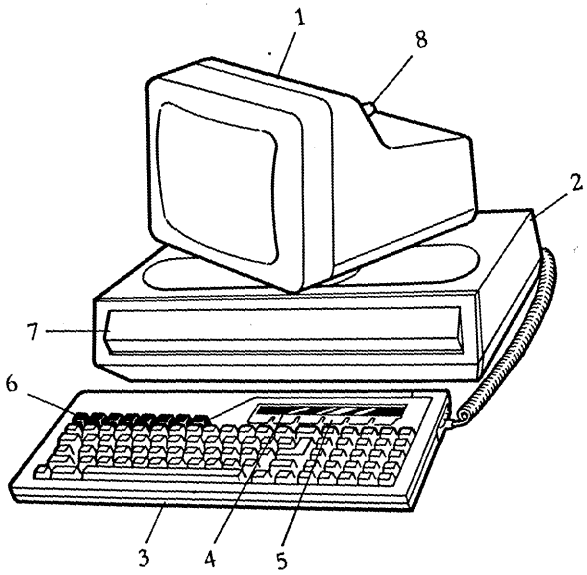


Figure 1a. Apricot Computer.

9. Video Cable
10. Keyboard Cable
11. Keyboard Connector
12. Centronics Connector
13. Serial Connector
14. Video Connector
15. Mouse Port
16. Mains Switch
17. Fuse
18. Mains Input

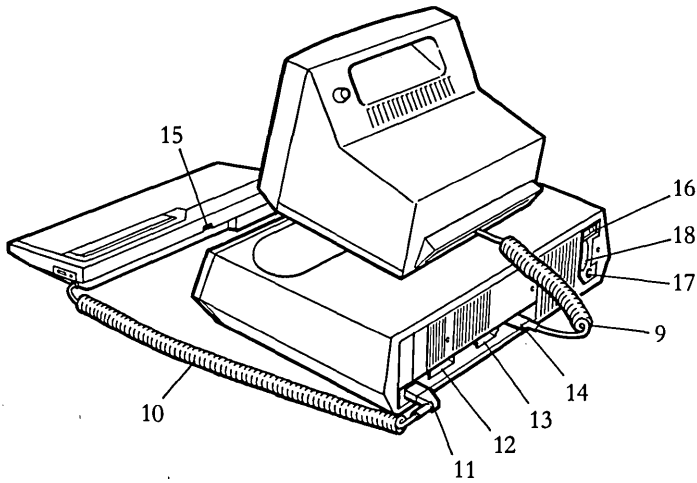


Figure 1b. Apricot Computer.

HARDWARE

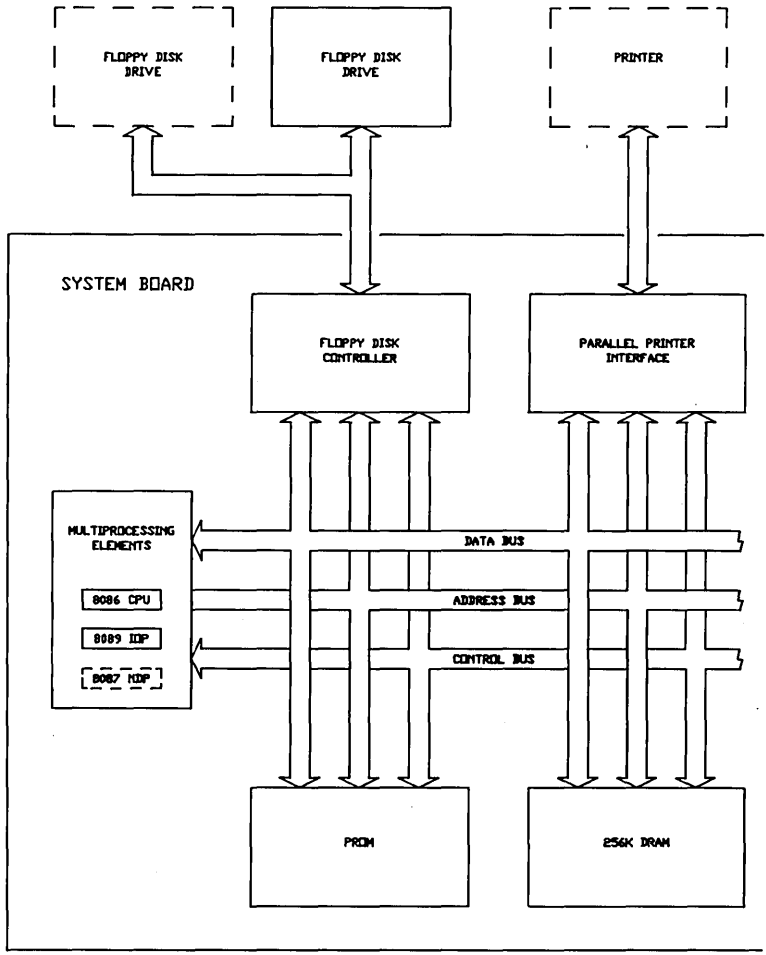


Figure 2a. Apricot Computer block diagram

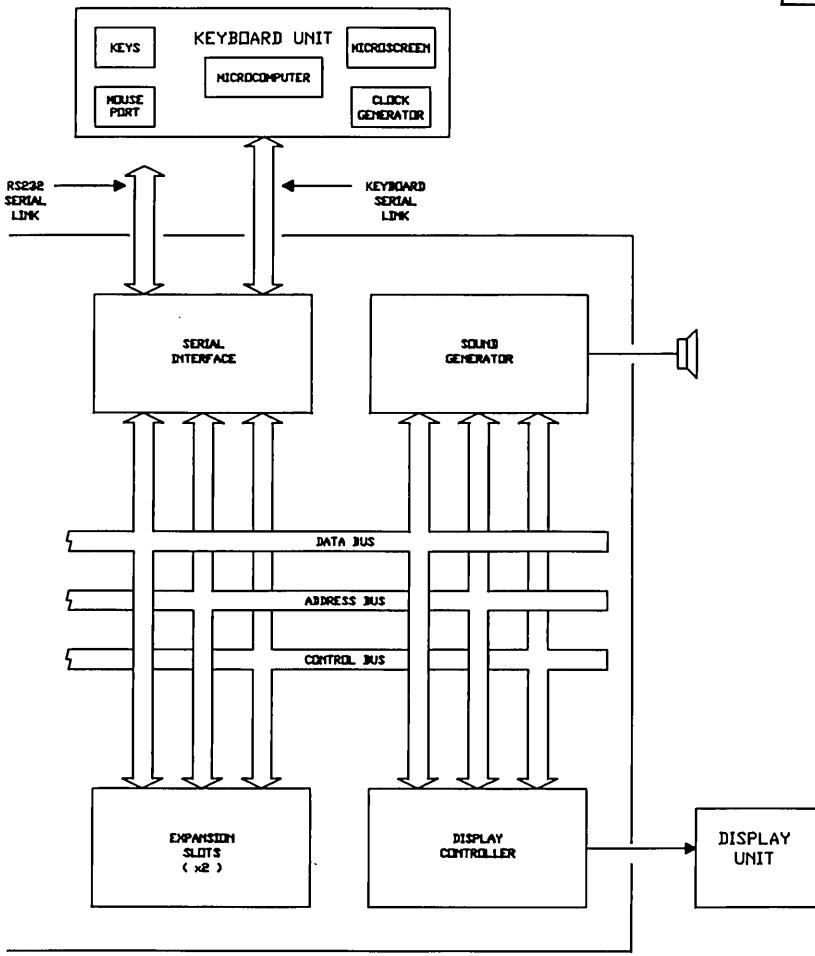


Figure 2b. Apricot Computer block diagram

List of Contents Page

INTRODUCTION

The control software for the Apricot Computer consists of two modules: the BIOS and the DOS, collectively referred to as the OPERATING SYSTEM. The BIOS (an acronym for Basic Input Output System) is generally 'in charge' of the hardware in the computer. The DOS (Disk Operating System) is the interface between the user, application program and the BIOS. The DOS is loaded into memory by the BIOS when the system is booted up and then takes control of the entire system. It can be considered as a "black box", which can request various services from the BIOS. For example, if the DOS wants to print a character on the screen, it has to interrupt the BIOS and give it the necessary instructions to carry out the operation.

The DOS is Microsoft's MS-DOS Version 2.0, which has many advanced features to make the running of application software extremely efficient. The BIOS was written by ACT (Advanced Technology) Ltd. specifically to run in conjunction with MS-DOS 2.0 on the Apricot, and contains a number of additions to make applications software both more powerful and more user-friendly.

SOFTWARE

MS-DOS 2.0 is a direct descendant of MS-DOS 1.25 which became widely used on 3rd generation 16-bit microcomputers such as the ACT Sirius 1, and (as PC-DOS) on the IBM PC. It was written by MicroSoft Corporation, one of the worlds leading micro-computer software houses. Cosmetically similar to CP/M-like operating systems, most of its advantages are with its internal commands, such as UNIX-type features including Piping, tree-structured directories and Background operations.

The BIOS consists of a number of hardware device drivers which MS-DOS makes use of. There are drivers for the keyboard, screen, MicroScreen, communications ports, clock, disk drives, sound and various hardware add-ons. There are also software drivers, which control graphics, background printing, disk cache and calculator.

The BIOS was written in 8086 assembly language and Pascal. Most of the hardware interfacing code, such as the keyboard, video display driver and low-level device I/O was written in assembler, leaving the code which involves large-scale decision making such as the disk handler to be written in Pascal. In source form, it consists of a number of modules which have to be linked together to form the BIOS.

BIOS FEATURES - OVERVIEW OF CAPABILITIES

The Apricot BIOS is a modular system written using the latest structured programming techniques. It is a fast, efficient piece of code which is capable of making maximum use of the Apricot's advanced hardware features.

This BIOS has many features implemented over and above the those incorporated in other IO systems.

Foreground and Background Operations

The Apricot BIOS has been implemented in such a manner that fully-fledged "foreground" and "background" processing is supported. That is all hardware operations to and from I/O drivers are buffered and interrupt driven. This makes it possible to use MS-DOS in a real-time environment with operations (such as printer spooling) being executed in the background.

Multi-sector Reads and Writes

The 8089 Input/Output Processor has been programmed to perform submerged (in the 'background' or simultaneous with 8086 operations) disk read and write functions on multiple sectors. This increases the speed of most disk operations and fits into the philosophy of MS-DOS 2.0 multiple driver request operations.

Tri-buffer

1. Disk Cache Memory. The cache memory is used by the disk driver to hold a number of disk sectors, and operates in the following manner:

Disk READ — The disk driver, on receiving a request for a sector to be read, first examines the cache to find out if the sector is resident in memory. If resident, the sector is immediately returned to the DOS. If not resident, the sector will be read from disk and if there is room within the cache, the sector is saved, as well as returned to the DOS. If there is no space, the cache overwrites the sector image within the cache which has not been accessed for the longest period of time with the new sector.

Disk WRITE — When a disk write request is made to the disk driver, the sector is written to disk as well as being saved within the cache. If there is no space, again the sector image within the cache not accessed for the longest period of time is overwritten with the new one. All sector writes to disk are performed in the background, so the BIOS does not need to wait around until the operation is complete before returning to the DOS. This background operation is possible because of the 8089 IOP and the cache.

2. Print buffer. The Print buffer has the same effect on printing as the disk cache on disk writes. To the DOS and the user, printing occurs at a much faster rate.

As a command is issued by MS-DOS for the Bios to print a character on a connected line printer, the Bios will place this character in a 2K buffer, and returns control back to MS-DOS immediately. When the lines from the printer state that it is ready for another character, the next entry in the buffer is sent to the printer to be printed and room is made in the buffer for another character.

3. Communications buffer. The communications buffer allows high speed transfer (up to 9.6 kbaud) of data between the Apricot and other computers. The serial driver supports full XON/XOFF protocol to prevent the loss of data when the buffer fills.

Apricot Control Device

This is a software device which enables the programmer to pass parameters to routines in the BIOS for controlling various hardware and software functions without having to resort to assembly language programming. As a result no low-level hardware knowledge of the Apricot is required by applications programmers.

System Configuration Block

The system configuration block is a small block of memory which contains the reference values that the BIOS uses for port characteristics, character sets and keyboard tables. By changing certain parts of this area it becomes easy for the programmer to change the hardware and software configuration of the Apricot.

MicroScreen driver

A simple method of controlling the MicroScreen has been devised whereby it is consistent with the standard Apricot (Sirius compatible) screen. The Escape sequences utilised on the large Apricot screen are the same as on the MicroScreen, for example, the clear screen command is the same as on the large screen.

Time and Date Clock

The time of day clock (located within the Keyboard) has been fully integrated within the BIOS. So on power up, MS-DOS will display the correct time and date which can easily be changed with the TIME and DATE commands.

Calculator

From power up, a simple four-function calculator is available without the need to insert a disk. This calculator can also be accessed from within an application and results of calculations can be sent to the screen as if they were typed in from the keyboard.

General Hardware support

The BIOS also supports the Apricot hardware to a greater extent than on other systems:

The Keyboard. Each of the keys on the Apricot keyboard can be made to produce up to three different codes, when pressed in the Unshifted, Shifted or Control modes. These codes can be:

1. A single character.
2. A string of characters, maximum length 255.
3. A number of prefix codes can be sent, such as ESCAPE.
4. The key can be defined as LOCAL, where its output is sent directly to the screen driver.
5. The key can have a combination of attributes, where it can be affected by the CAPS lock, or SHIFT lock modes.
6. An auto-repeat mode can be selected where a depressed key continues producing its code until released

Screen Driver. The screen display is totally compatible with the Sirius. All the escape codes can be used directly, and the screen buffer is physically in the same place within memory. All the character attributes (reverse, underline,

high intensity, strikeout) are supported. Also included is the ANSI escape sequence set, (which is IBM PC compatible) allowing extremely advanced screen modes such as windowing and left and right scrolling to be incorporated within applications.

Character sets. The character set cell is Sirius Compatible, using the full 10x16 matrix, with shapes easily changed by using the Font Editor.

Communications. An auxilliary serial RS232 and a parallel Centronics-compatible port, fully supported within the BIOS, ensure that the Apricot immediately has access to many hardware add-ons such as printers, plotters and modems.

Graphics. The Digital Research Graphics System Extension, fast becoming the "industry standard" graphics application software is fully supported.

It has its own application interface in the form of a GDOS, to the graphics device driver (called the GIOS) which facilitates easy development of graphics applications on the Apricot. The full 800 x 400 pixel resolution is used by GSX, creating a very powerful graphics system.

OPTIONS

List of Contents Page

INTRODUCTION

NOTE: All options are installed internally within the Apricot. This task must only be undertaken by a qualified dealer or a trained service engineer.

The basic configuration of the Apricot Computer can be altered by the addition of various hardware options to reflect the differing requirements of the user. These hardware options are listed below and described in greater detail in subsequent paragraphs.

(a) Floppy disk drive options. These include the addition of a second 3.5 inch disk drive and also utilising 3.5 inch disk drives which use MicroFloppy disks with a total storage capacity of 720 Kbytes of formatted data.
(b) 128 Kbyte RAM expansion boards.
(c) Modem.
(d) Numeric Data Processor.

FLOPPY DISK OPTIONS

A second 3.5 inch floppy disk drive which uses single sided disks with a storage capacity of 315 Kbytes of formatted data can be fitted into the System Unit without any modification to the existing hardware, to provide a dual MicroFloppy disk drive system. Each disk has 70 tracks and is soft-sectored employing the IBM system 34 format in the double density mode with 512 bytes per sector and 9 sectors per track.

OPTIONS

An alternative version of the 3.5 inch floppy disk drive can be installed within the System Unit, as a direct replacement for the floppy disk drives described above. This alternative provides an increase in the size of the floppy disk storage space available.

The alternative version uses 80 track double sided disks (i.e. 160 tracks per disk) and also employs the same format, mode and sector size as detailed above. Each disk has the capacity to store 720 Kbytes of formatted data, producing a corresponding increase in total disk capacity for a dual MicroFloppy disk drive system to 1.44 Mbytes.

128KBYTE RAM EXPANSION BOARD

The 128 Kbyte RAM Expansion Board fits into the expansion slots within the System Unit. Both expansion slots can accommodate the RAM board, thus providing an increase in the size of the RAM within the Apricot to either 384 Kbytes (one board), or 512 Kbytes (two boards).

MODEM

The Modem option is a communications facility to allow an Apricot computer to transmit and receive data via the Public Switched Telephone Network (PSTN). The option is installed within the Apricot by inserting the Modem Board into one of the expansion slots and making a minor modification to the rear panel of the System Unit to replace the expansion plate.

The Modem Board consists of a modem and a microprocessor control system. The modem conforms to both CCITT V23 and V21 standards and operates in any one of the following software selectable modes:

(a) Full duplex 1200/75 bps.
(b) Full duplex 300/300 bps.
(c) Half duplex 1200/1200 bps.

The microprocessor control system acts as the interface between the modem and the system processing elements on the System Board, and also provides the Apricot with an

autodialler facility. Telephone numbers can be "dialled" via the Apricot keyboard, transferred to the microprocessor control system on the Modem Board and retained in its internal memory. Automatic dialling of the number stored in modem memory, is performed under the control of the Modem microprocessor.

NUMERIC DATA PROCESSOR

The Numeric Data Processor (NDP) option is an Intel 8087 16-bit microprocessor which can be fitted onto the System Board within the System Unit to extend the processing capabilities of the Apricot in mathematical and scientific applications. The NDP is a simple plug-in item which fits into a socket already provided on the System Board.

The NDP acts as a co-processor to the 8086 central processing unit (CPU), and fits into the local multiprocessing configuration on the System Board. In effect, the NDP provides an extension of the instruction set of the CPU, allowing the Apricot to perform arithmetic computations and comparison operations on numeric types of varying size from 16 to 80 bits. In addition to arithmetic and comparison operations, the NDP also executes numerous transcendental functions (tangents, logarithms etc.).

SECTION 2:
HARDWARE DETAIL

SYSTEM UNIT

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The System Unit is the area of the computer, which houses the processors, the system memory, disk drive(s), the Power Supply Unit, a cooling fan and all the interface circuitry for connecting the Keyboard, Display Unit and other peripheral equipment.

All the electrical and electronic components are contained internally within the plastic case. A shallow recess at the top of the case is provided for the Display Unit.

The Unit is designed for ease of transportation, incorporating a carrying handle which slides out from underneath the disk drive(s), and a sliding cover to prevent dust or any other objects entering the disk drive inputs during transit. The Keyboard can also be carried attached to the underside of the System Unit, utilising the clips provided on the base plate.

SYSTEM UNIT

Connectors for the Keyboard, the Display Unit, the mains input and other peripheral equipment are all located on the rear panel of the Systems Unit. The connectors from left to right when looking at the rear panel are as follows:

1. The Keyboard connector (9-pin female D-type).
2. A parallel printer port (36-way female Centronics).
3. An RS232C communications port (25-pin female D-type).
4. The Display Unit connector (9-pin male D-type).

Also included on the rear panel is the mains switch and the mains input fuse. The indicator on the switch is illuminated when the Systems Unit is switched on.

Two plastic expansion plates are located above the Keyboard connector on the rear panel. These are removed to allow external equipment to be connected to the expansion boards with a minimum of modification to the rear panel.

All the processing circuitry is contained on a single printed circuit board, the System Board, which fits horizontally into the base of the Systems Unit. Located above the System Board inside the Unit, is the Power Supply Unit, a loudspeaker and either one or two disk drives. The cooling fan is attached to the inside of the rear panel.

SYSTEM BOARD

The System Board incorporates:

(a) The system processors.
(b) The Boot Proms.
(c) Thesystem memory (256 kbytes of dynamic RAM).
(d) TwoExpansion Slots.
(e) Circuitry for controlling the monochrome Display Unit.
(f) A dual channel Serial Interface, one channel for communicating with peripheral equipment via an RS232C link, one channel for keyboard communications.
(g) A Parallel Interface for connection to a printer.
(h) An interface for controlling the disk drive(s).
(i) A sophisticated Sound Generator.

A description of the System Board and all the major circuit elements on the board listed above are detailed in subsequent chapters.

DISK DRIVES

One or two disk drives can be fitted inside the System Unit. In a single drive system, the disk drive is fitted on the left hand side (as viewed from the front). Single drive systems can be upgraded into a dual drive system by adding a second disk drive without any modification to the hardware.

The disk drives are mounted on a metal chassis assembly above the System Board, and secured in place by two pairs of screws in each of the chassis side plates. The disk eject button and the activity indicator of the disk drive fit through the front facia of the System Unit. The function of the disk eject button is self-explanatory. The activity indicator is illuminated when disk read and write operations are in progress, and also momentarily when the disk is first inserted into the drive.

A ribbon cable assembly connects the disk drive to the disk interface on the System Board. Regulated power supplies are supplied to the drive from the System Board, via a 4-way cable assembly.

The standard disk drives use 70-track single sided MicroFloppy disks with a total storage capacity of 315 kbytes of formatted data. A detailed description of these disks and further details of the disk drives can be found in the Disk Drive section.

POWER SUPPLY DETAILS

General

The mains input into the System Unit is filtered (by a line filter mounted on the rear panel), and then routed via the input fuse and the mains switch. Switching the System Unit on supplies the mains voltage to the power supply unit (PSU) and operates the mains powered cooling fan.

SYSTEM UNIT

The power supply unit is of switched mode design, providing regulated outputs of +12V, +5V and –12V for use by the System Board, the Keyboard, the disk drives, the Display Unit and any expansion boards fitted. The power supply components are housed in a shielded case. A fuse is located in the mains input line, internally within the unit.

DC Supply Distribution

The regulated outputs from the PSU are supplied to the System Board via a 7-wire cable assembly, terminated at both ends in Molex connectors. The PSU provides two separate regulated supplies of +12V, a single regulated supply of +5V and a single regulated supply of –12V. The maximum current rating for the four supplies are detailed below. The "V" prefix corresponds to a pin/number of pins on the PSU DC connector.

V1	(+5V supply)	6.0A
V2	(+12V supply)	1.5A
V3	(+12V supply)	2.1A
V4	(-12V supply)	0.25A

Distribution of the supplies from the System Board to the other areas are via the board wiring to the appropriate connector. The board provides:

1. +12V to the Display Unit via the Display Unit connector.
2. +12V and –12V to the Keyboard via the Keyboard connector.
3. +12V, +5V and –12V to the expansion connectors.
4. +12V and +5V to the disk drive(s) via Molex connectors and 4-wire cable assemblies.

Fuse rating

The rating of the fuse on the rear panel of the System Unit is dependent on the mains input voltage, as detailed below.

240V mains input – T 2A, 20 mm slow blow.
115V mains input – T 3A, 20 mm slow blow.

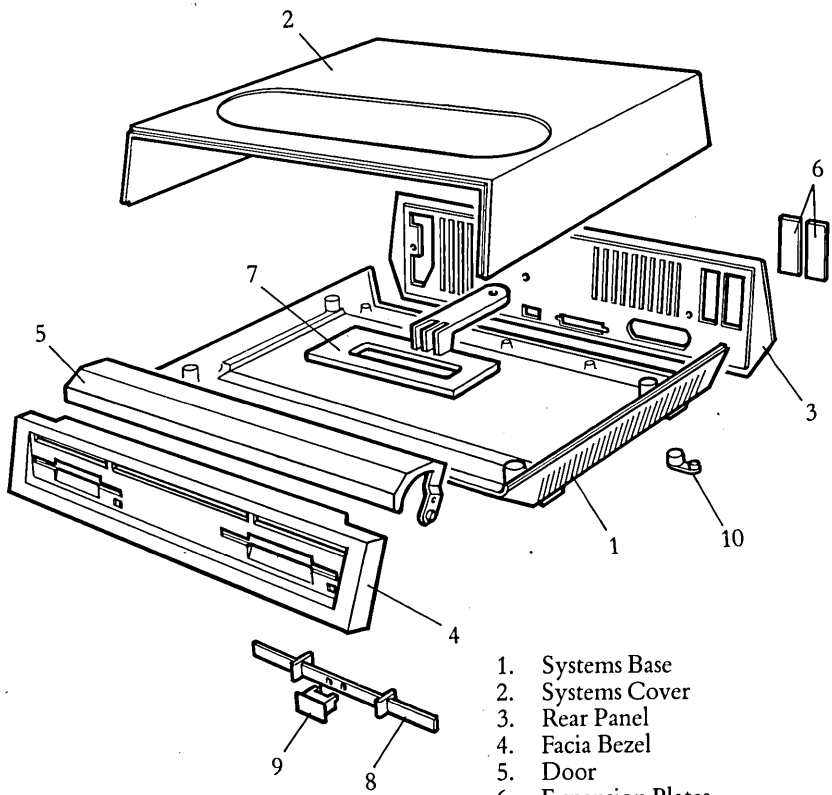
PHYSICAL DIMENSIONS

Height: 4.0 inches

Width: 16.5 inches

Depth: 12.5 inches

Weight: 14.2 lbs (Dual disk drive version, single disk drive version, 1.5 lbs less).



1. Systems Base
2. Systems Cover
3. Rear Panel
4. Facia Bezel
5. Door
6. Expansion Plates
7. Handle
8. Base Clip
9. Base Button
10. Keyboard Cable Clips

Figure 1. System Unit External Detail

SYSTEM UNIT

1. Disk Drive
2. Power Supply
3. Motherboard
4. AC Sub-Assembly
5. Main Chassis
6. Chassis Bridge
7. Expansion Slots
8. Loudspeaker

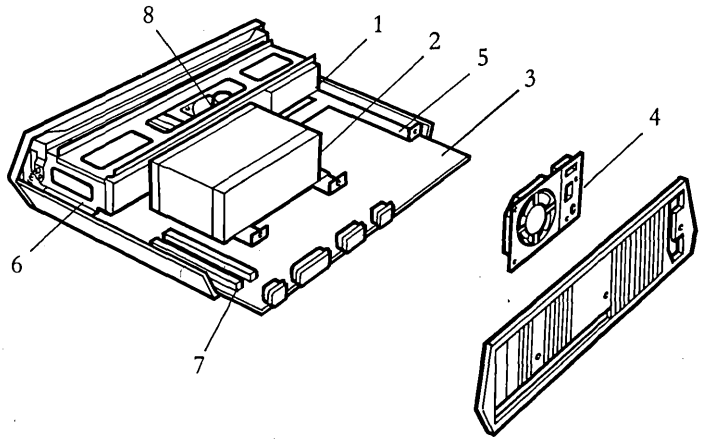


Figure 2. System Unit internal detail

List of Contents Page

INTRODUCTION

The architecture of the Apricot System Board can be split into 9 main sections, as detailed below:

1. Processors
2. Communications
3. Sound Generation
4. System Memory
5. Expansion Slots
6. CRT control
7. Floppy Disk control
8. Interrupt Handling
9. Timer

DESCRIPTION

Processors

The Apricot has, as standard, two 16-bit processors: the Intel 8086 microprocessor and the Intel 8089 I/O processor. Available as an option is the 8087 Mathematics co-processor.

SYSTEM BOARD

8086. The 8086 processor is directly language-compatible with the 8088 processor which is used on many 3rd Generation machines. It runs at a clock speed of 5 MHz, typically executing 1.4 million instructions per second, and has a 16-bit data-bus which enables it to read and write words into memory in one access as opposed to two with the 8088.

8089. The 8089 is an I/O processor and two-channel DMA device which is used mainly for disk reads and writes. It has the capability to transfer data from memory to memory, memory to port, port to memory and port to port

8087 (optional). The 8087 mathematics co-processor is used in parallel with the 8086. Full floating-point and intrinsic arithmetic is supported and the 8087 has its own instruction set which in effect extends the instruction set of the 8086.

Communications Handling

The Apricot has two peripheral ports, a parallel printer port driven by the Intel 8255 PIO, and a serial communications port driven by the Zilog Z80 SIO.

Z80 SIO. The Z80 Serial I/O Controller has two independant full-duplex channels with separate control and status lines for modems or other devices. Data rates from 0 to 500K bits/second can be accomplished. Full Synchronous and Asynchronous control is provided, but the current BIOS only supports Asynchronous protocols. In the Async mode, 5, 6, 7 or 8 bits/character and variable stop-bit configuration is available. Also included is Break Generation and detection, parity, overrun and framing error detection. The controller itself also has interfacing for daisy-chain interrupt vectoring without external logic.

8255A-5 PIO. The 8255 Parallel Input/Output interface provides the communications interface between the Apricot and an external printer, via the Centronics connector. It also produces a series of control outputs to various areas of circuitry under software control. Though bi-directional communication is supported in hardware, the current BIOS only uses the standard form of communication.

Sound Generation

The Apricot uses the Texas Instruments SN 76489 sound generator, which consists of the following major components:

1. Three programmable tone generators, each with associated control registers. (Tone generators 1, 2 and 3).
2. A single programmable noise generator with associated control registers.
3. An 8-bit parallel interface for transferring data from the data bus to the control registers.
4. An Audio output buffer stage.

Within the BIOS, the sound generator is used for keyclick (using the noise generator) or the "bell" tone (using one of the tone generator channels).

System Memory

Address (hex)	Utilisation
00000 – 3FFFF	Standard RAM
40000 – EFFFF	RAM Expansion
F0000 – F0FFF	Screen Buffer
F1000 – F7FFF	Unused
F8000 – FBFFF	Unused
FC000 – FFFFF	Bootstrap ROM

The standard System RAM consists of 32 64k x 1 bit DRAMs, arranged in two blocks of 128Kbytes.

The RAM expansion is by way of the Expansion Sockets.

The Screen Buffer RAM is located in two separate Static RAM units.

The boot PROMs are 2 2764 PROMs.

SYSTEM BOARD

CRT Controller

The CRT control circuitry is, centred around the Motorola 6845 CRT Controller. The hardware can be software configured for either standard 80 x 25 text mode, or 800 x 400 pixel graphics.

Floppy Disk Controller

The Floppy Disk Interface is the Western Digital WD2797-02 FDC, a series of buffers, a decoding circuit and the interface connector. Extensive use is made of the 8089 I/O Processor when reading and writing to disks.

Expansion Slots

The Apricot has two expansion slots which are essentially extensions of the standard system address, data and control busses.

Interrupt controller

The Intel 8259A Programmable Interrupt Controller (PIC) forms the interface between the devices capable of generating interrupt requests and the interrupt control line of the 8086 processor.

Timer

The Intel 8253-5 Programmable Interval Timer (TMR) utilizes two clock inputs from a divider circuit to generate a clock output, which is connected to an interrupt request line of the PIC. This provides the Clock Interrupt (see the Software section for more information). Also, two outputs are used to set the baud rates for the Z80 SIO.

Input/Output Space

All the peripheral components located on the board are mapped into the Input/Output space, with address locations allocated as detailed below.

DEVICE	I/O ADDRESS LOCATION (Hexadecimal)
PIC	0, 2
FDC	40, 42, 44, 46
PIO	48, 4A, 4C, 4E
SOUND GENERATOR	50
TIMER	58, 5A, 5C, 5E
SIO	6O, 62, 64, 66
CRTC	68, 6A
8089	70, 72
EXPANSION SLOTS	80 to 1FFF

PROGRAMMABLE INTERRUPT CONTROLLER

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The Intel 8259A Programmable Interrupt Controller (PIC) forms the interface between the devices capable of generating interrupt requests and the interrupt control line of the 8086 processor (CPU on the System Board). Interrupt requests are supplied to the PIC, when the device requires an associated service routine to be undertaken.

The PIC functions as the manager of the interrupt driven processing system; generating an active interrupt to the 8086 processor only when an incoming interrupt request has a higher priority than any interrupt service routine already in operation. On acknowledgement from the CPU that it is able to accept the generated interrupt, the PIC supplies vectoring data to the CPU which acts as a pointer to the required service routine. The vectoring data, the mode of operation of the PIC and the priority of the interrupt requests are set up by software within the Boot PROM and BIOS.

INTERRUPT CONTROLLER

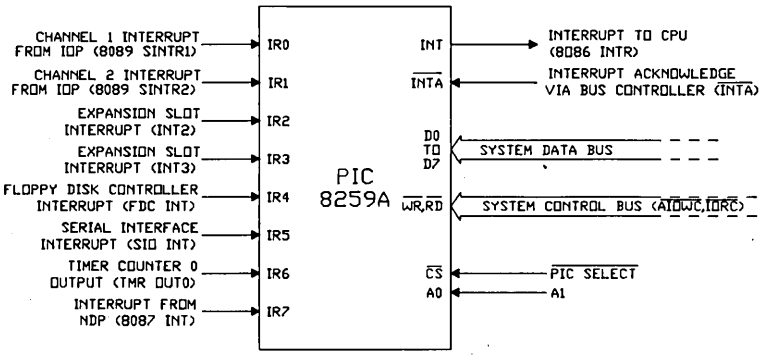


Figure 1. Interrupt Controller block diagram

PIC Pin Definition

IR0 to IR7	Interrupt Request inputs
INT	Interrupt output
$\overline{INTA}$	Interrupt Acknowledge
D0 to D7	Data bus connection
$\overline{RD}$	Read control line
$\overline{WR}$	Write control line
$\overline{CS}$	Chip select input
A0	System address bus input

DESCRIPTION

General

The device connections to the interrupt request lines of the PIC are illustrated on the block diagram on Figure 1 and expanded in more detail below. An active interrupt request is indicated by a logic high state on the appropriate request line.

8089 SINTR1	Interrupt from Input/Output Channel 1 of the 8089 Input Output Processor (IOP).
8089 SINTR2	Interrupt from Input/Output Channel 2 of the 8089 Input Output Processor.
INT2	Interrupt control line wired to both Expansion Slots.
INT3	Interrupt control line wired to both Expansion Slots.
FDC INT	Interrupt from the Floppy Disk Controller.
SIO INT	Interrupt from the SIO of the Serial Interface.
TMR OUT0	Output from Counter 0 of the Programmable Interval Timer (TMR).
8087 INT	Interrupt from the Numeric Data Processor (NDP)

INTERRUPT CONTROLLER

The system software views the PIC as two input/output ports, with each port able to accept and provide a variety of data bytes. The port address locations, defined by the PIC select line and the system bus connection, together with the abbreviations of the data bytes are given below. A detailed explanation of the data can be found in the PROGRAMMING CONSIDERATIONS section following.

Port Address	Data Bytes	Transfer Operation
0	IRR/ISR	Read
0	ICW1/OCW2/OCW3	Write
1	IMR	Read
1	OCW1/ICW2/ICW4	Write

Interrupt Sequence

The interrupt sequence, entered on receipt of a logic high on any of the interrupt request lines of the PIC is described in the following paragraphs. This sequence is the same regardless of the actual PIC interrupt request line being set active.

When one or more of the interrupt request lines are set into the active high state, corresponding bits in an 8-bit internal register within the PIC (the Interrupt Request Register-IRR) are also set. The PIC selects the highest priority bit stored in the IRR, which is not masked by the software, for comparison with bits stored in a second 8-bit register (the Interrupt Service Register-ISR), to determine whether an interrupt should be issued to the CPU.

The Interrupt Service Register contains bits which indicate the interrupt service routines currently being executed by the CPU. If the highest priority unmasked interrupt request is of a higher priority than the highest priority bit set within the ISR, the interrupt control line to the CPU is set into the active state. If the highest priority unmasked interrupt request is not of a higher priority than the highest priority bit within the ISR, no further action takes place. The priority of the interrupt requests is determined by the priority mode selected.

Providing the software controlled interrupt-enable flag of the CPU is enabled, the CPU acknowledges the interrupt, by issuing two Interrupt Acknowledge pulses (INTA pulses) to the PIC, via the 8288 Bus Controller. The first INTA pulse latches the highest priority interrupt request from the IRR through to the ISR. The second INTA pulse enables vectoring data associated with the highest priority bit within the ISR onto the system data bus.

The vectoring data (designated an interrupt number) is a single software defined byte, which is used by the CPU to specify the address location of the corresponding service routine. On completion of the service routine, the associated ISR bit within the Interrupt Service Register is reset under software control.

Interrupt Masking

Masking any of the interrupt requests is achieved under software control by utilising a third internal 8-bit register located within the PIC, known as the Interrupt Mask Register (IMR). The 8-bits within the IMR directly correspond to the 8-bits within the Interrupt Request Register. Setting an IMR bit prevents the PIC actioning any active state on the associated interrupt request line.

PROGRAMMING CONSIDERATIONS

General

Before the PIC can operate normally within the processing system, it has to be initialized with a series of command words which define the required operating mode, the vectoring data and the initial priority of the interrupt request inputs.

Once initialised, the operation of the PIC can be modified and controlled by a second series of command words which:

(a) Enable the interrupt request lines to be individually masked.
(b) Allow the ISR bits within the Interrupt Service Register to be cleared at the end of an interrupt service routine.

INTERRUPT CONTROLLER

(c) Enable the priority of the interrupt request lines to be changed.
(d) Allow the status of the bits of the three internal registers within the PIC (IRR, ISR and IMR) to be analyzed.

The first series of command words are called Initialization Command Words and the second series, Operational Command Words.

Initialization Command Words

Three Initialization Command Words (ICWs) are required to set the PIC into an initial operating condition. The three words have to be issued in a fixed sequence, and if any changes to the initial operating condition are required, the whole sequence must be reprogrammed. Once initialized; the priority of the interrupt requests are automatically assigned from IR0 (highest priority) through to IR7 (lowest priority); the PIC is able to accept interrupt requests.

The initialization programming sequence is ICW1 first, ICW2 next, and finally ICW4. ICW3 is not required since the system operates using a single PIC.

ICW1 defines two parameters when the PIC is used with an 8086 CPU. These are as follows:

(a) The way the PIC senses an active interrupt request (either edge or level sensitive interrupt request).
(b) Whether there is more than one PIC operating within the processing environment.

Due to the fact that the two interrupt request lines from the Expansion Slots are connected to both Expansion Slots (in effect in wire-ORed fashion), the level sensitive interrupt mode must be adopted. (In the edge sensitive mode, a transition on an interrupt request line from a device connected to one of the Expansion Slots would be undetected by the PIC, if the interrupt request line is already raised to logic high by a device connected to the second Expansion Slot).

Since the PIC has to respond to a level sensitive interrupt request and there is only one PIC, the format of ICW1 is fixed as detailed below. The address location ICW1 is written to, is 00H in the system input/output space.

D7							D0	
x	x	x	1	1	x	1	1	*ICW1*

x indicates that the logic state is immaterial program to zero.

The function of ICW2 is to define a base address for the interrupt number. This is signified by five bits within the control word. The three other bits required to form the whole interrupt number (the 3 least significant bits - LSB) are automatically inserted by the PIC and are dependent on the interrupt request line as detailed below.

IR	T2	T1	T0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

T* = Interrupt Number bit

The base address of the eight interrupt numbers assigned to the PIC is 50H as defined by the ICW2 format below. The I/O address location ICW2 is written to, is 02H.

T7	T6	T5	T4	T3				
0	1	0	1	0	x	x	x	*ICW2*

The interrupt number associated with each interrupt request line is generated by combining the base address (the 5 most significant bits - MSB of the interrupt number) with the bits automatically inserted by the PIC (the 3LSB). This

assigns the following interrupt numbers to the interrupt request lines.

IR	Input	Type Vector
0	IOP SINTR1	50H
1	IOP SINTR2	51H
2	INT2	52H
3	INT3	53H
4	FDC INT	54H
5	SIO INT	55H
6	TMR OUT0	56H
7	NDP INT	57H

The majority of the features provided by ICW4 are for use with systems utilising more than one PIC and therefore not relevant to the single PIC environment operated on the System Board. The command word still has to be issued, to provide the PIC with the following information:

(a) The associated CPU is an 8086 device.
(b) Termination of an interrupt service routine is to be signified by software using an Operational Control Word, and not by the automatic end of interrupt facility (AEOI), available during the hardware interrupt acknowledge cycle. (AEOI is only suitable for systems with interrupts which occur at a predetermined rate).

The required format for ICW4 is as detailed below and is written to the address location 02H in the system input/ output space.

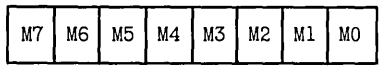
0	0	0	0	0	x	0	1	*ICW4*

Operational Command Words

After initialization the interrupt request lines are all operative with a decreasing order of priority from IR0 through to IR7 (i.e. IR0 highest priority, IR7 lowest). The operation of the PIC can then be further controlled or

modified using any one of three Operational Control Words (OCW1, OCW2 or OCW3). The OCWs are not dependent on each other and can be issued at any time during program execution.

OCW1 provides the facility for enabling/disabling individual interrupt request lines. This is achieved by issuing the control word to the 8-bit Interrupt Mask Register (IMR) within the PIC. The address location of the IMR is 02H within the system input/output space and the format of OCW1 is as follows.



OCW1

Each bit within the IMR directly corresponds to an interrupt request line (M0 of the IMR affects IR0, M1 of the IMR affects IR1, M2 affects IR2 etc.).

Interrupt request lines are disabled (masked) when the corresponding bit within the IMR is set to logic high, and enabled (not masked), when the corresponding bit is set to logic low.

The status of the bits within the mask register can also be read by the programmer at address location 02H within the I/O space.

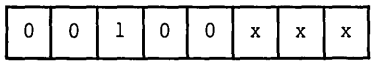
OCW2 is a dual purpose control word, which allows the priority of the interrupt requests assigned during the initialization sequence to be altered, and is also used to inform the PIC that an interrupt service routine is terminating.

The facilities provided by OCW2 for altering the priority of the interrupt requests are not required, since the actual hardware IR connections have been made on the basis of the priority assigned during the initialization routine.

The format of OCW2, used to inform the PIC that the interrupt service routine is at an end (enabling the PIC to reset the associated ISR bit within the Interrupt Service

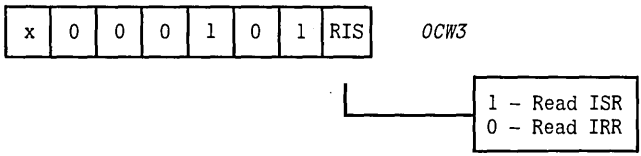
INTERRUPT CONTROLLER

Register), is as detailed below. The address location OCW2 is written to, is 00H within the I/O space.



OCW3 is a command word which provides more facilities to alter the priority of the interrupt request lines, and also allows the status of two internal registers within the PIC, the Interrupt Request Register (IRR) and the Interrupt Service Register (ISR), to be checked. Facilities for altering the priority of the interrupt request lines are not required as explained above.

To read either of the two registers requires two operations to be carried out; a write operation using OCW3 to select the register; followed by a read operation to access the data within the register. The format of OCW3 to select the registers is detailed below. The address location of OCW3 is 00H within the I/O space. The address location, the register data is read from is also 00H within the system input/ output space.



PROGRAMMABLE INTERVAL TIMER

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The Intel 8253-5 Programmable Interval Timer (TMR) is located on the System Board. The timer utilizes two clock inputs from a divider circuit to generate:

(a) A clock output (OUT0), which is connected to an interrupt request line (IR6) of the Interrupt Controller (PIC). The output provides a means of generating accurate timing delays under software control.
(b) Two squarewave clock outputs (OUT1 and OUT2) which can be used to set the baud rates for the RS232C serial interface. The frequencies of the two clock outputs are determined by software.

TIMER

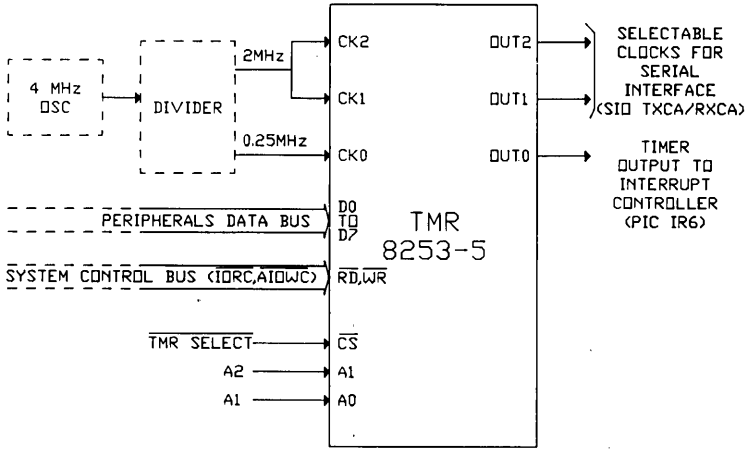


Figure 1.Programmable Interval Timer block diagram

TMR Pin Definition

CK0	Clock input for Counter 0
CK1	Clock input for Counter 1
CK2	Clock input for Counter 2
OUT 0	Output from Counter 0
OUT 1	Output from Counter 1
OUT 2	Output from Counter 2
D0 to D7	Data bus connection
$\overline{RD}$	Read control line
$\overline{WR}$	Write control line
$\overline{CS}$	Chip select input
A0,A1	System address bus inputs

DESCRIPTION

General

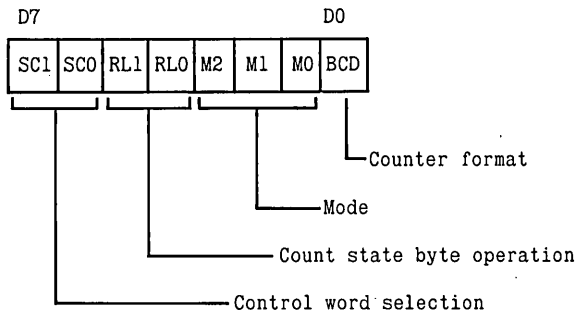
The timer is organized internally as three independent 16-bit counters, each with an associated control word register which determines the operating mode of the 16-bit counter. The counters count down on the negative edge of the respective clock pulse input.

The system software views the three counters and the control word register set as an array of peripheral input/output ports. The port address locations defined by the timer select and system address bus connections are detailed below:

Port	Address	Data
Counter 0	58H	Count state 0
Counter 1	5AH	Count state 1
Counter 2	5CH	Count state 2
Control word register set	5EH	Control word

Data can be written to all four address locations but can be read only from the three counter locations. Each counter has to be initialized with the required mode of operation, the count state format, and the number of bytes in the count state, using the control word, prior to loading the count state. The counters begin counting downwards on completion of the count state load operation. The count state can be one or two bytes. The format of the control word is as follows:

Control Word Format



Control word selection

SC1	SC2	
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Invalid

TIMER

Count state byte operation

RL1	RL2	
0	0	Latch count state into storage register
0	1	Read/load most significant byte only
1	0	Read/load least significant byte only
1	1	Read/load least significant byte followed by most significant byte.

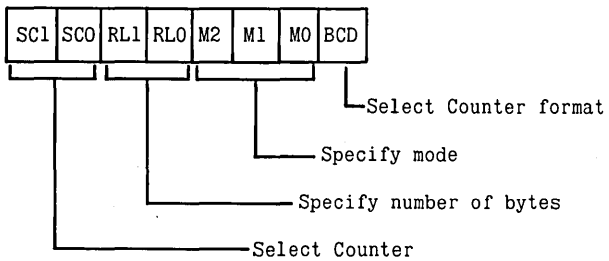
Mode

M2	M1	M0		
0	0	0	Mode 0	Interrupt on terminal count
0	0	1	Mode 1	Programmable one shot
x	1	0	Mode 2	Rate generator
x	1	1	Mode 3	Squarewave rate generator
1	0	0	Mode 4	Software triggered strobe
1	0	1	Mode 5	Hardware triggered strobe

Counter format

BCD	
0	Binary Counter
1	BCD Counter

Initializing a selected counter requires the control word to be assembled as detailed below and then written to the control word register set address location.

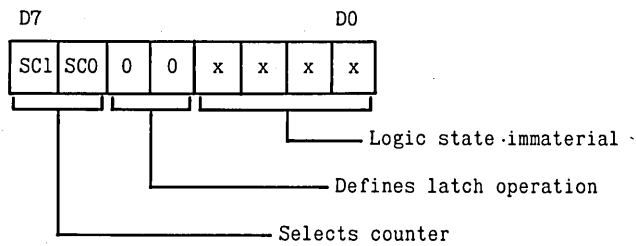


Once initialized, the counter can be loaded with the count state at any time following initialization. The only constraint is that when more than one byte is to be loaded, the bytes must be loaded into the counter address location in the order specified by the control word. Selecting the

condition for loading one byte only into the counter, automatically sets the other byte to zero count state. The counter begins decrementing after the full count value has been loaded.

Two methods are available to read the count values reached by the counters, whilst the counters are counting down. Both methods have no affect on the operation of the counters. The first method is achieved by reading the data stored at the selected counter address location using two consecutive read operations. The first read operation returns the least significant byte of the count state and the second, the most significant byte.

The second method ensures that a stable count state reading is obtained by utilising the control word to latch the count state into a storage register associated with each counter. The count state is then obtained in the same manner as the first method. (By performing two consecutive read operations to access the stored count values at the counter address location). To latch the count state into the storage register, the control word is written into the address location of the control word register set, with the format as detailed below.



Counter 0

For the output of Counter 0 (OUT 0) to be used as an interrupt request line to the Interrupt Controller, the counter must be set to operate in Mode 0. When the counter is initialized, output (OUT 0) is set to logic low. After the counter is loaded, the counter begins to count down as illustrated in the timing diagram, Figure 2 below.

TIMER

The count state is decremented on the negative edge of each 0.25 MHz clock pulse. On reaching zero count state, the counter output is set to logic high. This condition remains static until reset, (either by re-initializing the counter or by loading a new value into the counter). The counter continues to decrement after the zero count state is reached,starting from the maximum count state of the counter, until reset. (2^{16} for a binary counter, 10^4 for a BCD counter). This feature allows the software to determine the exact time of the interrupt request by reading the Counter 0 count state and performing a simple calculation.

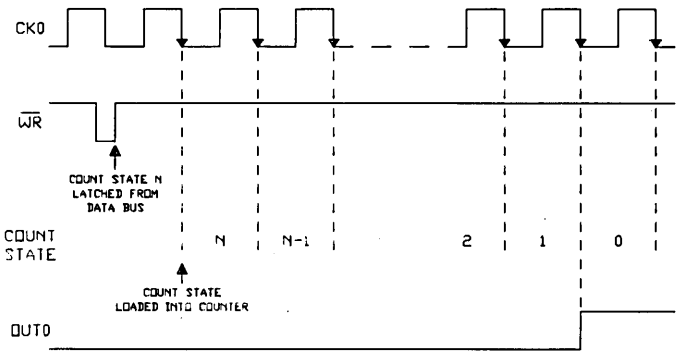


Figure 2. Mode 0 timing diagram.

Counter 1 and 2

Counter 1 and Counter 2 are set to operate in Mode 3, to produce a squarewave output for use by the RS232C serial interface. In Mode 3, a true squarewave is only produced by programming the counter with an even count state. Under this condition, the counter remains at logic high for one half of the count state, and at logic low for the second half. The counter is decreased by two on the negative edge of each 2MHz clock pulse as illustrated in the timing diagram, Figure 3.

When the counter reaches zero count state, the original count value is reloaded into the counter and the process repeated. This produces a squarewave output with a frequency equivalent to the input clock frequency (2 MHz) divided by the count state, and a 50% duty cycle.

When the counter is programmed to operate in Mode 3 using an odd count value, the frequency of the output is equivalent to 2 MHz divided by the count value as for an even count value, but the output waveform is asymmetrical. The asymmetry being more pronounced for low count values (high output frequencies). If the value is odd, the counter output remains at logic high for (N+1)/2 clock pulses and remains at logic low for (N-1)/2 clock pulses as illustrated in the timing diagram Figure 3, where N represents the count value.

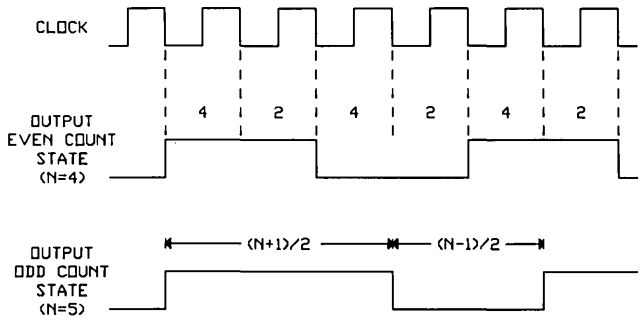


Figure 3.Mode 3 timing diagram

TIMER

Baud Rates

The count values for programming Counter 1/Counter 2 to produce some of the commonly used baud rates , are detailed in tabular format below. These values only apply to asynchronous communications, where the outputs supplied from the counters are divided by 16 internally within the SIO prior to setting the baud rate.

Equivalent Baud Rate (Bauds)	Count Value N (Hex.)
50	09C0
75	0683
110	046F
134.5	03A0
300	01A1
600	00D0
1200	0068
1800	0045
2400	0034
3600	0023
4800	001A
7200	0011
9600	000D
19200	0007

CRT CONTROL

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The CRT Control circuitry is located on the System Board and provides the interface for displaying text or graphics on the screen of a high resolution monochrome display unit.

In the text mode of operation (alphanumerics), a display area of 80 characters by 25 lines is available, using a character cell of 10 pixels wide by 16 pixels high. The characters for display are accessed from a character font loaded by the BIOS into the System RAM. Each character can be modified using a series of programmable display

attributes. These allow screen characters to be in either normal or reverse video with the addition of any combination of the following display features, as required:

(a) Highlight (High intensity characters).
(b) Strikethrough (Horizontal line through the centre of the character).
(c) Underline. (Horizontal line under the character).

In the graphics mode, the screen is memory-mapped directly into the System RAM (i.e. each individual pixel on the screen corresponds to a bit written into the System RAM). The resolution of the screen available in this mode is 800 pixels wide by 400 pixels high, requiring just under 40 Kbytes (800 x 400 bits) of System memory to map every pixel.

All information displayed on the screen is generated using a raster scan imaging system, with the CRT control circuitry providing the video drive signal, and the synchronising pulses to control the movement of the CRT electron beam.

Connections to the Display Unit are supplied via a 9-pin male D-type connector located at the rear of the System Unit.

DESCRIPTION

General

A simplified view of the CRT control circuitry is illustrated in the block diagram Figure 1, which is used in the following paragraphs to provide an introduction to the principles involved in generating the video signal for the two different display modes. Subsequent sections of the chapter explain the operation of the circuitry in more detail.

The block diagram in effect, presents the view of the circuitry as seen from the CRT Controller (CRTC), and is simplified by the omission of:

(a) The system data/address/control bus connections to the Screen and System RAM.

(b) The memory contention circuitry which determines when the CRTC is allowed access to the Screen and System RAM and when the processors have access.
(c) The peripherals data bus, system address and control bus connections to the CRTC.
(d) A few ancillary control connections.

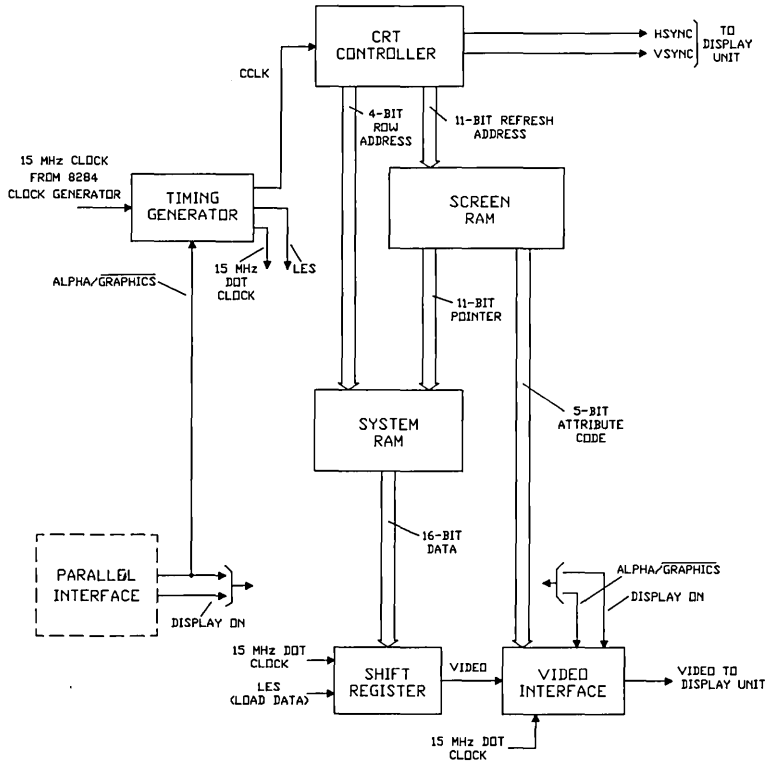


Figure 1. CRT control circuitry block diagram

The display raster on the screen of the Display Unit is controlled by the two outputs from the CRTC, HSync and VSync. Hsync controls the horizontal movement of the electron beam across the screen. Vsync controls the vertical retrace of the beam back to the top of the screen at the end of the display period.

CRT CONTROL

As the electron beam scans across the screen (i.e. during the active line period), the CRT circuit supplies a serial video signal to the Display Unit. The video signal modulates the electron beam to produce the display information on the screen.

The method of generating the video signal is detailed in the following paragraphs.

The CRTC generates addresses (refresh addresses and raster addresses) during the active scan line period, which access data stored in the two areas of RAM. The same sequence of addresses are automatically repeated at the screen refresh rate (i.e. frame rate).

The CRTC addresses also map the text character/graphics cell position on the screen. Each text character on the screen is bounded within a 10 by 16 pixel character cell. A graphics cell is defined as an area 16 pixels wide by 16 pixels high on the screen.

The first refresh address generated by the CRTC at the start of each new frame period (i.e. after the vertical retrace period), corresponds to the first character/graphics cell position on the screen; the second refresh address corresponds to the second cell position on the screen, etc., as illustrated in Figures 2 and 3.

The refresh address changes every cell period during the active scan line period, but repeats the same sequence of addresses over 16 adjacent scan lines. The raster address changes every scan line but repeats the same sequence every 16 scan lines and defines the row of each character/graphics cell displayed on the screen.

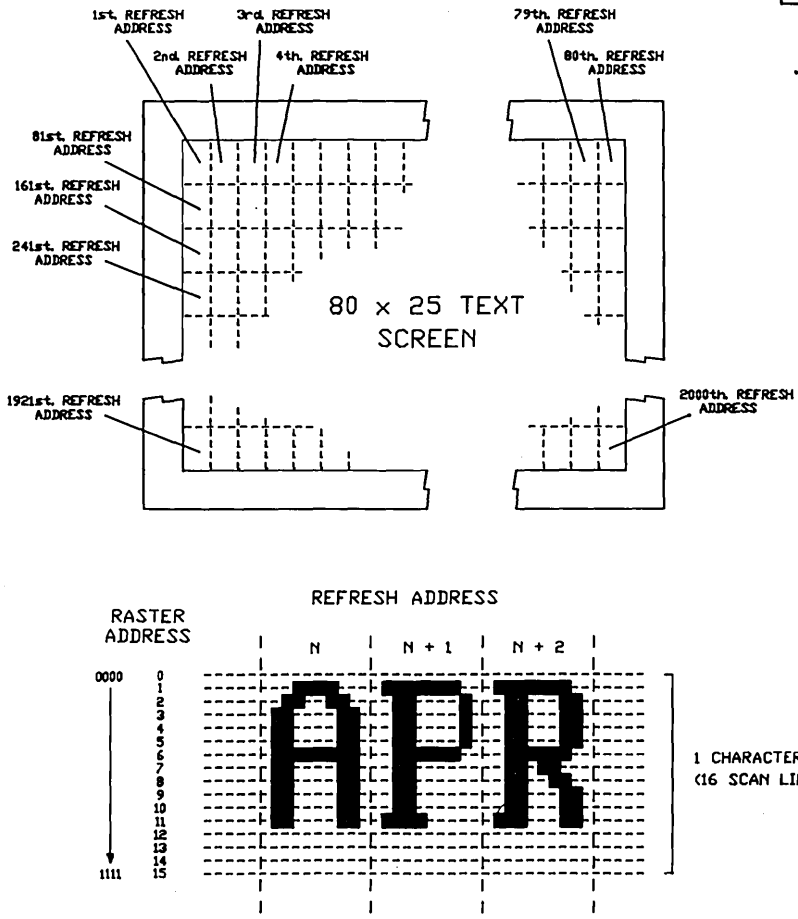


Figure 2. Text Screen

CRT CONTROL

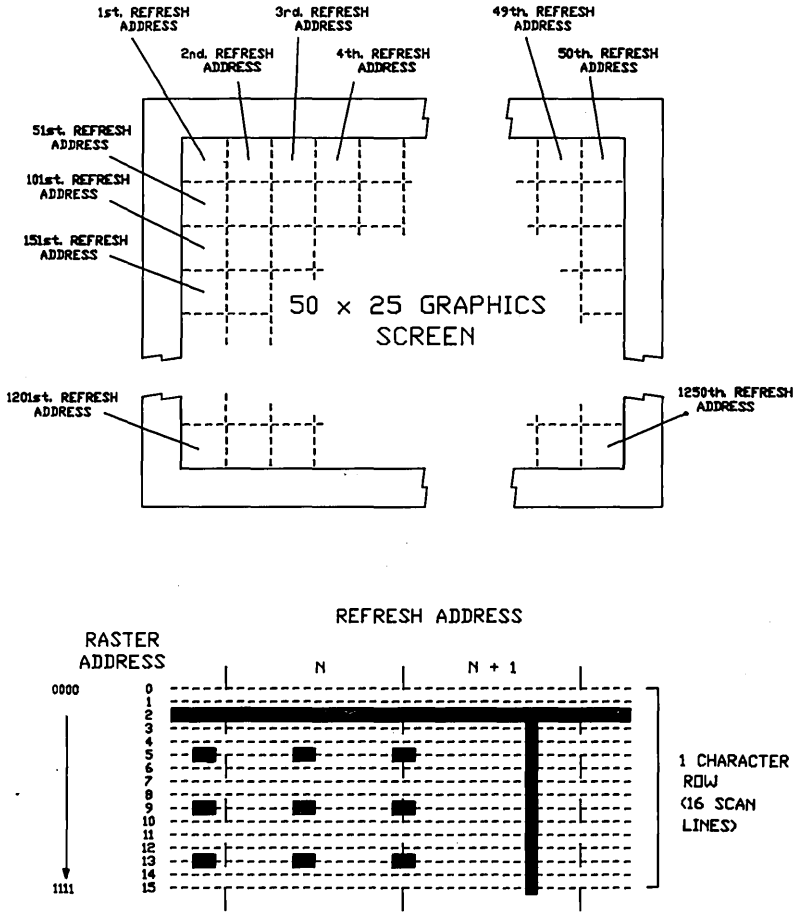
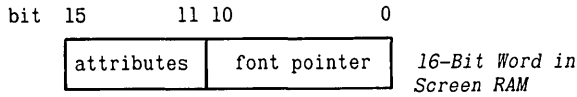


Figure 3. Graphics Screen

The data accessed from the Screen RAM and System RAM is dependent on the mode of operation selected.

In text mode, the Screen RAM is programmed with 16-bit words, each word consisting of a 11-bit font pointer and a 5-bit attribute code. The font pointer specifies the character for display and is combined with the raster address from the CRTC to access each row of the character stored in the System RAM. The attribute code is supplied to the video interface to determine the display attributes of the character accessed from the System RAM.



font pointer + raster address = font cell address

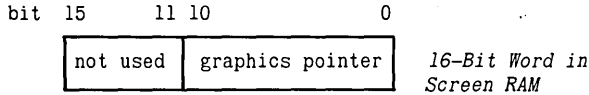
The character data in the System RAM is stored in a character font consisting of 256 characters. Each character consists of 32 bytes in contiguous memory locations and is organised as 16 rows of 16-bit words (defined as a font cell). Within each row of the font cell, only the 10 LSB comprise the actual character. The 6 MSB are not displayed on the screen.

The 10 character bits from each row are latched into the video shift register, converted into serial format and then shifted out to the video interface circuitry at the dot clock frequency.

The video interface modifies the serial data according to the attribute code (e.g. inverts the serial data stream, if reverse video is selected), and then supplies the video signal to the Display Unit.

In graphics mode, the Screen RAM is programmed with 16-bit words, of which the 11 LSB are a graphics pointer. The remaining 5-bits of each word are redundant, since attributes are not used in the graphics mode. The graphics pointer is combined with the raster address from the CRTC to access each row of the 16-bit graphics cell image written into the System RAM.

CRT CONTROL



graphics pointer + *raster address* = *graphics cell image address*

Each graphics cell image consists of 32 bytes of data in contiguous memory locations and is organised as 16 rows of 16-bit words. Each pixel on the screen is mapped directly by a corresponding bit within each 16-bit row of the graphics cell image.

The full 16-bits from each row are latched into the video shift register, converted into serial format and then shifted out to the video interface at the dot clock frequency. The video interface then supplies the video signal to the Display Unit.

Unlike the text mode, where the Screen RAM has to be updated with a new font pointer and attribute bits to specify a different character for display on the screen, the graphics pointers remain unchanged once programmed into the Screen RAM. Changing the display on the screen in graphics mode is achieved by updating the image of the graphics cell stored in the System RAM.

Mode Selection

Whether the circuitry operates in the text (alphanumerics) or graphics mode depends on:

(a) The logic state on the mode select line alpha/ graphics from the Parallel Interface, which is directly under software control.
(b) The initialisation of the CRTC.
(c) The data programmed into the Screen RAM.
(d) The data programmed into the System RAM.

The mode select line affects two different areas of the circuitry, the timing generator and the video interface. The effect produced on the timing generator is to change the frequency of the two signals, CCLK (character clock) supplied to the CRTC, and LES (load data) supplied to the shift register.

CCLK is the basic timing signal used by the CRTC and defines the rate at which the refresh addresses change, and thus the character width on the screen. The clock frequency is derived from the 15 MHz dot clock. In the text mode, the timing generator divides the dot clock by 10 (10 pixel width text cell: refresh addresses change at a frequency of 1.5 MHz during the active scan period). In the graphics mode, the dot clock is divided by 16 (16 pixel width graphics cell; refresh addresses change at a frequency of 15/16 MHz during the active scan period).

LES controls the loading of the 16-bit data from the System RAM into the shift register. The shift register moves the data out of the register at the 15 MHz dot clock frequency. In the text mode, the frequency of LES is 1.5 MHz, so that every time 10 bits are shifted out of the register a new 16-bit character row is loaded in (In the text mode only the 10 LSB of the 16-bit data are used for display). In the graphics mode, the frequency of LES is 15/16 MHz, so that the full 16-bits of data are shifted out of the shift register, before a new 16-bit graphics cell row is loaded in.

On the video interface, the mode control line determines whether the attribute bits and cursor control signal are enabled or disabled. The attribute bits and cursor control signal are enabled in the text mode, disabled in graphics.

CRT CONTROL

CRTC DETAIL

General

The basis of the CRT control circuit is the Motorola MC6845 CRT Controller (CRTC). Once initialised with operational parameters for the Display Unit (scan line rate, frame rate, mode, etc.), the CRTC:

(a) Automatically and repetitively generates addresses which access data from the System Ram to refresh the screen of the display.
(b) Generates horizontal (line) and vertical (field) sync pulses for controlling the scanning of the electron beam across the screen.
(c) Generates an output signal which defines the active display period of the screen.
(d) Allows the software to control the movement of a cursor (in the text mode only) and also scroll the screen.

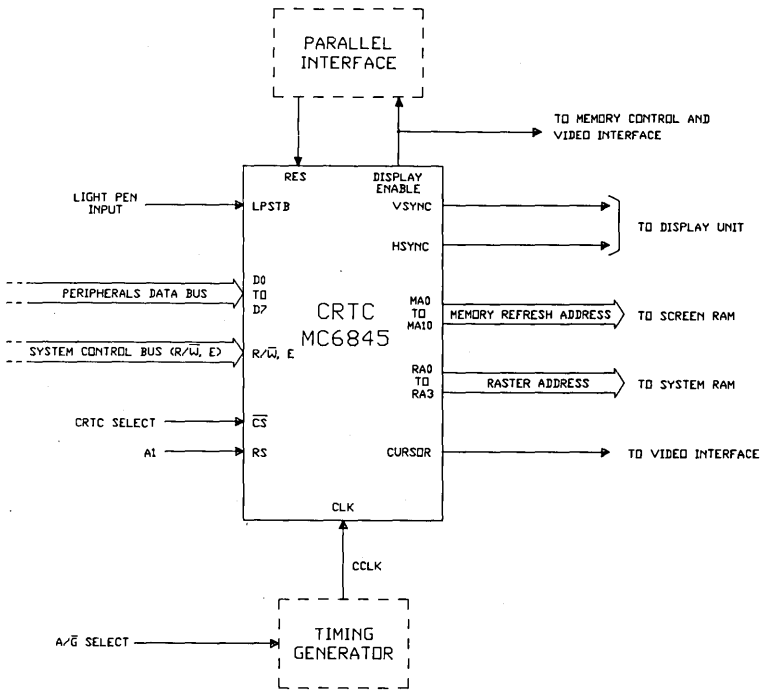


Figure 4. CRTC block diagram

The refresh addresses generated by the CRTC are of two types, refresh address lines (MA0 to MA10), and raster address lines (RA0 to RA4). The refresh address lines access the data stored in the Screen RAM (the font cell pointer plus the character attributes in the text mode/the graphics cell pointer in graphics mode).

The raster address lines are combined with the pointer to access the System RAM to select the row of the font/ graphics cell, for display.

The CRTC is programmed to operate in a raster scan mode defined as interlace sync and video. In this mode, the display raster on the screen is generated using two fields, an even field and an odd field, consisting of even and odd scan lines respectively. (Similar to the method employed within a standard domestic television). Half the rows of each font/graphics cell are displayed in each field.

In the even field, as the electron beam sweeps out the even scan lines, the even rows of a font/graphics cell are accessed for displaying on the screen. In the odd field, as the electron beam sweeps out the odd scan lines, the odd rows of a font/graphics cell are accesssed for displaying on the screen. Each frame of video information is thus defined by two consecutive fields.

The movement of the electron beam across the screen is controlled by the two outputs from the CRTC, Hsync and Vsync. Hsync controls the horizontal sweep of the beam across the screen during each field (line scan rate). Vsync controls the vertical retrace of the beam back to the top of the screen at the end of each field (field rate).

The active display period on the screen is indicated by the Display Enable (DE) output, which is supplied to the video interface, memory control circuits, and the Parallel Interface. On the video interface, the DE signal enables the video signal output during the active periods, and inhibits the output during line and field flyback periods.

The DE output is also used as an enable/inhibit signal for the memory control circuits. During the active display

CRT CONTROL

period, the screen address (cell pointer combined with raster address lines) and the processors are allowed access to the System memory. During the flyback periods, the processors are still allowed access, but the screen address is inhibited and the memory control refresh counter enabled. The refresh counter provides addresses to perform the hardware refresh of the System memory dynamic RAMs.

Supplying the DE signal to the Parallel Interface enables the processors to detect the active and flyback periods as required.

Cursor movement in the text mode, is controlled by changing the contents of a pair of registers within the CRTC, which specify the cursor position on the screen. The cursor is displayed by the CRTC, supplying a cursor display signal (cursor) to the video interface as the electron beam sweeps the required cursor character position on the screen. In graphics mode, the cursor display signal is inhibited at the video interface.

The CRTC contains a pair of registers which define the height of the text cursor, and also provide an optional facility to allow the cursor to blink at one of two different rates. The width of the text cursor is fixed and corresponds to the width of a character cell (10 pixels).

Scrolling of the screen is achieved by writing a different start address into another pair of registers located internally within the CRTC. The start address is the first refresh memory address to be supplied to the Screen RAM, at the start of each field period.

The CRTC also provides a facility to connect a light pen with its associated control electronics, for detecting the electron beam as it scans across the screen.

NOTE: The standard 9 inch green on black Apricot Display Unit utilises a long persistence phosphor (P39), which may not support the use of a light pen interface.

Internally the CRTC consists of a collection of registers, counters and comparators, which time all the logic activities (generation of addresses, sync. pulses, etc.) as the

interlaced raster scan proceeds. The basic clock frequency used by the counters is CCLk (Character Clock) supplied from the timing generator circuit.

The period of CCLK is dependent on the mode selected, and corresponds to the display time of the character cell in each scan line. CCLK is derived from the 15 MHz dot clock. In the text mode, CCLK is produced by dividing the dot clock by 10 (corresponding to 10 pixels per character cell). In the graphics mode, CCLK is produced by dividing the dot clock by 16 (corresponding to 16 pixels per graphics cell).

Other registers within the CRTC are programmed with data for controlling the operation of the counters to determine various parameters for the Display Unit (e.g. active line period, vertical retrace period etc.). A definition of each of the registers is provided below.

Register Description

The CRTC contains 19 registers. 10 of the registers are programmed with operational parameters to match the Display Unit to the selected screen format (text or graphics). Four of the remaining registers are involved with the text cursor. Two of these registers specify the format of the cursor; the other two define the cursor position on the screen. The start address accounts for another two. Two registers are used to store the position of a light pen. The remaining register is programmed with a pointer which specifies the address location of the other registers. A summary of the registers is provided below. This is followed by a detailed description of each individual register.

AR	Address Register	R9	Max scan line address
R0	Horizontal Total	R10	Cursor format (start)
R1	Horizontal Displayed	R11	Cursor format (end)
R2	HSync Position	R12	Start Address (H)
R3	Hsync Width	R13	Start Address (L)
R4	Vertical Total	R14	Cursor Position (H)
R5	V. Total Adjust	R15	Cursor Position (L)
R6	Vertical Displayed	R16	Light Pen (H)
R7	VSync Position	R17	LightPen(L)
R8	InterlaceMode		

CRT CONTROL

Address Register. 5-bit register located at 68H in the system I/O space and can be written to only. Acts as the pointer for the other control registers. Prior to any data transfer between the processors and any of the 18 control registers, the source/destination register has to be defined by programming the address register with the address pointer. The pointer for each control register is the binary equivalent of the decimal "R" number, e.g. Interlace Mode control register R8 is specified by programming the address register with 08H.

Horizontal Total Register (R0). The contents of this register determine the line frequency of the HSync output and is specified in CCLK periods. The actual number is equivalent to the total number of character periods in the active scan line time and the line retrace period minus one (active + retrace - 1 character periods).

Horizontal Displayed Register (R1). This register specifies the number of characters displayed on each scan line.

HSync Position Register (R2). The contents of this register specify the position of the horizontal sync pulse relative to the start of the active scan line period. The value is programmed in CCLK periods. The effect of increasing the value in R2 is to shift all characters displayed on the screen to the left. Decreasing the value shifts the whole character display to the right.

HSync Width Register (R3). The contents of R3 set the width of the Hsync pulse in units of the character clock period.

Vertical Total Register (R4). This register with R5 define the frequency of the VSync pulses. R4 is programmed with a value, in character row periods (i.e. 16 scan line periods), just less than the desired frequency. The fine adjustment to achieve the exact VSync frequency is provided by R5. The actual value of R4 is one less than the desired number of character row periods.

Vertical Total Adjust Register (R5). This register provides the fine control of the frequency of VSync and is programmed in scan line periods.

Vertical Displayed Register (R6). The contents of this register determine the number of displayed character rows on the screen.

VSync Position Register (R7). The contents of this register specify the position of the vertical sync pulse relative to the start of the first active scan line period. The value is programmed in Character row periods. Increasing the value in R7 shifts the whole display on the screen upwards. Decreasing the value shifts the display downwards.

Interlace Mode Register (R8). This register selects the raster scan mode. A choice of non-interlace (00H), interlace (01H), or interlace sync and video (03H) modes are available. In the non-interlace mode, the scan lines are refreshed at the frame rate. In the two interlace modes, the frame is divided into two fields, an odd field and an even field. In the interlace mode, the same information is displayed in both fields. In interlace sync and video, the even lines of a character are displayed in the even field, and the odd lines in the odd field.

Max Scan Line Address Register (R9). This register determines the number of scan lines per character row and is programmed with a number one less than the desired value.

Cursor Format (Start) Register (R10). The contents of R10 with the contents of R11 specify the height and position of the text cursor within the boundaries of the character cell. R10 also selects the cursor display mode. The lower 5 bits of R10 specify the start address of the cursor within the character cell and is a value corresponding to a raster row address. Bits 6 and 7 determine the cursor display mode as detailed below.

bit 7	bit 6	Cursor Display Mode
0	0	Non-blinking
0	1	Non-display
1	0	Blinking, 1/16th field rate
1	1	Blinking, 1/32nd field rate

CRT CONTROL

Cursor Format (End) Register (R11). This register sets the end address of the cursor within the character cell and is a value corresponding to a raster row address either higher or the same value as the cursor format start address.

Start Address Register (R12). The contents of this register with the contents of R13 define the first refresh address of each new frame period. R12 is programmed with the 6 most significant bits of a 14-bit word. The 8 least significant bits are provided by R13.

Start Address Register (R13). See R12 above.

Cursor Position Register (R14). The contents of this register with the contents of R15 define the position of the text cursor on the screen. R14 is programmed with the 6 most significant bits of a 14-bit word, corresponding to a refresh address value. The 8 least significant bits are provided by R15. Both registers R13 and R14 can be written to and read from.

Cursor Position Register (R15). See R14 on the previous page.

Light Pen Register (R16). When a positive edge occurs on the LPSTB input of the CRTC, the current refresh address is latched into the two registers R16 and R17, to define the position of the light pen on the screen. R16 stores the 6 most significant bits of the 14-bit address and R17 the 8 least significant bits. The position of the light pen is then determined by reading the contents of these registers.

Light Pen Register (R17). See R16 above.

Initialising the CRTC

The values supplied to the control registers to initialise the CRTC to operate with the standard 9 inch monochrome Display Unit for the two different display modes are detailed below. The control registers are programmed using two separate write operations. The first write is to address location 68H in the system input/output space (the address register) to specify the control register pointer address. The

second write is to address location 6AH in the system input/output space, which loads the value into the control register, specified by the pointer.

Register	Text (Hex)	Graphics (Hex)	Register	Text (Hex)	Graphics (Hex)
R0	5E	3B	R8	03	03
R1	50	32	R9	0E	0E
R2	4F	30	R10	00	00
R3	0C	0C	R11	0F	0F
R4	19	19	R12	00	00
R5	0A	0A	R13	00	00
R6	19	19	R14	00	00
R7	19	19	R15	00	00

CRTC Connections

D0 to D7	Data bus. Used to transfer data between the internal registers of the CRTC and the processors.
R/$\overline{W}$	Read/write control input connected to the DT/$\overline{R}$ control line of the system control bus. Used in conjunction with $\overline{CS}$, RS and E to control data transfers between the CRTC and the processors. R/$\overline{W}$ determines the direction of data transfer; logic high from the CRTC (read), logic low to the CRTC (write).
$\overline{CS}$	Chip Select. Address input. Active state, logic low. When active indicates that the CRTC is selected for a data transfer operation.
RS	Register Select. Input connected to A1 of the system address bus. Used to select either the address register (logic low) or one of the eighteen control registers (logic high).

continued ....

CRT CONTROL

E	Enable input. Data strobe signal for latching data to/from the peripherals data bus. Logic high to low transition at the end of the second processor clock cycle after the address is valid.
RES	Input signal to reset the CRTC. Active state, logic low. Generated by the Parallel Interface. When active, all counters within the CRTC are cleared and all outputs are forced to logic low. The contents of the registers are unaffected.
CCLK	Character clock. Input signal from the timing circuit, with a 50% duty cycle, derived from the 15 MHz dot clock. Used as the basic CRTC timing signal. In text mode, the frequency is 1.5 MHz. In graphics, 937.5 kHz.
MA0 to MA10	Refresh memory address signals. Address outputs, which change every character clock period during the active scan line period to access the Screen RAM and thus refresh the CRT screen.
RA0 to RA3	Raster address signals. Address lines to the System RAM to select a row within a character font cell to be displayed in text mode, a row within the graphics cell image in the graphics mode.
VSYNC	Sync pulses supplied to the Display Unit to control the retrace of the electron beam back to the top of the screen at the end of each active field period. Pulses are generated at a rate of approximately 72 Hz.
HSYNC	Sync pulses supplied to the Display Unit to control the horizontal sweep of the beam across the screen during the active field period. The pulses are generated at a scan rate of 15.79 kHz.

continued ....

Display Enable	Output generated by the CRTC which defines the active scan line/retrace periods of the electron beam. Logic high indicates the active periods, logic low the retrace periods. Supplied to the Parallel Interface, the video interface and the memory control circuits.
Cursor	Output signal supplied to the video interface, used to control the cursor on the screen in the text mode. Every time the CRTC produces a refresh address which matches the cursor address programmed into control registers R14 and R15, the cursor output is set to logic high, producing a positive going pulse for one CCLK period.
LPSTB	Light Pen Strobe input. Connected to a Molex connector (LP) on the System Board. Allows a light pen and associated control circuitry or similar device to be connected to the CRTC. Every time a positive edge is generated on this input, the current refresh address is latched into control registers R16 and R17.

SCREEN RAM

General

The Screen RAM consists of two 2k x 8 bit static RAMs arranged in a 2k x 16 bit block and occupies 4 Kbytes in the system memory space at address location F0000H to FOFFFH (2048 16-bit words). The RAM is dual port memory, being able to be accessed by both the processors and the CRTC. Access to the RAM is controlled by a memory contention circuit. The memory contention circuit guarantees the CRTC access to the RAM once every character clock period to refresh the screen. Wait states are automatically added to the processor memory cycle, if the processors attempt to access the RAM, during a CRTC access (1 to 3 wait states in text mode, 1 to 5 wait states in graphics).

CRT CONTROL

The 16-bit words programmed into the Screen RAM are accessed by the refresh addresses from the CRTC. These addresses are incremented every character clock period during the active scan line period but repeat the same sequence of addresses over sixteen adjacent scan lines (as illustrated in Figures 2 and 3). The CRTC automatically cycles through the same sequence of addresses each screen refresh period, unless the CRTC start address is updated.

In the text mode, the CRTC is programmed to cycle through 2000 different refresh addresses to map the whole text screen of 25 lines of 80 characters. These repetitively access the 16-bit words programmed into the first 2000 word address locations in the Screen RAM.

In the graphics mode, the CRTC is programmed to cycle through 1250 different refresh addresses to map the whole graphics screen of 50 columns by 25 rows. These repetitively access the 16-bit words programmed into the first 1250 word address locations in the Screen RAM.

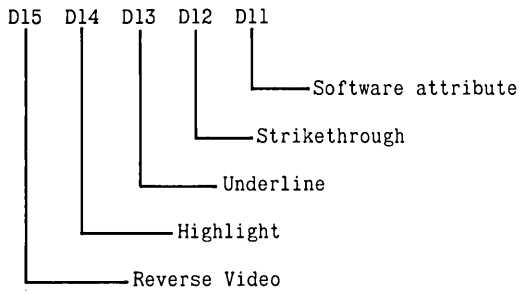
The 16-bit data programmed into the Screen RAM is dependent on the selected mode.

Text Mode

In the text mode, the Screen RAM is programmed with 16-bit words, each word consisting of a 11-bit font pointer and a 5-bit attribute code. The font pointer is used to select characters stored in the System RAM. The attribute code is supplied to the video interface to determine the display attributes of the character accessed from the font. Changing a character on the screen is achieved by writing a different 16-bit word into the appropriate address location of the Screen RAM.

The 11 least significant bits of the 16-bit word (D0 to D10) in the Screen RAM are the font pointer. The 11 bits of the font pointer are combined with the raster address lines from the CRTC to form a word address, to access a character row in the System RAM (The 11-bits are shifted 5 binary places in significance to bits 5 to 16 and combined with the raster

address, see System RAM description below). The attribute bits are defined as follows:



Software Attribute (D11). This bit is for use by applications programs to mark screen characters for various display functions, e.g. mark the beginning of a line, end of a block of text, etc. The bit is not used for display generation functions.

Strikethrough (D12). This bit works in conjunction with one of the non-displayed bits within a font cell. If the strikethrough bit is set; a horizontal line is superimposed through the character, at a position determined by bit 14 in the font cell (see System RAM description).

Underline (D13). This bit works in conjunction with one of the non-displayed bits within a font cell. If the underline bit is set, a horizontal line is drawn the full width of the character cell, underneath the character at a height determined by bit 15 in the font cell (see System RAM description).

Highlight (D14). If this bit is set, the character is displayed in high intensity on the screen. If not set, the character is displayed in normal intensity.

Reverse Video (D15). If this bit is set, the character, underline and strikethrough are all displayed in reverse video, (i.e. black character on a green background). If not set, the character is displayed in normal video (green character on a black background).

CRT CONTROL

The position of the selected character on the screen is determined by two factors, the address location of the 16-bit word in the Screen RAM and the start address programmed into the CRTC. If the CRTC start address is programmed to 0000H, the first character displayed on the screen is the one selected by the font pointer stored at word address location F0000H in the Screen RAM; the second character displayed on the screen, the one selected by the font pointer stored at word address location F0002H; the third character, the one selected by the font pointer at F0004H, etc.

If the CRTC start address registers are programmed to a different value than 0000H, the first character displayed on the screen is selected by the pointer stored at the address in the Screen RAM corresponding to F0000H + twice the CRTC start address; the second character displayed, selected by pointer stored at the Screen address F0000 + twice the CRTC start address + 2, etc.

For example, if the start address is programmed to 0050H (decimal 80), the first character displayed on the screen is specified by the pointer, stored at Screen RAM address F00A0H, the second character is specified by the pointer stored at Screen RAM address F00A2H, etc. Due to the cyclical nature of the generation of the refresh addresses from the CRTC, the last character (the 2000th.) displayed on the screen, is specified by the pointer stored at Screen RAM address F009EH.

This example illustrates how scrolling the screen upwards by one text line is achieved. If the CRTC start address is incremented by 50H, the text line originally the second line is moved to the top of the screen, and the text line originally the first text line is moved to the bottom of the screen. New parameters (pointers and attribute bits) then can be issued to the original first text line address locations, as required.

Graphics Mode

In the graphics mode the first 1250 word address locations in the Screen RAM are initialised with 16 bit words, of which the 11 least significant bits (LSB) are the 11-bit graphics pointer. The 5 most significant bits (MSB) of the

word are redundant. The graphics pointer is used to select graphics cell images programmed into the system RAM. The values of the 11-bit pointers never need to be changed and since the CRTC start address is also never changed, the pointers always access the same graphics cell image as corresponding pixels are mapped onto the screen.

SYSTEM RAM UTILISATION

General

The utilisation of the System RAM by the CRT control circuitry is dependent on the selected mode. In text mode, the BIOS allocates space in the System RAM for three character fonts, a default font and two user specified fonts. Each font has space for 256 characters. The default font loaded by the BIOS for UK use, is a slightly modified ASCII SINTL 01 character set (strikethrough bit added). This occupies 8 Kbytes of system memory from address location 0800H to 27FFH. The two user font areas occupy the next 16 Kbytes of system memory (2800H to 47FFH and 4800H to 67FFH respectively).

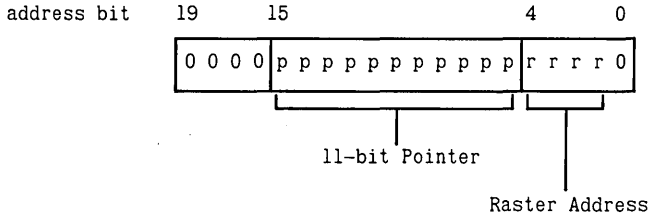
In the graphics mode, 40 Kbytes of the system memory are required to map every pixel on the 800 x 400 resolution screen. The System RAM allocated to the graphics screen image is from 2800H to C440H, which means the two user fonts are always overwritten when graphics is selected.

The CRT control circuitry is guaranteed access to the System RAM, once every character clock period during active display time to refresh the screen. Contention between processor access to the System RAM and CRT control access is managed by the system memory control circuits. The control circuits automatically add wait state(s) to a processor memory cycle if the processors attempt to access the RAM, during the CRT control access period. The number of wait states inserted for text is from 1 to 3, for graphics 1 to 5.

The address generated by the CRT control circuit to access the System RAM is produced by offsetting the 11-bit pointer from the Screen RAM by 5 significant places and then combining the result with the 4-bit raster address, as

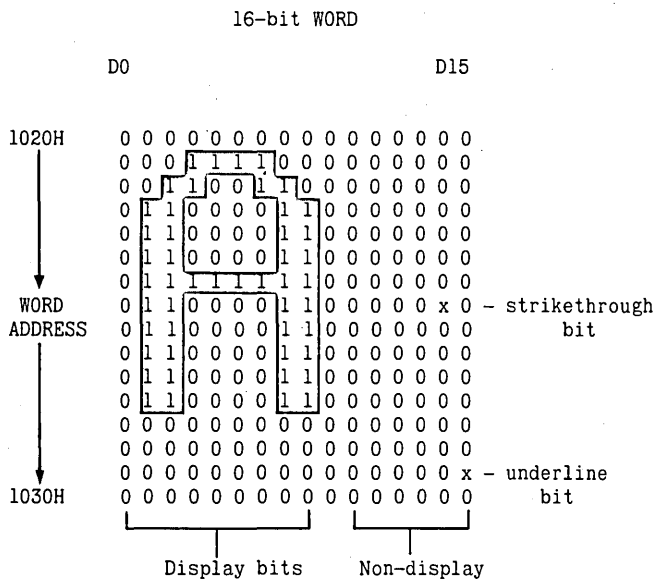
CRT CONTROL

detailed below. Since words are accessed from the RAM, the LSB is held permanently at the low state by a hardwired connection. The 4 MSB are also always low since the areas to be accessed are within the bottom 64K of the system memory map.



Text Mode

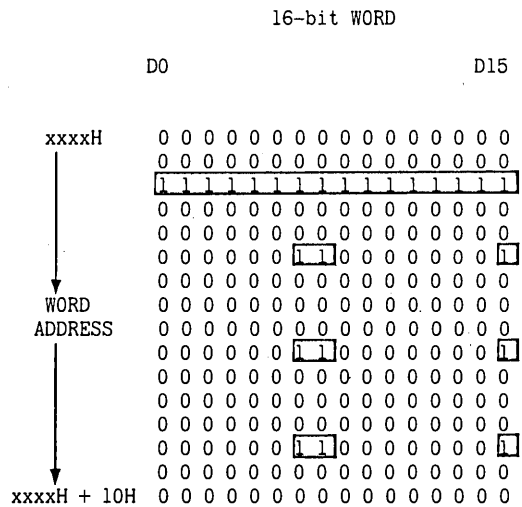
In the text mode, each character for display is stored in 32 contiguous bytes in memory, organised as 16 consecutive 16-bit words, defined as a font cell. An example of a font cell is detailed below. This represents the character **A** (of the default font) as illustrated on Figure 2.



The strikethrough bit defines the position of the horizontal line drawn through the character, if the strikethrough attribute is set for the character in the Screen RAM. Similarly, the underline bit defines the position of the line drawn under the character, if the underline attribute is set.

Graphics Mode

In the graphics mode, each character for display is stored in 32 contiguous bytes in memory, organised as 16 consecutive 16-bit words, defined as a graphics cell image. An example of a graphics cell image is detailed below. This represents the graphics cell N as illustrated on Figure 3.



CRT CONTROL

DISPLAY UNIT CONNECTOR DETAIL

The Display Unit connects to the System Board via the 9-pin male D-type connector located on the rear of the System Unit. The mating connector from the Display Unit is attached to a 6-wire cable assembly. The cable carries the video signal, the horizontal and vertical sync pulses, and also provides the +12V supply voltage for the Display Unit. The connections to the D-type connector are as detailed in the table below.

Pin	Description
1	+12V out
2	N.C.
3	0V
4	Horizontal Sync
5	Vertical Sync
6	Frame Ground
7	N.C.
8	Ground
9	Video signal

Both sync signals are at standard positive TTL levels. The frequency of the Horizontal Sync pulses to match the standard 9 inch Display Unit is 15.79 kHz; the frequency of the Vertical Sync pulses, 72 Hz. The amplitude of the video signal is fixed at 0.3V for normal intensity video, 0.4V for high intensity. The maximum current allowed to be drawn from the +12V supply is 1.0A.

FLOPPY DISK INTERFACE

List of Contents Page

List of Illustrations Figure

FLOPPY DISK INTERFACE

INTRODUCTION

The Floppy Disk Interface is located on the System Board and consists of the elements of circuitry as illustrated on the block diagram opposite. The interface provides all the control functions necessary for formatting and transferring data to/from the MicroFloppy Disks via the system disk drive(s).

The configuration of the interface incorporates all the necessary facilities to control either single or dual MicroFloppy Disk Drive systems, operating with either single or double-sided disks.

All disks, whether single or double-sided, are soft-sectored and encoded with the same disk format. This format is logically developed from the IBM system 34 format (a standard format for 8 inch disks), and is specifically designed to obtain the optimum number of data bytes on a 3.5 inch MicroFloppy disk. The format employs double density MFM coding with 512 bytes per sector and 9 sectors per track. The number of tracks per side of disk is solely a function of the version of disk drive, and may be either 70 or 80.

A brief description of the disk format is included at the end of this section (under the heading TRACK FORMAT). This should be read now, if the reader is unfamiliar with the method of recording data on disks.

Control connections between the interface and the drive(s) are all made internally within the System Unit. The Floppy Disk Drive Connector is a 26-way male IDC terminal mounted on the component side of the System Board. An associated ribbon cable assembly links the interface to the drive(s).

Regulated power supply voltages for the disk drives are supplied directly from the System Board. Separate power supply connections are provided for each disk drive, using Molex connectors (mounted on the component side of the System Board) and 4-wire cable assemblies.

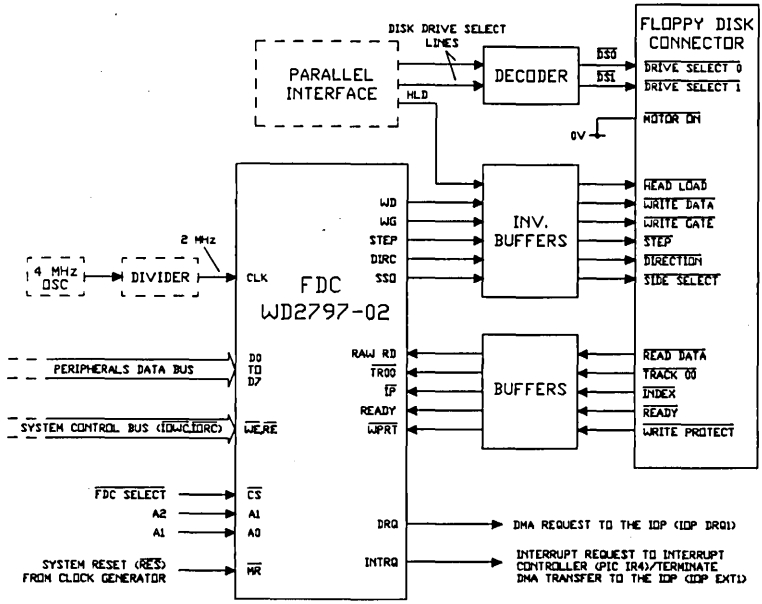


Figure 1. Floppy Disk Interface block diagram

FDC Pin Definition

WD	Write Data	DRQ	Data Request
WG	Write Gate	INTRQ	Interrupt Request
STEP	Step pulse output	CLK	Clock input
DIRC	Direction control	D0 to D7	Data bus
SSO	Side Select Output	$\overline{WE}$	Write Enable
RAW RD	Read Data	$\overline{RE}$	Read Enable
$\overline{TR00}$	Track 00	$\overline{CS}$	Chip Select
$\overline{IP}$	Index Pulse	A0,A1	Register selects
READY	Ready input	$\overline{MR}$	Master Reset
$\overline{WPRT}$	Write Protect		

FLOPPY DISK INTERFACE

DESCRIPTION

General

The Floppy Disk Interface consists of a Western Digital WD2797-02 Floppy Disk Controller (FDC), a series of buffers, a decoding circuit and the interface connector.

Three control lines for the disk drives are generated directly under software control, via the Parallel Interface on the System Board. Two of these control lines are routed via the decoding circuit to provide individual select lines ($\overline{DS0}$, $\overline{DS1}$), for the disk drives. The third control line is the head load control signal (HLD) for loading the read/write head of the selected disk drive.

The motor control line (Motor On) is hard-wired to 0V, thus providing a permanently active control signal to the disk drives. The configuration of the drives is set so that the motor operates, only when a disk is within the drive.

The remaining functions for, controlling the movement of the read/write head, transferring data to and from the disks, and monitoring status signals from the drives are implemented by the FDC. A detailed description of all these connections and all other connections to the interface is provided in tabular form at the end of this chapter.

All disk transfer operations performed by the FDC (formatting, reading disk data, writing data onto disks), are controlled by the system software, and involve both the 8086 CPU and the 8089 IOP. The IOP is used to perform DMA data transfers between system memory and the FDC on a byte-by-byte basis.

The actual sequence of events performed by the processors and the FDC during disk transfer operations is quite complex, involving an interchange of information between the two processors and the generation of various disk drive control signals. Due to this complexity, a simplified view of disk write, disk read and disk formatting operations is presented in the paragraphs below to illustrate some of the basic principles. A more detailed approach is taken in subsequent paragraphs.

Disk Write

Disk write operations are initiated by the CPU issuing a write command byte to the FDC over the peripherals data bus. The FDC then searches for the required destination on the disk for the data (correct side, track and sector, with the correct ID field CRC bytes). When the required location is found, the FDC signifies to the IOP that a data byte is required by issuing a DMA request, via the DRQ output. The disk write cycle then proceeds as follows:

1. The IOP writes the first data byte into a holding register (termed the Data Register) within the FDC.
2. The FDC resets the DRQ output and transfers the contents of the Data Register to an encoding circuit, and signifies to the IOP that another data byte is required.
3. The encoding circuit converts the byte into MFM encoded data and then writes the data to the disk.
4. The IOP responds by writing the next data byte into the Data Register.
5. The process is then repeated.

After the last data byte in the sector is written onto the disk, a two-byte CRC is computed internally within the FDC, which is then also written onto the disk.

On completing a write cycle (completed write operation to either a single sector or multiple sectors within a track), the FDC generates an interrupt request on the INTRQ output. The interrupt request performs a dual function. It informs the IOP to terminate the DMA transfer routine and also indicates to the CPU the end of the command cycle (via the Interrupt Controller). The CPU then can check a status register within the FDC to see whether any errors occurred during the write operation. Reading the Status Register or writing a new command to the FDC, automatically resets the INTRQ output.

Disk Read

Disk read operations are initiated by the CPU issuing a read command byte to the FDC over the peripherals data

FLOPPY DISK INTERFACE

bus. The FDC then searches the disk for the source of the data (correct side, track and sector, with correct ID CRC bytes). When the required location is found, the FDC begins the process of reading data from the disk.

The MFM encoded data is decoded, assembled into parallel data bytes, and transferred into the Data Register.

Every time a parallel data byte is transferred into the Data Register, the FDC signifies to the IOP that a data byte is available by issuing a DMA request, via the DRQ output.

Every time the IOP reads the assembled data byte, the FDC resets the DRQ output.

This process continues until the whole of the read operation is completed (reading the data from either a single sector or from multiple sectors within a track). The FDC then generates an interrupt request on the INTRQ output, which informs the IOP to terminate the DMA transfer routine and also indicates to the CPU, the end of the command cycle. Any errors produced during the transfer can then be checked by the CPU analysing the FDC status register.

Disk Formatting

Disk formatting is a similar process to disk write operations but involves transferring both data and gap information (unintelligent information used to separate areas of data on the disk, see Figure 2) onto the disk, and is executed on a track-by-track basis. The process is initiated by the CPU issuing a Write Track command to the FDC, via the peripherals data bus. The FDC responds by signifying to the IOP that a byte is required by issuing a DMA request, via the DRQ output.

The IOP then writes the first gap byte into the holding register (Data Register) within the FDC. The FDC resets the DRQ output and then waits for the Index Pulse (the marker for the start of a track) to be detected.

On detecting the Index Pulse, the FDC transfers the byte within the Data Register to an encoding circuit and issues another DMA request to the IOP. The IOP responds by writing the next byte to the Data Register and the process is repeated. The encoding circuit converts the byte into MFM data and writes the data onto the disk.

This process continues until the FDC detects the next Index Pulse, which causes the FDC to generate an interrupt request on the INTRQ output. The interrupt request again has the same functions as described before; i.e. informs the IOP to terminate the DMA transfer routine and also interrupts the CPU, via the Interrupt Controller. Resetting the INTRQ output is achieved by reading the Status Register or issuing a command to the FDC.

FLOPPY DISK INTERFACE

Read/Write Head Positioning

Prior to performing any of the disk transfer routines described above, the read/write head has to be positioned over the required track on the selected disk and the head load control signal issued to engage the read/write head. The head positioning operation is achieved under software control by issuing a command byte to the FDC.

The FDC responds to five different head positioning command bytes; each moves the head to a position specified by the command. The five commands are Restore, Seek, Step, Step-in and Step-out. The function of each command is detailed in the table below.

Head Positioning Commands

Command	Function
Restore	Positions the head over Track 0 on the Disk.
Seek	Positions the head to the track number specified in the Data Register.
Step	Moves the head to an adjacent track in the same direction as the previous head positioning command.
Step-in	Moves the head to the adjacent track in the direction away from Track 0.
Step-out	Moves the head to the adjacent track in the direction towards Track 0.

On completion of a head positioning command, the FDC generates an interrupt request on the INTRQ output to indicate to the CPU the end of the command cycle. Reading the Status Register or writing a new command to the FDC resets the INTRQ output.

An optional feature of each of the five commands is to automatically verify the track position of the head, by comparing the track number stored within an internal register (Track Register) with the track number contained in the ID fields on the disk. The verification operation also checks the ID field for errors utilising the ID field CRC bytes. Failure to detect the same track number or incorrect CRC detection, sets error status bits within the Status Register. Since this feature involves reading the disk ID field, the head has to be loaded prior to the command.

The signals supplied by the FDC to the disk drive, to position the read/write head are the STEP and DIRC (direction) outputs. A step pulse with a duration of 2 μs is issued to the drive every time the FDC wants to move the head by one track location. The FDC determines the direction of movement implied by the command and sets the state of the direction output accordingly. (Logic high to move the head away from Track 0, logic low to move the head towards Track 0).

FDC DETAIL

General

The FDC can be divided from a descriptive point of view into three areas of circuitry, a processor interface, a disk drive controller, and a series of registers.

The processor interface handles the transfer of data, commands and status, between the internal registers and the system processors, and also generates the interrupt and DMA request signals.

The disk drive controller responds to commands from the CPU, providing the neccessary circuitry, and input and output lines for:

(a) Controlling the positioning of the drive read/write head.
(b) Transferring data to and from the disks.
(c) Monitoring the disk drive status.

FLOPPY DISK INTERFACE

The internal registers provide the means of exchanging information (commands, status, positional data) between the disk drive controller and the processor interface.

Processor Interface

The connections to the processor interface are detailed in the table "System Connections" at the end of the chapter. The majority of the circuitry is a straightforward interface between the 8-bit bi-directional peripherals data bus and the series of addressable registers located within the FDC.

The system software views the registers as a series of peripheral ports located in the system input/output space. The port address locations as defined by the $\overline{\text{FDC select}}$ and the system address bus connections, are detailed below. The Data, Sector and Track Registers can be written to and read from. The Command Register can only be written to, and the Status Register can only be read from.

Register	Address
Command	40H
Status	40H
Track	42H
Sector	44H
Data	46H

The remaining circuitry of the processor interface consists of the control section, which produces the INTRQ and DRQ outputs.

The DRQ output is activated (set to logic high), during data transfer operations to and from the disk, and follows the state of an associated control bit within the Status Register. During disk read operations, the DRQ output is set active when the FDC has data available from the disk within the Data Register. The output is reset to the inactive state when the byte is read by the IOP.

During data write and formatting operations, the DRQ output is set active when the FDC Data Register is empty,

and the FDC requires another byte from the IOP. The output is reset when the Data Register is loaded with a new byte.

The INTRQ output is activated (set to logic high) on the successful completion of disk read, write, or formatting operations; and automatically reset following these operations, on reading the Status Register or issuing a new command to the Command Register.

The INTRQ output is also set to logic high, for a variety of other conditions (premature termination of a disk transfer command sequence due to an error condition, completion of a read/write head position command, etc.). In all cases, INTRQ can be reset by either reading the Status Register or issuing a command to the Command Register. These other conditions are detailed in the following paragraphs.

The first operation performed by the FDC on receiving a disk read or disk write command is to check whether the disk is ready for a transfer operation by analysing the READY input from the disk drive. If the input is set to logic high, the command sequence is initiated. If the input is set to logic low, the command sequence is immediately aborted, the Not Ready bit set within the Status Register, and the INTRQ output activated.

The FDC also analyses the Write Protect input ($\overline{\text{WPRT}}$) from the disk drive on receiving a disk write or formatting command. If the $\overline{\text{WPRT}}$ input is activated (logic low, indicating that the disk is write protected), the write operation is terminated, INTRQ is activated and the Write Protect bit set within the Status Register.

Prior to writing data to or reading data from the disk, the FDC locates the correct sector for the transfer operation, by analysing the ID fields on the disk. Failure to detect an ID field with the correct track number, correct side number, correct sector number and correct CRC within five revolutions of the Disk, sets the Record Not Found (RNF) bit in the Status Register, activates the INTRQ output and terminates the transfer operation.

FLOPPY DISK INTERFACE

At the start of disk write and formatting operations, the FDC signifies to the IOP that the first byte is required, via the DRQ output. If during formatting, the IOP fails to supply the first byte before the FDC detects the Index Pulse, the Lost Data (LD) bit is set within the Status Register, INTRQ is activated and the formatting operation is terminated. If during disk writes, the IOP fails to supply the first byte before the start of the data field, the same process occurs; the Lost Data bit is set, INTRQ is activated and the operation is terminated.

During disk read operations, the FDC checks the two-byte CRC code at the end of the sector data field to ensure the validity of the data. If a CRC error is present, the CRC error bit in the Status Register is set, INTRQ is activated and the read operation is terminated (even if the operation is a multiple sector read).

In addition to the transfer commands, the INTRQ output is also activated at the end of the command sequence for positioning the read/write head of the drives. The success of the operation is again signified by the status bits within the Status Register.

The first operation performed by the FDC on receiving the Restore command is a check of the Track 0 input ($\overline{TR00}$). If the $\overline{TR00}$ input is set low (indicating that the head is already positioned over the first track), and the verification option is not selected, the FDC clears the Track Register to zero and activates the INTRQ output.

If the $\overline{TR00}$ input is high, the FDC issues step pulses until the $\overline{TR00}$ input is set low, which then produces the same effect as described above, providing the verification option is not selected (i.e. INTRQ activated, Track Register cleared). If the $\overline{TR00}$ input is not set low within 255 step pulses, the operation is aborted, INTRQ is activated, and the Seek Error bit with the Status Register is set.

If the verification option is selected with any of the head positioning commands, the FDC moves the head to the specified position and then reads the first encountered ID field on the disk. The track number from the ID field is

compared with the track number stored in the Track Register. Providing the two numbers are identical and the ID field CRC bytes are correct, the track position is deemed true, and the INTRQ output is activated.

If the track numbers match, but the CRC is incorrect, the CRC error bit within the Status Register is set and the next encountered ID field is analysed. If the FDC fails to detect an ID field with a matching track number and correct CRC within 5 revolutions of the disk, the operation is aborted, INTRQ is activated, and the Seek Error bit within the Status Register is set.

The FDC can be also issued with a command (Force Interrupt), which sets the Status Register to monitor the state of the input status lines from the drive, and causes the INTRQ output to be activated for any of the conditions detailed below:

(a) Immediately the command is received.
(b) Every time an Index Pulse is detected.
(c) On detecting a transition on the Ready input.

Disk Drive Control

The disk drive controller circuitry acts as the interface between the disk drives and the other areas of circuitry within the FDC and consists of the disk data encoding and decoding sections, the head positioning control section and a status monitoring section.

Connections to and from the FDC disk controller circuitry are detailed in the table "Disk Drive Connections" at the end of the chapter.

At the start of a data transfer operation to the disk (write operation), the Write Gate output is activated and the IOP begins the process of transferring data bytes to the Data Register in parallel format, under DMA control. The FDC transfers each byte from the Data Register to the encoding circuit. The encoding circuit converts the bytes into MFM double density encoded data, which is then supplied to the disk via the Write Data output.

FLOPPY DISK INTERFACE

When writing to tracks with a track number greater than 43, the encoding circuit provides automatic write precompensation. The precompensation value is set by a potentiometer (WPW), located on the System Board.

The decoding section of the disk controller circuitry decodes the MFM encoded data from the Raw Read input, during transfer operations from the disk. The decoder is a phase-locked loop data separator circuit based around an internal voltage controlled oscillator (VCO) and phase detector. The centre frequency of the VCO is set by a variable capacitor (VC1), located on the System Board. A second variable control RPW sets the read window pulse width.

The head positioning control section responds to the head positioning commands supplied to the Command Register and controls the STEP and Direction (DIRC) outputs. The rate at which the 2 μs step pulses are issued to the drive, to move the head from track-to-track is specified by the command. Issuing a step pulse to the drive to move the head towards Track 0, automatically decrements the Track Register. Issuing a step pulse to the drive to move the head away from Track 0, automatically increments the Track Register.

The status monitoring section monitors the logic state on the four inputs from the disk drive, READY, $\overline{\text{TR00}}$, Index Pulse ($\overline{\text{IP}}$)and Write Protect ($\overline{\text{WPRT}}$). All the four inputs can be monitored by issuing any of the head positioning commands or the Force Interrupt command, and then examining the contents of the Status Register. The effect of the status inputs on the operation of the FDC is dependent on the command issued to the Command Register and the logic state of the input line.

Command Register

The 8-bit Command Register holds the command supplied from the CPU which determines the type of operation carried out by the FDC, and is located at address location 40H in the I/O space. The register can only be written to with one of eleven predefined command words. A

command currently in progress is indicated by an associated control bit within the Status Register.

The eleven commands can be divided into four different categories as detailed below. A detailed description of each command is provided in the PROGRAMMING CONSIDERATIONS section.

Type 1. Read/write head positioning commands: Restore, Seek, Step, Step-in, Step-out.

Type 2. Data transfer commands: Read Sector, Write Sector.

Type 3. Format code transfer commands: Read Address, Read Track, Write Track.

Type 4. Interrupt command: Force Interrupt.

Status Register

The 8-bit Status Register holds status information, which is dependent on the command operation performed by the FDC. The register is located at address 40H in the system I/O space, and can only be read from. Some of the bits within the register signify the state of the control inputs from the disk drive, whilst others indicate the status of the command operation.

Track Register

The 8-bit Track Register indicates the track number of the position of the drive read/write head and is located at address location 42H in the system I/O space. The register can be written to and read from. The FDC updates the track register during head positioning command operations, every time the drive head is moved to an adjacent track.

Sector Register

The 8-bit Sector Register is used to store a sector number, which indicates to the FDC the desired location on the disk for a transfer operation. The register is located at address

location 44H in the system input/output space and can be written to and read from. During multiple sector transfer operations, the sector register is updated by the FDC.

Data Register

The 8-bit Data Register is the holding register during transfer operations to and from the disk, and is located at address location 46H in the system input/output space. The register performs a different function during the Seek Command, when the register is programmed with the desired track location.

PROGRAMMING CONSIDERATIONS

Disk Drive Selection

Selecting a disk drive for operation is achieved by writing data to the Parallel Interface. The address location and data required to select a specific drive is detailed in the section on the Parallel Interface.

Disk drives are specified as Drive 0 or Drive 1 according to the position of a switch located at the rear of each drive. A logic low on the Drive Select 0 control line selects the disk drive configured as Drive 0, and a logic low on the Drive Select 1 control line selects the disk drive configured as Drive 1. The decoding circuitry of the interface is employed to ensure that both drives are not selected simultaneously under software control, by only allowing one of the drive select lines to be set to the active logic low state at any one time.

Since the Track Register contains a record of the position of the head of the drive selected, changing operation from one drive to the second drive in dual drive systems, requires the Track Register to be updated accordingly.

Head Loading

Loading the read/write head of the selected disk drive is achieved by writing data to the Parallel Interface. The address location and data required to load the head are detailed in the section on the Parallel Interface.

The time taken for the standard disk drive (the version using 70 track single-sided disks), to engage the read/write head after the head load signal is set active, is of the order of 60 ms. A 60ms delay should therefore be implemented after the head load signal is set active, to ensure the head is engaged before performing disk data transfer operations. All command operations can be performed with the head loaded without any detrimental effect to the disk.

Head Positioning

Positioning the read/write head of the selected disk drive is achieved by issuing one of the five Type 1 commands to the Command Register of the FDC. Termination of a command (successful or otherwise) is signified by an interrupt request to the Interrupt Controller (PIC). The format of each of the head positioning commands is detailed below.

Type 1 Commands

D7							D0	Command
0	0	0	0	1	V	r1	r0	Restore
0	0	0	1	1	V	r1	r0	Seek
0	0	1	T	1	V	r1	r0	Step
0	1	0	T	1	V	r1	r0	Step-in
0	1	1	T	1	V	r1	r0	Step-out

T = Track Update Flag
V = Verify Flag
r1,r0 = Stepping motor rate

Each of the commands contain a Verify Flag bit and stepping motor rate bits. The stepping rate bits have to be set according to the track-to-track access time of the disk drive. The combination of bits allow four different stepping motor rates to be selected as detailed below. The track-to-track access time of the standard disk drives is 15 ms.

r1	r0	Rate
0	0	3 ms
0	1	6 ms
1	0	10 ms
1	1	15 ms

FLOPPY DISK INTERFACE

The Verify Flag bit determines whether the FDC verifies the track position after positioning the head, by reading the sector ID fields on the disk. Verification is performed, when the Verify Flag bit is set to logic high, and requires the head to be loaded prior to the command.

The verification operation checks the track number in the sector ID field with the number contained in the Track Register, and also checks the ID field CRC character.

Failure to match the track number or failure to find a matching track with a valid CRC, within five revolutions of the disk, causes the Seek Error bit to be set in the Status Register. If the verification operation detects a CRC error within any of the ID fields checked, the CRC error bit is set within the Status Register.

The Step commands contain a Track Update Flag, which determines whether the Track Register is updated every time the head moves to an adjacent track. The Track Register is updated if the Track Update Flag is set to logic high.

Restore Command. The Restore command is used to position the head over Track 0 of the disk. The FDC will issue up to 255 step pulses at the rate specified by the stepping rate bits, in an attempt to locate the first track on the disk. Failure to locate Track 0 (by monitoring the state of the $\overline{\text{TR00}}$ input) within 255 step pulses, causes the command to terminate, and the Seek Error bit to be set within the Status Register. If the Verify Flag bit is set, verification of the track position is carried out as detailed above.

Seek Command. Prior to issuing the Seek command, the Data Register has to be loaded with the desired track number and the Track Register is presumed to contain the current position of the head. On receiving the command, the FDC sets the Dirc output to move the head in the direction of the desired track. Step pulses are then issued at the rate specified by the stepping rate bits, and the Track Register updated on each pulse, until the number in the Track Register coincides with the number in the Data

Register. If the Verify Flag is set, verification of the track position is carried out.

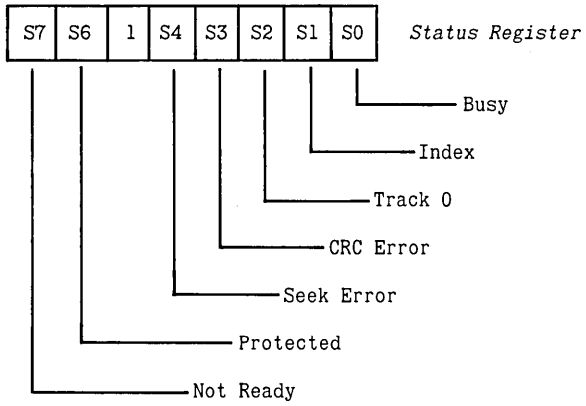
Step Command. Issuing a Step command moves the head one track location in the direction specified by the previous command. If the Track Update Flag is set, the Track Register is updated following the issue of the Step pulse. If the Verify Flag is set, verification of the track position is carried out, after a delay determined by the stepping rate bits (i.e. A delay of 15 ms. for the standard disk drives).

Step-in Command. Issuing the Step-in command moves the head one track location away from Track 0. If the Track Update Flag is set, the Track Register is incremented following the issue of the Step pulse. If the Verify Flag is set, verification of the track position is carried out, after a delay determined by the stepping rate bits.

Step-out. Issuing the Step-out command moves the head one track location towards Track 0. If the Track Update Flag is set, the Track Register is decremented following the issue of the Step pulse. The Verify Flag functions in the same manner as described above for the Step and Step-in commands.

FLOPPY DISK INTERFACE

Status Register. The format of the Status Register following a head positioning command is as detailed below. On issuing a command to the Command Register, the FDC resets the Status Register to monitor certain status conditions implied by the new command. Due to internal timing delays, the Status Register does not contain valid status information, until 14 μs after the new command is issued.



Busy	A logic high on Busy indicates that the command is in progress; logic low indicates that the command sequence is complete.
Index	Inverted copy of the Index Pulse input from the disk drive. Logic low when the pulse occurs.
Track 0	Inverted copy of the $\overline{\text{TR00}}$ input from the drive. Logic high indicates that the head is positioned over Track 0.
CRC Error	Logic high indicates a CRC error was detected within a sector ID field during verification of the track.

continued ....

Seek Error	Logic high indicates a matching track number was not located within a sector ID field during the verification operation. The bit is also set high if the FDC fails to detect a logic low on the $\overline{TR00}$ input following a Restore command.
Protected	Inverted copy of the $\overline{WPRT}$ input from the drive. Logic high indicates that the selected disk drive contains a write protected disk.
Not Ready	Inverted copy of the READY input from the drive. Logic high indicates that the drive is not ready for a data transfer operation.

Data Transfers

Transfer of data to and from the disk is controlled by the two Type 2 commands, Write Sector and Read Sector. Prior to issuing the command to the Command Register, the Sector Register must be loaded with the desired sector number, to determine the source/destination of the data. Termination of the command (successful or otherwise) is signified by an interrupt request to the PIC. Both command operations are prematurely terminated, if the disk drive is not ready for the transfer operation, as indicated by the READY input.

The format of the data transfer commands is detailed below.

Type 2 Commands

D7							D0	Command
1	0	0	m	1	1	U	0	Read Sector
1	0	1	m	1	1	U	0	Write Sector

m = Multiple Record Flag
U = Update SSO

Both commands contain a Multiple Record Flag bit and an Update SSO bit. The Update SSO bit is used to select the disk side for the data transfer operations and affects the

logic state of the Side Select Output (SSO), supplied to the disk drive. When U is set to logic low, the SSO is updated to logic low (side 0). When U is set to logic high, SSO is updated to logic high (side 1).

The Multiple Record Flag bit selects whether data transfers are from/to a single sector or multiple sectors within a track. Single sector transfers are initiated when the m Flag is set to logic low, multiple sector transfers when the m Flag is set to logic high.

Write Sector. On receipt of the Write Sector command, the FDC begins the process of searching the sector ID fields of the track for the desired destination for the data. When an ID field is found with the correct track number (as specified by the Track Register), the correct side number (as specified by the U bit in the command), the correct sector number (as specified by the Sector Register), and correct CRC character; the FDC generates a DMA request via the DRQ output, to inform the IOP to write the first data byte into the Data Register.

If an ID field is not found containing the correct information within five revolutions of the disk, the command is aborted and the Record Not Found bit in the Status Register set.

If any of the ID fields encountered, contain an incorrect CRC character, this is also recorded in the Status Register.

On receipt of the first data byte, the FDC activates the Write Gate output and on detecting the start of the data field, writes the data byte to the disk. The process then continues with the FDC generating DMA requests every time a new byte of data is required. After the 512th data byte is written to the disk, a two-byte CRC character is automatically generated and written onto the disk. If the data written to the sector only fills a part of the data field, the remaining area should be programmed with a series of zeroes, to complete the whole data field.

If a single sector write operation was specified by the Write Sector command, the Write Gate output is then deactivated and the command operation terminated.

If the Write Sector command specified a multiple sector transfer, the Sector Register is incremented and the process repeated, starting from the verification operation on the next ID field. The multiple sector transfer operation continues until terminated either by issuing a Force Interrupt command, or after the sector number is incremented to a value exceeding the number of sectors on the track. In the latter case, the FDC automatically terminates the command after five revolutions of the disk, since the verification of the ID field will not be able to locate a matching sector number.

Failure of the IOP to write the first data byte to the Data Register before the arrival of the sector data field causes the FDC to, set the Lost Data bit in the Status Register, activate the INTRQ output, and abort the command. Failure of the IOP to supply a data byte to the Data Register on receiving a DMA request after the first byte (i.e. within 16 μs), causes the FDC to write a byte of zeroes onto the disk and also set the Lost Data bit, but does not terminate the command sequence.

Read Sector. On receipt of the Read Sector command, the FDC begins the process of searching the sector ID fields of the track for the desired source of data. When an ID field is found with the correct track number (as specified by the Track Register), the correct side number (as specified by the U bit in the command), the correct sector number (as specified by the Sector Register), and the correct CRC character; the FDC reads the data bytes from the following data field, and informs the IOP, via the DRQ output, every time a data byte is stored in the Data Register.

If an ID field is not found containing the correct information within five revolutions of the disk, the command is aborted and the Record Not Found bit in the Status Register set.

If any of the ID fields encountered, contain an incorrect CRC character, this is also recorded in the Status Register.

Failure of the IOP to read the contents of the Data Register before it is overwritten with the next byte of data

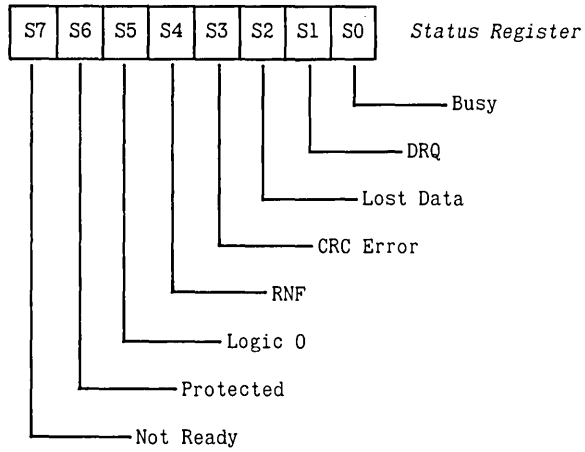
from the disk, causes the FDC to set the Lost Data bit in the Status Register.

If a single sector read operation was specified by the Read Sector command, the sequence terminates with the FDC testing the CRC character at the end of the data field. If the CRC character is incorrect, the CRC Error bit is set within the Status Register.

If the Read sector command specified a multiple sector transfer, the Sector Register is incremented and the process repeated for the next sector, providing the CRC character tested at the end of the previous data field was true. If the CRC character was incorrect, the multiple sector routine is prematurely aborted.

If no CRC Errors are detected, the multiple sector operation continues until terminated either by issuing a Force Interrupt command, or after the sector number is incremented to a value exceeding the number of sectors on the track. In the latter case, the FDC automatically terminates the command after five revolutions of the disk, since the verification of the ID field will not be able to locate a matching sector number.

Status Register. The format of the Status Register following a data transfer command is as detailed below. On issuing a command to the Command Register, the FDC resets the Status Register to monitor certain status conditions implied by the new command. Due to internal timing delays, the Status Register does not contain valid status information, until 14 μs after the new command is issued.



Busy	A logic high on Busy indicates that the command is in progress; logic low indicates that the command sequence is complete.
DRQ	A copy of the DRQ output to the IOP. A logic high indicates that the FDC has data available during a read operation, and requires data during a write operation.
Lost Data	Logic high indicates that the IOP did not respond to the DRQ output in time, and as a result; valid data was not written onto the disk during a write operation, valid data was not read from the disk during a read operation.
CRC Error	During the write operation, a logic high indicates a CRC error was detected within one or more sector ID fields. During the read operation, a logic high indicates a CRC error was detected either within one or more ID fields, or a CRC error detected at the end of a data field.

continued ....

FLOPPY DISK INTERFACE

RNF	Record Not Found. A logic high indicates that the transfer operation was unable to locate an ID field containing the correct side, track and sector number, with a valid CRC character.
Protected	Inverted copy of the WPRT input from the drive, during write operations. Logic high indicates that the selected disk drive contains a write protected disk. During read operations set to logic low.
Not Ready	Inverted copy of the READY input from the drive. Logic high indicates that the drive is not ready for a data transfer operation.

Formatting Commands

The formatting commands are the three Type 3 commands, Write Track, Read Track and Read Address. Write Track is the command for formatting disks. The two other commands enable the programmer to check the disk format. Termination of the command is signified by an interrupt request to the Interrupt Controller. The command operations are terminated prematurely, if the disk drive is not ready for the transfer operation as indicated by the READY input.

The format of these commands is detailed below.

Type 3 Commands

D7							D0	Command
1	1	0	0	0	1	U	0	Read Address
1	1	1	0	0	1	U	0	Read Track
1	1	1	1	0	1	U	0	Write Track

U = Update SSO

Each of the three commands contain an Update SSO bit. This bit is used to select the disk side for the formatting operations and affects the logic state of the Side Select Output (SSO), supplied to the disk drive. When U is set to logic low, SSO is updated to logic low (side 0). When U is set to logic high, SSO is updated to logic high (side 1).

Track Format. The track format used on the MicroFloppy disks is illustrated at the end of the chapter.

Read Address. On receipt of the Read Address command, the FDC searches the disk for an ID field. The first ID field encountered is then read from the disk. The FDC transfers the six ID field bytes detailed below, from the disk to the Data Register one byte a time. Every time a data byte is stored in the Data Register, the FDC generates a DMA request to the IOP. The ID field track number is also transferred to the Sector Register.

Byte	Description
1	Track Number
2	Side Number
3	Sector Number
4	Sector Length (02H)
5	CRC 1
6	CRC 2

If the FDC fails to locate an ID field within five revolutions of the disk, the command is aborted and the Record Not Found bit in the Status Register set. If the ID field contains an incorrect CRC character, this is also recorded in the Status Register.

Failure of the IOP to read the contents of the Data Register, before the register is overwritten with the next byte of data from the disk, causes the FDC to set the Lost Data bit in the Status Register.

Read Track. The Read Track command allows a complete track of information (Gaps, Headers and Data bytes) to be

FLOPPY DISK INTERFACE

read from the disk. On receipt of the Read Track command, the FDC monitors the logic state on the Index Pulse input. On detecting the leading edge of the pulse, the FDC transfers all the Gap, Header and Data bytes from the track into the Data Register on a byte-by-byte basis.

Every time a byte is transferred into the Data Register, the FDC informs the IOP that a byte is available, via the DMA request line. The command terminates after one full revolution of the disk, on detecting the Index Pulse again. No CRC error checking is carried out.

Every time the FDC detects the ID field and Data field address marks, it synchronises the internal decoding circuitry, to ensure that all information following each sector ID address mark is valid. Since the FDC might not be synchronised to the gap information previous to the ID address mark, this information is not guaranteed to be valid.

Failure of the IOP to read the contents of the Data Register, before the register is overwritten with the next byte of data from the disk, causes the FDC to set the Lost Data bit in the Status Register.

Write Track. Write Track is the command used to format the tracks on the disk. All data and gap information is supplied to the disk by building a image of the track in memory as detailed on the next page, and transferring the format data to the Data Register under DMA control, using the IOP.

On receipt of the Write Track command, the FDC generates a DMA request via the DRQ output, to inform the IOP to write the first byte from the track image into the Data Register. On detecting the leading edge of the Index Pulse, the FDC transfers the first byte to the disk and requests another byte from the IOP. The process then continues with the IOP supplying the format bytes with the FDC generating DMA requests every time a new byte is required. The command sequence terminates after one full revolution of the disk, on detecting the leading edge of the next Index Pulse.

Track Image

Number of bytes required	Byte Value (Hex)	Description	
80	4E	Gap	
12	00	Sync	
3	FE	Control Bytes	*PREAMBLE*
1	FC	Index Mark	
50	4E	Gap	
12	00	Sync	
3	F5	Control Bytes	
1	FE	ID Address Mark	
1	00 to 45*	Track Number	
1	00 or 01	Side Number	
1	01 to 09	Sector Number	*SECTOR*
1	02	Sector Length	(Repeated 9
1	F7	CRC character command	times
22	4E	Gap	incrementing
12	00	Sync	the Sector
3	F5	Control Bytes	Number each
1	FB	Data Address Mark	time)
512	00	Data Bytes	
1	F7	CRC character command	
84	4E	Gap	
598**	4E	Gap	*POSTAMBLE*

* For 70 track disks.

** Nominal figure, write operation continues until terminated by the INTRQ output, on detecting the Index Pulse.

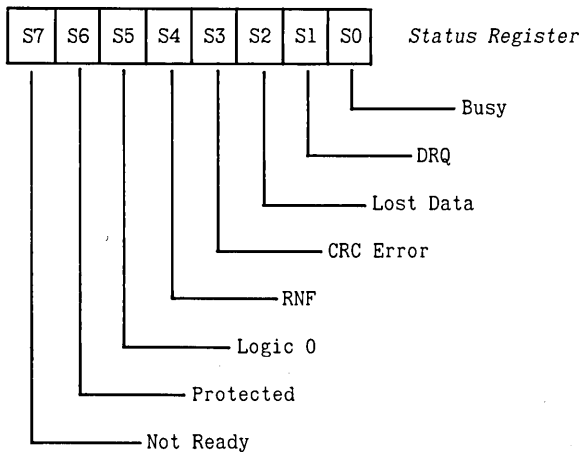
Some of the bytes supplied to the Data Register are not written onto the disk, but act as control bytes to the FDC. These are the three bytes F5H, F6H and F7H. F5H commands the FDC to perform two functions; write a unique byte pattern corresponding to A1 with a missing clock transition, onto the disk; initialize the CRC generator circuitry. F6H commands the FDC to write a unique pattern corresponding to C2H with a missing clock transition onto the disk. These patterns enable the FDC to interpret the byte following the three byte pattern as an address mark, during decoding operations. The byte F7H commands the

FLOPPY DISK INTERFACE

FDC to write the computed two-byte CRC character onto the disk.

Failure of the IOP to write the first byte to the Data Register before the arrival of the leading edge of the Index Pulse causes the FDC to, set the Lost Data bit in the Status Register, activate the INTRQ output, and abort the command. Failure of the IOP to supply a data byte to the Data Register on receiving a DMA request after the first byte, causes the FDC to write a byte of zeroes onto the disk and also set the Lost Data bit, but does not terminate the command sequence.

Status Register. The format of the Status Register following a formatting command is as detailed below. On issuing a command to the Command Register, the FDC resets the Status Register to monitor certain status conditions implied by the new command. Due to internal timing delays, the Status Register does not contain valid status information, until 14 μs after the new command is issued.



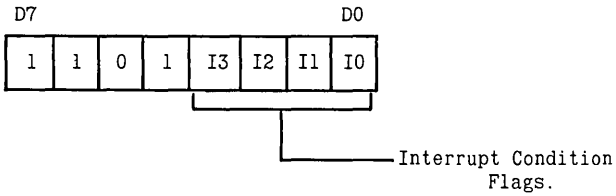
Busy	A logic high on Busy indicates that the command is in progress; logic low indicates that the command sequence is complete.
DRQ	A copy of the DRQ output to the IOP. A logic high indicates that the FDC has data available during the Read Track/Read Address operations, and requires data during a Write Track operation.
Lost Data	Logic high indicates that the IOP did not respond to the DRQ output in time, and as a result; valid data was not written onto the disk during a Write Track operation/valid data was not read from the disk during a read operation.
CRC Error	During the Read Address operation, a logic high indicates a CRC error was detected within the sector ID field. During the Read Track and Write Track operations set to logic low.
RNF	Record Not Found. A logic high indicates that the Read Address operation was unable to locate an ID field. During Read Track and Write Track set to logic low.
Protected	Inverted copy of the $\overline{\text{WPRT}}$ input from the drive, during the Write Track operation. Logic high indicates that the selected disk drive contains a write protected disk. During read operations set to logic low.
Not Ready	Inverted copy of the READY input from the drive. Logic high indicates that the drive is not ready for a data transfer operation.

FLOPPY DISK INTERFACE

Force Interrupt Command

The Force Interrupt command is a Type 4 command and is used to force the FDC to generate an interrupt via the INTRQ output on detecting a certain condition. The conditions are defined by setting certain bits within the command byte as detailed below.

Type 4 Command



Interrupt Condition Flags

I3	I2	I1	I0	Interrupt Condition
x	x	x	1	Not Ready to Ready Transition
x	x	1	x	Ready to Not Ready Transition
x	1	x	x	Every Index Pulse
1	x	x	x	Immediate Interrupt
0	0	0	0	No Interrupt

More than one condition can be set at any time by the appropriate combination of Interrupt Condition Flags.

If any of the Force Interrupt commands are issued to the Command Register when a command is already in operation, the current command is terminated and the Busy bit in the Status Register reset. All the other bits in the Status Register are left unchanged. If the command is issued when there is no other command currently in operation, the Status Register is set to monitor the same conditions as a Type 1 command, as previously described.

The No Interrupt command functions in an identical manner to the other Force Interrupt commands, as described in the paragraph above, but does not produce an interrupt request on INTRQ.

The INTRQ output is reset following any of the Force Interrupt commands, by issuing any new command to the Command Register or by reading the Status Register. The only method available to reset the INTRQ output after issuing the Immediate Interrupt command, is to issue the No Interrupt command.

After issuing a Force Interrupt command to the Command Register:

(a) Another command should not be issued for at least 8 μs, to allow the FDC to process the interrupt command.
(b) The Status Register should not be read for 14 μs to ensure the register contains valid status information.

INTERFACE CONNECTION DETAIL

System Connections

D0 to D7	Data bus. Used to transfer data, commands and status information between the processing elements and the FDC.
$\overline{WE}$	Write Enable. Write control input connected to the $\overline{IOWC}$ control line of the system control bus. Active state, logic low. Used in conjunction with $\overline{CS}$ and the register select inputs (A0, A1) to transfer data and commands from the data bus to the FDC.
$\overline{RE}$	Read Enable. Read control input connected to the $\overline{IORC}$ control line of the system control bus. Active state, logic low. Used in conjunction with $\overline{CS}$ and the register select inputs (A0, A1) to transfer data and status information onto the data bus from the FDC.
$\overline{CS}$	Chip Select. Address input. Active state, logic low. When active, indicates that the FDC is selected for a data/command transfer operation.

continued ....

FLOPPY DISK INTERFACE

<table>
<tr><td>A0, A1</td><td>Register select lines. Inputs connected to A1 and A2 of the system address bus. Used to select internal registers within the FDC for data/command/status transfers, via the data bus as detailed below.

A1 A0
<table><tr><td>0</td><td>0</td><td>Status/Command registers</td></tr><tr><td>0</td><td>1</td><td>Track register</td></tr><tr><td>1</td><td>0</td><td>Sector register</td></tr><tr><td>1</td><td>1</td><td>Data register</td></tr></table></td></tr>
<tr><td>INTRQ</td><td>Interrupt Request. Output connected to an interrupt request line of the Interrupt Controller (PIC) and to the external terminate line of channel 1 of the Input Output Processor (IOP). Active state, logic high. When active, signifies that the FDC requires attention from the processing system and that any current DMA transfer operation should be terminated.</td></tr>
<tr><td>DRQ</td><td>Data Request. Output connected to the DMA request line of channel 1 of the IOP. Active state, logic high. When active, signifies to the IOP that the FDC is ready for a DMA transfer operation.</td></tr>
<tr><td>$\overline{MR}$</td><td>Master Reset. Input connected to the System Reset control line. Active state, logic low. When active, causes the FDC to reset the internal registers.</td></tr>
<tr><td>CK</td><td>Clock input. 2 MHz clock with a 50% duty cycle for internal timing within the FDC.</td></tr>
</table>

Disk Drive Connections

$\overline{DS0}$	Drive Select 0. Control signal supplied from the Parallel Interface via the decoder. Active state, logic low. When active, selects the disk drive configured as drive 0.
$\overline{DS1}$	Drive Select 1. Control signal supplied from the Parallel Interface via the decoder. Active state, logic low. When active, selects the disk drive configured as drive 1.
HLD	Head Load. Control signal supplied from the Parallel Interface. Active state, logic high. When active, engages the read/write head of the selected disk drive against the disk.
SSO	Side Select Output. Control signal from the FDC used to select the side 0 or side 1 of double-sided disks. Follows the state of an associated control bit within an internal register. A logic high on SSO selects side 0 and a logic low, side 1.
STEP	Step Pulse. Pulsed output generated by the FDC for positioning the disk drive read/write head. Each positive going pulse moves the head to an adjacent track location in the direction determined by the DIRC output.
DIRC	Direction Control. Control signal generated by the FDC to determine the direction of movement of the disk drive read/write head. When DIRC is set to logic high, each step pulse causes the head to step in one track (away from track 0). When DIRC is set to logic low, each step pulse causes the head to step out one track (towards track 0).

continued ....

FLOPPY DISK INTERFACE

WG	Write Gate. Control signal generated by the FDC. Active state, logic high. When active, enables current to flow into the disk drive read/write head.
WD	Write Data. Output for writing MFM encoded data to the disk drive.
RAW RD	Raw Read. Input to the FDC for MFM encoded data from the disk drive.
READY	Input to the FDC from the disk drive. When set to logic high, indicates that the selected disk is ready for data transfer operations. When set to logic low, indicates that the selected disk is not available. In this condition attempted data transfer operations between the FDC and the disk drive are inhibited, and cause an active interrupt request to be generated on INTRQ. READY also sets an associated control bit within an internal register according to the logic state on the control line.
$\overline{\text{TR00}}$	Track 00. Control input from the disk drive. When $\overline{\text{TR00}}$ is set to logic low, indicates to the FDC that the read/write head of the selected drive is positioned over track 0 of the disk.
IP	Index Pulse. Control input from the disk drive. A negative going pulse generated every revolution of the disk, which informs the FDC of the start of the first sector of each track.
$\overline{\text{WPRT}}$	Write Protect. Control input from the disk drive. When $\overline{\text{WPRT}}$ is set to logic low, any attempted write operation to the selected drive is inhibited. $\overline{\text{WPRT}}$ also sets an associated control bit within an internal register according to the logic state in the control line.

TRACK FORMAT

Data is recorded on a disk in concentric circles, known as tracks. When a disk is inserted into the disk drive, the Auto Shutter is opened to allow the read/write head of the disk drive access to the disk surface. When the head is loaded, the head makes physical contact with the radial slot of magnetic material exposed, when the shutter is open. Information on a track is read and written serially as the disk rotates within its protective plastic shell.

Each track of the disk is divided into nine sectors by software formatting (soft sectoring). The sectors are recorded onto each track of the disk by issuing a format command to the FDC and then writing all the bytes onto the disk as illustrated in Figure 2. The start of each track is marked by a single index pulse, which is generated by the disk drive every revolution of the disk.

Each sector has an identification (ID) field and a data field, separated by gaps of unintelligent information. The gaps are required to allow the FDC to process information from the disk during disk reads, and also to take into account variations in drives, such as motor speed variations. The ID field defines the data field that follows, by specifying its track number, side number, sector number and length of the data field in bytes.

Both the ID and data fields begin with an address mark and end with a two-byte cyclic redundancy check (CRC) character for detecting errors in the previous field. Different address marks are used to distinguish between the two fields.

All the serial data on the track (ID fields, data fields and gap information) are recorded on the disk by Modified Frequency Modulation (MFM). In MFM encoding, both serial data bits and clock pulses are interleaved into the data stream.

In order to distinguish the address marks on the disk from data bytes which may be identical, the address marks are recorded with missing clock pulses in predefined locations.

FLOPPY DISK INTERFACE

The address marks are the only bytes on the disk with missing clock bits, and are produced by the FDC during formatting. These are then used by the FDC for recognition and synchronisation during disk accesses.

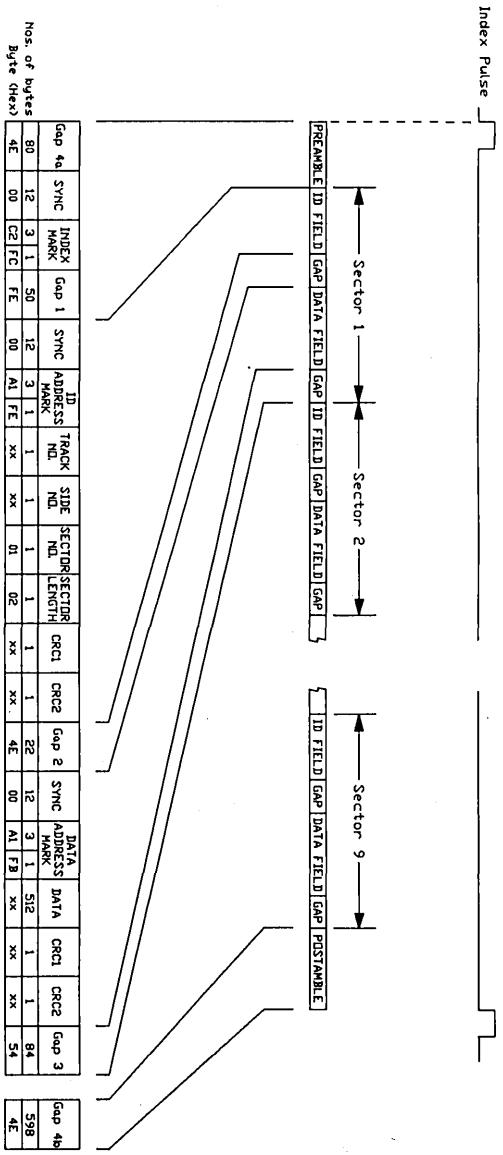


Figure 2. Track Format

SERIAL INTERFACE

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The Serial Interface is located on the System Board and consists of the elements of circuitry as illustrated on the block diagram Figure 1. The interface provides two separate serial channels; one channel for communication between the Keyboard and the System Unit, the second channel for communication between the Apricot and external equipment via an RS232C link.

SERIAL INTERFACE

The serial channel between the Keyboard and the System Unit supports full duplex asynchronous communications at a fixed baud rate, and also carries the hardware system reset control signal which is multiplexed onto the Data in line from the Keyboard.

The RS232C channel can be programmed to operate in either asynchronous or synchronous modes, with transmit and receive baud rates determined either via the Programmable Interval Timer (TMR), or via the external data communications equipment.

The RS232C interface is able to support:

(a) Asynchronous communications with 5, 6, 7 or 8 bits per character and various choices of stop bits and parity sense.
(b) Bit oriented synchronous communications, such as SDLC and HDLC.
(c) Byte oriented synchronous communications, such as Monosync and Bisync.

A further function of the Serial Interface is to provide the Parallel Interface on the System Board with a means of generating interrupt vectors, which act as pointers to associated interrupt service routines. This is achieved by utilising spare facilities in the keyboard serial channel.

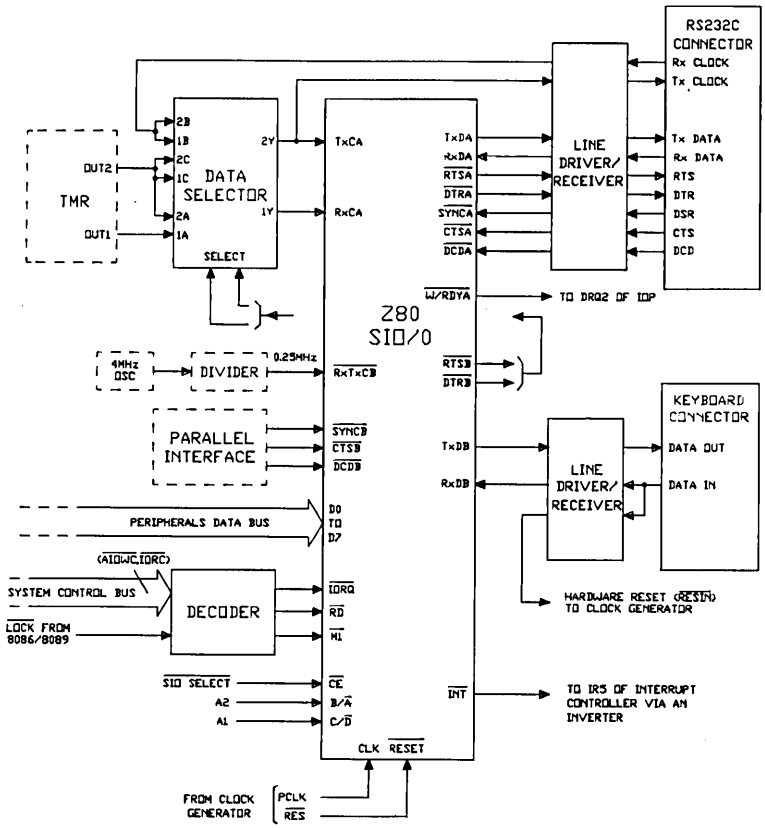


Figure 1. Serial Interface block diagram

DESCRIPTION

General

The Serial Interface consists of; a serial input/output controller, a data selector, a number of line drivers/receivers and two connectors (a 25 pin D-type female connector for the RS232C channel and a 9 pin D-type female connector for the keyboard link).

The major circuit element of the Serial Interface is the serial input/output controller which is a Zilog Z80 SIO/0. The SIO is a programmable device which incorporates all

SERIAL INTERFACE

the circuitry for interfacing between the system processors and the two serial communication channels. A description of each connection to the SIO is provided at the end of the chapter.

The data selector allows the programmer to select the clock source which determines the transmit and receive baud rates for the RS232C channel.

Due to the relative complexity of the SIO compared with other programmable devices on the System Board, a brief overview of the device is presented in the following paragraphs. Subsequent paragraphs describe the internal structure and the utilisation of the device within the system in more detail.

SIO Overview

The SIO is a peripheral device, which performs the function of an intelligent two-channel parallel to serial/ serial to parallel converter. The two channels (designated Channel A and Channel B) are totally independent of each other and can be programmed to operate in either asynchronous or synchronous serial data communications modes.

Once programmed with the required serial data communications mode, the processors are able to view the SIO as two simple parallel input/output ports for the majority of data transfer operations; one bi-directional port for each channel. The SIO provides all the necessary facilities for organising data transfers over the two serial channels. These include:

(a) Conversion of the parallel data into the correct serial data format for transmission over the selected serial channel. In asynchronous modes, this involves the automatic insertion of start, stop and parity bits, and assembling the parallel data into the correct number of serial bits per character.

(b) Conversion of the input serial data from the serial channels into parallel form, and checking the incoming data for errors. In asynchronous modes, this involves stripping the start, stop and parity bits from the serial data and checking for parity and framing errors.

When operating in synchronous modes, similar automatic facilities are available within the SIO, dependent on the selected mode.

In Bisync and Monosync, pre-programmed sync characters and CRC characters are automatically inserted during transmissions. The SIO strips sync characters from received data, and also analyses the received data for errors, utilising the received CRC characters.

In SDLC and HDLC, the SIO performs the following functions:

(a) Abort sequence generation and detection.
(b) Zero insertion and deletion.
(c) Flag insertion between messages.
(d) Address field recognition.
(e) I-field residue handling.
(f) CRC generation and detection.

The baud rates for the two serial channels are determined by clock sources external to the SIO. In asynchronous modes, the clocks can be divided internally within the SIO under program control, prior to being used to set the baud rates. This facility is not available in synchronous modes.

Each channel also provides inputs and outputs which are specifically tailored to function as modem control lines for co-ordinating serial data transfer operations. The outputs are under direct control by the programmer. The inputs can be monitored under software control by polling, or be programmed to generate an interrupt to the CPU. These control lines can also be redefined as general purpose input and output control signals.

SERIAL INTERFACE

The SIO can also be programmed to generate interrupts to signal to the processors the status of various transmit and receive conditions in the two communication channels. These include, receive data available, transmit data required, and the detection of various error conditions.

The interrupt structure of the SIO is such that on generating an interrupt to the CPU, it can also produce an interrupt vector internally, which defines the cause of the interrupt. This can then be used by the CPU to point to an associated service routine. The vector is obtained from the SIO by performing an interrupt acknowledge cycle.

The interrupt method of communication between the SIO and the CPU is the one adopted on the System Board. The SIO only generates an interrupt to the CPU when the device requires a specific servicing routine to be carried out, leaving the processors free to service the other devices on the board.

The SIO interrupt is not wired directly to the CPU but is supplied via the Interrupt Controller (PIC). The PIC provides a vector to the CPU to indicate the device generating the interrupt. If the device is the SIO, the actual process within the device requiring attention is then specified by the interrupt vector from the SIO.

SIO Architecture

Internally, the SIO consists of four areas of circuitry:

(a) A Processor interface.
(b) Internal interrupt control logic.
(c) The two independent full duplex serial communication channels, channel A and channel B.

The processor interface handles all communications via the data bus between the processors and a series of internal registers, associated with each of the two serial channels. These are of three different types; data registers, command registers and status registers.

Both channels have two data registers each, which are

accessed by the processors; one handles the transmit data from the processors (transmit buffer), and one stores receive data from the serial channel (receive buffer).

The mode and operation of each channel is determined by the contents of the command registers (Write Registers). These are initialised with control words prior to any data transfer over the two serial channels, and may be modified as data transfer operations proceed. Seven Write Registers are associated with each channel. An eighth Write Register, which is located in channel B is shared by both channels.

The information contained in the status registers (Read Registers) indicate the status within each channel. Two Read Registers are associated with each channel. A third register accessed via channel B is common to both channels.

The interrupt control logic section of the SIO assigns the priority to the various sources of interrupts, generates the interrupt output to the PIC and produces the interrupt vector. The priority of the interrupts is fixed with channel A always assigned a higher priority than channel B. The order of priority assigned within each channel is receiver interrupts (highest priority), followed by transmitter interrupts, followed by external/status interrupts (lowest priority).

Separate transmit and receive data paths are provided within the two serial communications channels.

The receiver ports are quadruply buffered by an input shift register and three storage registers. The shift register converts the incoming receive serial data into a parallel byte. The three storage registers are configured in a FIFO (first in first out) arrangement. The first parallel data byte from the shift register is loaded into the bottom of the FIFO stack and then shifted through to the top at a rate, determined by an internal clock.

The register at the top of the stack is the receive buffer which acts as the storage buffer for the receive character until read by the processors. Reading the character in the receive buffer automatically transfers the next character (if any) to the top of the stack.

SERIAL INTERFACE

Error flags associated with each receive character are also buffered by a similar arrangement. These are loaded into the register of a parallel receive error FIFO stack at the same time as the character. Each time the receive character is shifted through the character FIFO stack, the error flags are moved accordingly. The top of the receive error FIFO stack is one of the Read Registers.

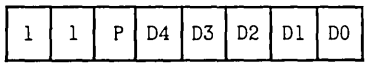
The contents of this Read Register reflect the status of the character stored in the receive buffer, but may also contain any receiver overrun and parity errors received from previous characters. These two error conditions remain within the status register, even when new character error flags are loaded, until cleared by an error reset command.

If the character FIFO stack is full (i.e. contains three characters) and the CPU fails to read the character within the receive buffer before another receive character is supplied to the stack, the last character placed in the stack is overwritten with the new character, and the receiver overrun flag is recorded in the corresponding register in the receive error FIFO stack. The first two characters in the stack are never overwritten, even if more characters are received; the last character is continuously overwritten.

Error conditions stored in the FIFO stack can be programmed to generate an interrupt to the CPU, on being loaded into the Read Register at the top of the stack.

In the asynchronous mode, using 8 bits per character, the serial receive data is stripped of the start, stop and parity bits, prior to being supplied to the serial shift register. For character lengths of less than 8 bits, the receiver circuits automatically insert logic 1's in the most significant places to assemble a byte of data, if a parity bit is not included with the character. If a parity bit is included, the parity bit is also assembled with the bits of the character, with any remaining bits set to logic 1. For example, a 5-bit receive

character with a parity bit is assembled in the receive buffer in the following format:



D = 5-bit character.
P = Parity bit.

In the byte oriented synchronous modes, Monosync and Bisync, receive data is not transferred to the receive FIFO register stack until character synchronisation is established. Detecting a byte of receive data which matches a sync character stored in a Write Register establishes synchronisation in Monosync. Detecting two consecutive bytes of receive data which match the sync characters stored in two of the Write Registers establishes synchronisation in Bisync.

Searching for the character sync is achieved by programming the channel to operate in the first phase of the synchronous reception process, termed the hunt phase. The second phase is the actual reception of data, where the receive data is automatically transferred to the FIFO stack, after detecting the character sync.

Also included in the receive path in synchronous modes, is a CRC error detection circuit which sets an error flag according to the result of CRC comparisons.

The receiver circuits in the bit oriented synchronous modes operate in a similar manner to the byte oriented modes, operating with two separate phases, a hunt phase and a receive phase. Receive data is not transferred to the receive FIFO register stack until the hunt phase is sucessfully completed. This requires detecting an opening flag sequence corresponding to a flag pattern, stored in one of the Write Registers. Reception is terminated on detecting the closing flag at the end of the message (EOM). This prevents any further data being supplied to the FIFO register stack. The receive data is also supplied to the CRC error detection circuits.

SERIAL INTERFACE

The transmitter section of the serial channels, consist of an 8-bit data register (the transmit buffer), supplied with character data from the processors, and a 20-bit transmit shift register. The shift register converts the parallel data in the transmit buffer into serial format and also performs a variety of other functions depending on the mode of operation.

In the asynchronous mode, the data from the transmit buffer is automatically formatted with start, stop and parity bits within the shift register, prior to transmission.

In Monosync and Bisync, the shift register is loaded with the sync characters stored in the Write Registers, at the beginning of the message, and then data from the transmit buffer as transmission proceeds. The shift register also supplies the serial data to the CRC generator, which produces the two byte CRC at the end of the message.

In the bit oriented modes, the shift register is loaded with the flags stored in the Write Register, at the beginning and end of the message. The shift register supplies the serial data to the CRC generator which generates the 2-byte CRC at the end of the data field, and to the transmit data output.

For character lengths of less than 8 bits, the characters sent to the transmit buffer have to be right justified, by the programmer. The logic state of the unused bits within each character byte (MSB) are immaterial for character lengths of 6 and 7 bits, since the extra bits are automatically ignored by the SIO. For a 5-bit character, the unused bits have to be programmed with logic 0's. Character lengths of less than 5 bits require a combination of logic 1's and logic 0's, to be inserted in the MSB as detailed in a later section.

Processor Interface

The connections to the SIO processor interface are detailed in the table "System Connections" at the end of the section. The interface handles the transfer of data, commands and status information (via the 8-bit bi-directional peripherals data bus), between the processors and the series of addressable registers within the SIO.

The system software views the SIO as four bi-directional peripheral ports, located in the system input/output space. The port addresses, as defined by the SIO select and the system address bus connections, are detailed below.

Address	Port
60H	Channel A transmit/receive data
62H	Channel A commands/status
64H	Channel B transmit/receive data
66H	Channel B commands/status

Writing data to the I/O address location 60H loads data into the transmit buffer of channel A; reading data from the same location accesses data stored in the receive buffer of channel A. Similarly, writing data to the I/O address location 64H loads data into the transmit buffer of channel B; reading data from the same location accesses data stored in the receive buffer of channel B.

Writing data to the Write Registers (command registers) and reading data from the Read Registers (status registers) in the two channels, generally requires two data transfers. The first transfer is a write to the commands/status address of the channel, and provides the SIO with a pointer to determine the register for the next read or write operation.

If the next operation is writing to the command/status address location, a command byte is loaded into the Write Register specified by the pointer. If the next operation is reading from the commands/status location, a status byte is accessed from the Read Register specified by the pointer.

Following the second transfer operation the pointer within the SIO is always cleared to zero.

Write Register Definition

Both channels contain seven Write Registers each, which configure the SIO to match the desired mode and application. Five of these registers are command registers

SERIAL INTERFACE

which define the basic mode of operation (e.g. asynchronous), and configuration of the channel within the mode (e.g. number of bits per character, etc.). The two other registers are sync character registers. An eighth register, which is written to via channel B is common to both channels and is programmed with the base address of the interrupt vector.

A summary of the functions of the Write Registers is provided in tabular format below. This is followed by a more detailed description of the commands for each individual register.

The only constraint on the order of programming the registers is that initialising each channel of the SIO following a hardware system reset or a software channel reset, requires the parameters of Write Register 4 to be issued before the parameters/commands of Write Registers 1, 3, 5, (6 and 7 if used).

As a general rule, if changing from one operational configuration to another (e.g. 8-bit asynchronous to 7-bit asynchronous), the whole initialisation sequence should be repeated with the new parameters.

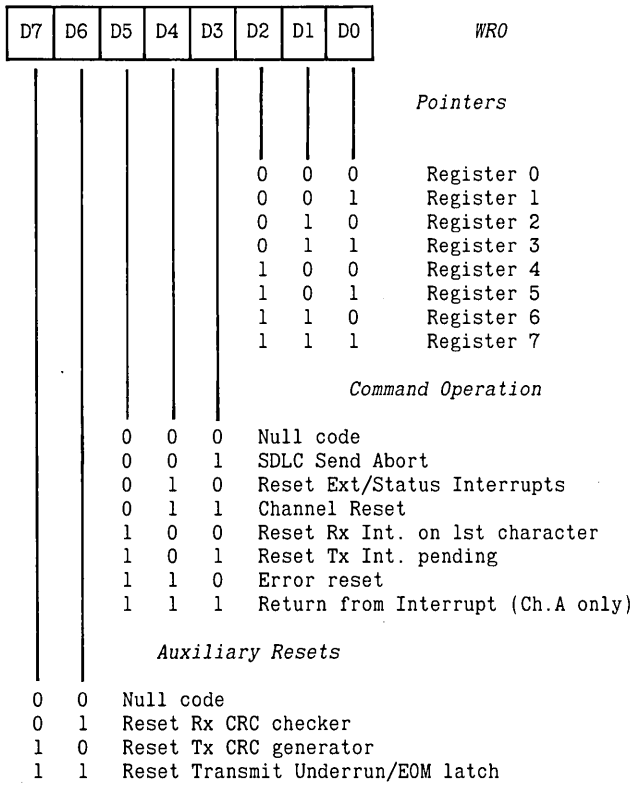
Write Register Summary

WR0	Write Register 0. Allows the programmer to perform a number of basic reset functions in addition to acting as the pointer to select the register for the next command/status transfer operation.
WR1	Write Register 1. Contains the bits which, select the interrupt mode, allow the interrupts to be enabled/masked, and also control the function of the DMA control output, W/RDY.
WR2	Write Register 2. This register is programmed with the base address of an interrupt vector and is addressed through channel B only.

WR3	Write Register 3. The bits written to this register configure and control the channel receiver circuitry.
WR4	Write Register 4. This register is programmed with bits which select the mode of operation (asynchronous or synchronous), and some of the operational parameters of the selected mode.
WR5	Write Register 5. The bits written to this register configure and control the channel transmit circuitry.
WR6	Write Register 6. This register is programmed with a sync character in Monosync and Bisync, and a check address in the bit oriented synchronous modes.
WR7	Write Register 7. This register is programmed with a sync character in Monosync and Bisync, and a flag character in the bit oriented synchronous modes.

SERIAL INTERFACE

Write Register 0.



Pointers (D0 to D2). These bits signify the register for the next command/status transfer operation. If the next operation is a write, the pointer specifies a write register. If the next operation is a read, the pointer specifies a read register. Following a read or write to any register (except WR0), the pointer points to register 0 (i.e. Write Register 0 or Read Register 0).

Commands (D3 to D5). These bits specify eight different commands, the function of which are detailed below.

1. Null Code. This command enables the programmer to specify the next register for a command/status transfer using the pointer bits, without affecting the operation of the SIO.

2. SDLC Send Abort. Used only in the bit oriented synchronous modes. The command causes the SIO to generate an abort sequence which informs the receiver of data from the SIO to terminate reception.

3. Reset Ext/Status Interrupts. After an External interrupt (caused by a change of state on a modem control input), or a Status interrupt (caused by detecting a break/abort/ condition in receive data or a transmit underrun/EOM condition in transmit data), corresponding status bits within Read Register 0 are latched. This command re-enables the bits and allows further interrupts to occur.

4. Channel Reset. This command performs the same function as a hardware reset but on the specified channel only. Issuing the command to channel A also resets the interrupt logic, clearing any current and all pending interrupts. All Write Registers in the channel must be reprogrammed, following the command.

5. Reset Rx Interrupt on First Character. If the Interrupt on First Receive Character Mode is in operation (bits 3 and 4 of WR1), an interrupt is generated on receiving the first character. At the end of the message, this command is issued to reactivate the mode.

6. Reset Tx Interrupt Pending. If the Transmit Interrupt is enabled (bit 1 of WR1), the SIO generates an interrupt every time the transmit buffer is empty. Issuing the Reset Tx Interrupt Pending command, resets the transmit interrupt, and prevents the interrupt being raised again until the transmit buffer is loaded with a character and becomes empty again.

7. Error Reset. Parity and Overrun errors are latched into Read Register 1 (the status register at the top of the receive error FIFO stack). These error conditions remain within the register until the Error Reset Command is issued.

8. Return from Interrupt. This command is used to signify to the SIO that the interrupt routine in service specified by the last interrupt vector supplied to the CPU has been completed. The command is issued through channel A only and resets the in-service interrupt within the SIO, and allows the highest priority interrupt pending (if any) to interrupt the CPU.

Auxiliary Resets (D6, D7). These bits specify four commands, the function of which are as described below.

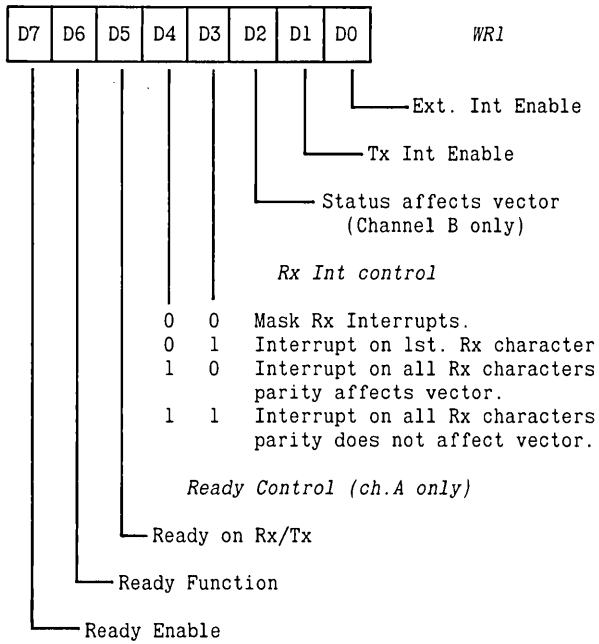
1. Null Code performs the same function as previously described for the Null Code of the Command bits.

2. Reset Rx CRC Checker. This command resets the CRC error detection circuitry located in the channel receiver circuit.

3. Reset Tx CRC Generator. This command resets the CRC generator located in the channel transmitter circuit.

4. Reset Tx Underrun/EOM. When the SIO detects either a Transmit Underrun condition or the End of a Message (EOM), a corresponding bit is set within Read Register 0 (bit 6). The bit is reset by issuing this command.

Write Register 1



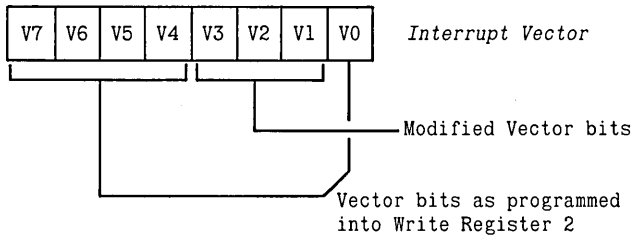
Ext. Int Enable (D0). The logic state of this bit determines whether the External/Status interrupts are enabled (logic high) or masked (logic low). If enabled, an interrupt is produced as a result of the following conditions.

1. Transitions on the modem control lines; $\overline{\text{DCD}}$, $\overline{\text{CTS}}$, or $\overline{\text{SYNC}}$.
2. Detecting a break or abort condition in the receive data.
3. At the start of transmission of either CRC or sync characters, in synchronous modes, when the Transmit Underrun/EOM latch becomes set.

Tx Int Enable (D1). The logic state on this bit determines whether the transmit interrupt is enabled (logic high) or masked (logic low). If enabled, an interrrupt is produced when the transmit buffer becomes empty.

SERIAL INTERFACE

Status Affects Vector (D2). This bit is active only in channel B. If set to logic low, the interrupt vector programmed in Write Register 2 is supplied to the CPU, during the interrupt acknowledge cycle. If set to logic high, the interrupt vector supplied to the CPU is modified according to the interrupting condition as detailed on the next page.



Modified Vector Bits

V3	V2	V1	Interrupt Condition
0	0	0	Ch.B Transmit buffer empty
0	0	1	Ch.B Ext/status Interrupt
0	1	0	Ch.B Receive Character Available
0	1	1	Ch.B Special Receive Condition *
1	0	0	Ch.A Transmit buffer empty
1	0	1	Ch.A Ext/status Interrupt
1	1	0	Ch.A Receive Character Available
1	1	1	Ch.A Special Receive Condition *

* Special Receive Conditions include Parity error, Rx Overrun Error, Framing Error, End of Frame (SDLC).

Rx Int Control bits (D3, D4). These two bits select the various receive conditions able to generate an interrupt.

1. Mask Rx Interrupts. Disables all receive interrupts.

2. Interrupt on 1st Rx Character. Enables an interrupt to be generated on detecting:

   (a) The first receive character in the receive buffer.
   (b) Any special receive condition.

3. Interrupt on all Rx Characters, parity affects vector. Enables an interrupt to be generated on detecting:

(a) Any receive character within the receive buffer.
(b) Any special receive condition.

4. Interrupt on all Rx Characters, parity does not affect vector. Enables an interrupt to be generated on detecting:

(a) Any receive character within the receive buffer.
(b) Any special receive condition apart from a Parity error.

Ready Controls (D5 to D7). These three bits control the state of the $\overline{W/RDYA}$ output of channel A, which is connected to the DMA request line DRQ2 of the Input Output Processor (IOP). A W/RDYB output is not provided on a Z80 SIO/0, so these bits are redundant in channel B. The function of the three bits are as detailed below.

1. Ready On Rx/Tx (D5). With the Ready Function and Ready Enable bits set to logic high, the condition for generating a DMA request is dependent on the state of the Ready On Rx/Tx bit, as follows:

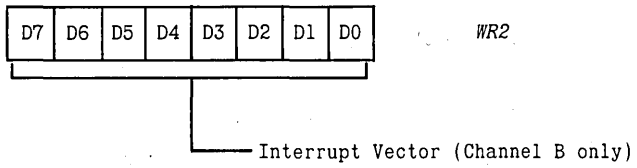
(a) When programmed to logic low, the DMA request line is set and reset according to whether the transmit buffer is empty or full. Every time the transmit buffer is empty, the SIO sets $\overline{W/RDYA}$ to logic low, which signifies to the IOP that transmit data is required. Every time the IOP responds by writing a new byte of data into the transmit buffer, the SIO resets the $\overline{W/RDYA}$ output to logic high.
(b) When programmed to logic high, the DMA request line is set and reset according to whether the receive buffer is empty or full. Every time the receive buffer is full, the SIO sets $\overline{W/RDYA}$ to logic low, which signifies to the IOP that receive data is available. Every time the IOP responds by reading the byte of data, the SIO resets the $\overline{W/RDYA}$ output to logic high.

SERIAL INTERFACE

2. Ready Function (D6). This bit determines whether the W/RDYA output functions as a Ready output (bit set to logic high) for DMA transfers or a Wait output (bit set to logic low) to the CPU. Since the output is wired to the IOP, the function required is Ready.

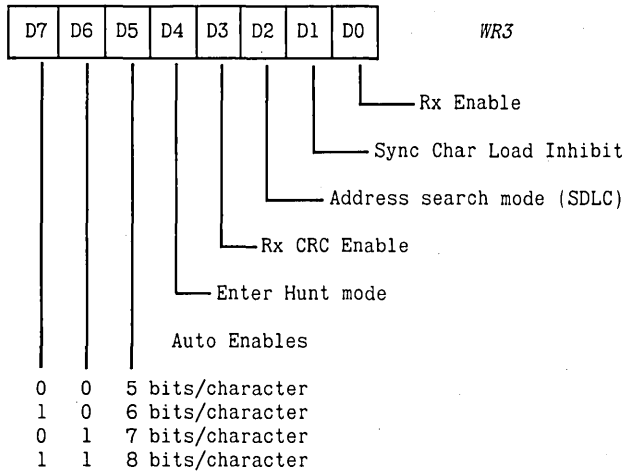
3. Ready Enable (D7). If this bit is set to logic high, the W/RDYA output is able to initiate DMA transfers as detailed above. If set to logic low, the state on the DMA request line is set permanently inactive (logic high).

Write Register 2



Write Register 2 is programmed with the base address for the interrupt vector supplied to the CPU during the interrupt acknowledge cycle. If the Status Affects Vector bit within Write Register 1 (WR1, D2) is set to logic high, the interrupt vector supplied to the CPU is the base address with the LSB modified as previously described. If the Status Affects Vector bit is set to logic low, the interrupt vector supplied to the CPU is the base address unmodified.

Write Register 3



Rx Enable (D0). If set to logic high, the receiver circuits are enabled to accept data from the serial link. If set to logic low, the receiver circuits are disabled.

Sync Character Load Inhibit (D1). Setting this bit to logic high prevents any sync characters being loaded into the receive FIFO stack.

Address Search Mode (D2). If a bit oriented synchronous mode is selected and this bit is set to logic high, only messages with an address (following the opening flag) matching either the programmed address in Write Register 6 or the global address (FFH), are accepted by the receiver circuits. If set to logic low, no address search is carried out.

Rx CRC Enable (D3). This bit enables (logic high)/disables (logic low) the receiver CRC error detection circuits.

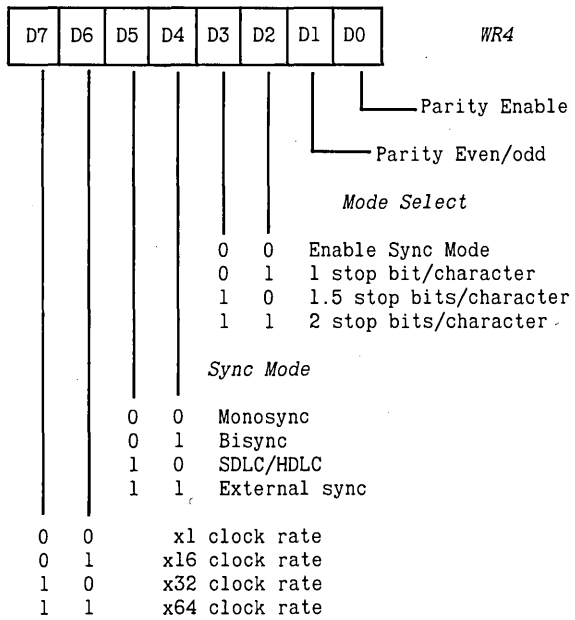
Enter Hunt Mode (D4). The Hunt mode in synchronous modes is automatically entered following a reset. If character synchronisation is lost (any Sync mode) or the incoming data is not required (SDLC), the hunt phase can be re-entered by setting D4 to logic high.

SERIAL INTERFACE

Auto Enables (D5). Setting this bit to logic high allows the two modem input control lines, DCD and CTS to control data transfers over the serial data link. DCD controls the SIO receiver circuits, CTS the transmitter circuits.

Rx bits/character (D6, D7). The combination of these two bits determine the number of receive bits assembled to form a character.

Write Register 4



Parity Enable (D0). If this bit is set to logic high, a parity bit is added to the transmit character data and is expected in receive data.

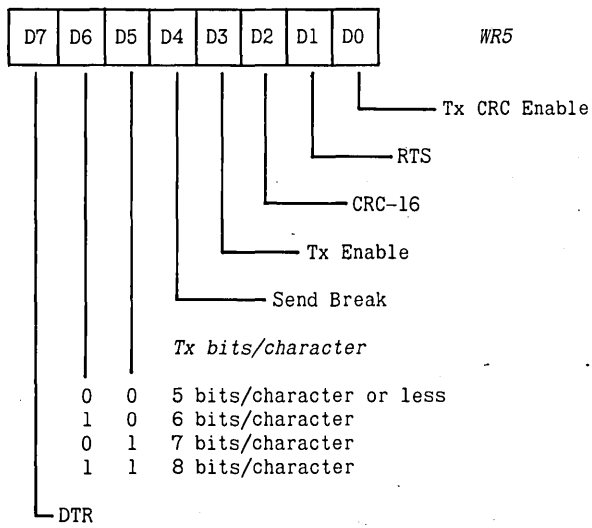
Parity Even/Odd (D1). This bit is used when the Parity Enable bit is set to logic high to determine the sense of the parity code in the transmit data and the expected parity code in receive data. Even parity is signified by setting this bit to logic high, odd parity by setting the bit to logic low.

Mode Select (D2, D3). The combination of these two bits differentiate between asynchronous and synchronous modes of operation and also specify the number of stop bits added to the transmit data in the asynchronous mode. The receiver circuits always check for one stop bit in the receive data, regardless of the number of stop bits in the transmit data.

Sync Mode (D4, D5). If synchronous mode is selected by the Mode Select bits, these bits determine the type of synchronous mode, for transmit and receive data.

Clock Rate (D6, D7). These bits specify the multiplier applied to both the transmit and receiver input clock rate prior to being used to set the transmit and receive baud rates. For synchronous modes the x1 clock rate must be selected. Any rate may be selected for the asynchronous mode, apart from the x1 rate.

Write Register 5



Tx CRC Enable (D0). This bit enables (logic high)/disables (logic low) the CRC generator in the transmit data path.

SERIAL INTERFACE

RTS (D1). This bit controls the state of the modem control output Request to Send ($\overline{RTS}$). When the RTS bit is set to logic high, the $\overline{RTS}$ output is set active (logic low/line spacing). When the RTS bit is reset to logic low, the $\overline{RTS}$ output is reset to the inactive logic high state (line marking). In asynchronous mode, the $\overline{RTS}$ output is not reset by setting the RTS bit low, until after transmission of the character in the transmit buffer.

CRC-16 (D2). This bit selects the CRC polynomial used by the transmitter and receiver circuits. If set to logic high, the CRC-16 polynomial ($X^{16} + X^{12} + X^5 + 1$) is selected. If set to logic low, the SDLC polynomial ($X^{16} + X^{15} + X^2 + 1$) is selected.

Tx Enable (D3). This bit acts as the enable/disable control signal for the transmitter output. Character data in the transmit buffer cannot be transmitted or a break condition sent unless this bit is set to logic high. If set to logic low, the transmit output is held in the line idle marking condition.

Send Break (D4). When set to logic high and the transmitter is enabled, the transmit data output is forced to the spacing condition, regardless of any data transmission in progress.

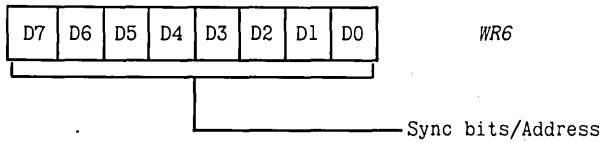
Tx bits/character (D5, D6). The combination of these two bits specify the number of bits per character in the transmit data. Characters have to be assembled into the correct format by the programmer. All character lengths of less than 8 bits have to be right justified, with the following format.

D7	D6	D5	D4	D3	D2	D1	D0	Bits/Character
0	D	D	D	D	D	D	D	7
0	0	D	D	D	D	D	D	6
0	0	0	D	D	D	D	D	5
1	0	0	0	D	D	D	D	4
1	1	0	0	0	D	D	D	3
1	1	1	0	0	0	D	D	2
1	1	1	1	0	0	0	D	1

D = Character data bits

DTR (D7). This bit controls the modem control output Data Terminal Ready ($\overline{DTR}$). When the DTR bit is set to logic high, the $\overline{DTR}$ output is set to the active state (logic low). When reset to logic low, the $\overline{DTR}$ output is reset to the inactive logic high state.

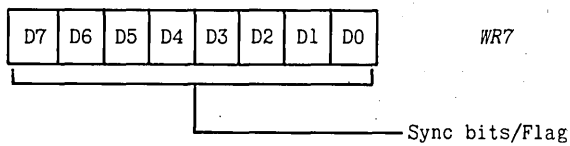
Write Register 6



Write Register 6 is programmed with:

(a) The transmit sync character in Monosync.
(b) The 8 LSB of the 16-bit sync character in Bisync.
(c) The 8-bit frame address byte in bit oriented modes.

Write Register 7



Write Register 7 is programmed with:

(a) The receive sync character in Monosync.
(b) The 8 MSB of the 16-bit sync character in Bisync.
(c) A flag character in bit oriented modes.

Read Register Definition

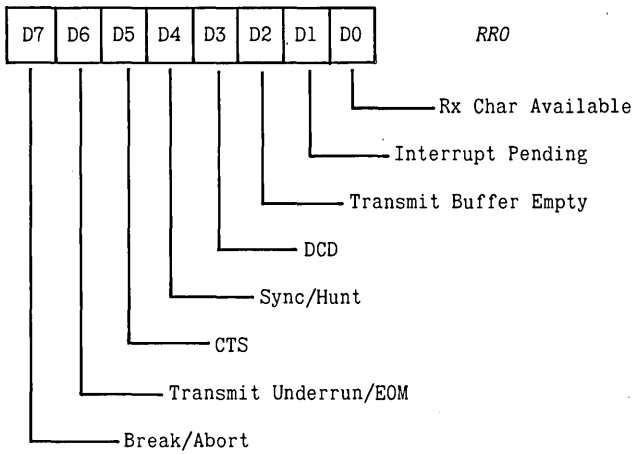
Both channels contain two Read Registers each (RR0 and RR1), which define the status within the channel. A third Read Register (RR2) accessed through channel B contains a copy of the interrupt vector, which indicates the SIO interrupt routine (if any), currently in service.

SERIAL INTERFACE

Reading the contents of Read Registers 1 and 2 requires a two byte transfer operation. The first, a write operation to the commands/status address location using the pointer bits of Write Register 0 to specify the Read Register; the second, the actual read operation from the same address to access the contents of the register. Following any read or write operation (apart from writing to Write Register 0), the address pointer within the SIO is always cleared to zero, allowing the contents of Read Register 0 to be accessed using a single read operation.

The definition of each register is detailed below.

Read Register 0



Rx Char Available (D0). This bit is held at logic high, until all characters within the receive FIFO stack, are read. Logic low indicates that no more receive characters are available.

Interrupt Pending (D1). This bit only has significance in channel A. In channel B, the bit is always at logic low. When set to logic high, the bit indicates that an interrupt condition is present within the SIO.

Transmit Buffer Empty (D2). This bit is set to logic high every time a character is transmitted out of the transmit

buffer. It is reset to the logic low inactive state by reloading the transmit buffer with a new data character.

DCD (D3). The DCD bit indicates the state of the modem control input Data Carrier Detect ($\overline{DCD}$). This state is latched every time any external/status interrupt condition occurs, and remains in the latched state until reset by writing the reset external/status interrupt command to Write Register 0. Therefore, to ensure that the current state of the $\overline{DCD}$ input is obtained, the bit should be read, immediately following a reset external/status interrupt command. The DCD bit indicates the inverse of the state on the $\overline{DCD}$ input.

Sync/Hunt (D4). In asynchronous modes, this bit indicates the state of the Sync input and operates in a similar manner to the DCD bit, with regard to the latching process. The Sync bit indicates the inverse of the state on the $\overline{Sync}$ input. In synchronous modes, the bit reflects the phase of the synchronous receive operation. Initiating the Hunt phase causes the bit to be set to logic high. (i.e. On reset or setting the enter hunt mode bit in Write Register 3). On achieving character synchronisation, the bit is set to logic low as the receive phase begins and remains in this condition unless the Hunt phase is initialised again.

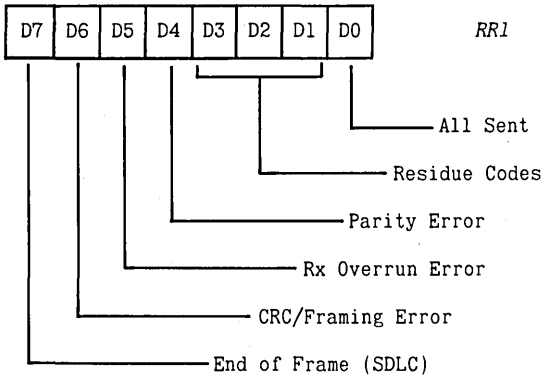
CTS (D5). This bit functions in a similar manner to the DCD bit with regard to the latching process, but indicates the inverse of the state on the Clear to Send input ($\overline{CTS}$).

Transmit Underrun/EOM (D6). In synchronous modes, the bit is set to logic high following a system/channel reset, allowing sync/flag characters to be sent when the transmit buffer becomes empty. When the reset transmit underrun/ EOM command is issued to Write Register 0, the transmit underrun/EOM bit is set to logic low. This enables CRC characters to be automatically sent instead of the sync/flag characters.

Break/Abort (D7). In asynchronous modes, this bit is set to logic high, when a break is detected in the receive data. The bit is not reset until, a reset external/status command is issued to Write Register 0 and the break condition is

removed. In the bit oriented modes, the bit is set to logic high on detecting an abort sequence in the receive data. The bit is reset to logic low on loading Write Register 0 with the reset external/status command. The bit is not used in the byte oriented synchronous modes.

Read Register 1



All Sent (D0). In asynchronous modes, this bit is set to logic high when all the bits of the character have been transmitted onto the serial link. In synchronous modes the bit is permanently set to logic high.

Residue Codes (D1 to D3). The combination of these three bits indicate the length of the I-field in the bit oriented modes where the I-field is not an integral multiple of the character length.

Parity Error (D4). When parity is enabled, this bit is set to logic high on detecting a receive character whose parity does not match the sense programmed by bit 1 of Write Register 4. The bit remains set in the error condition until reset by loading the error reset command into Write Register 0.

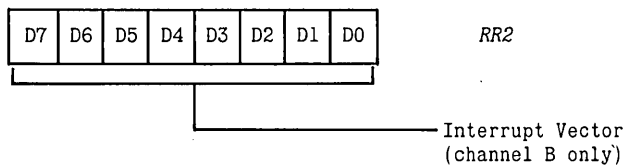
Rx Overrun Error (D5). This bit is set to logic high when one or more receive characters have been overwritten in the receive FIFO buffer. The bit remains set in the error

condition until reset by loading the error reset command into Write Register 0.

CRC/Framing Error (D6). The function of this bit is dependent on the mode selected. In asynchronous modes, the bit is set to logic high on detecting a receive character with incorrect stop bits (framing error). The error condition only persists for the particular character stored in the receive buffer. In synchronous modes, the bit indicates the result from the receive CRC error detection circuit. A logic high indicates a CRC error. The error conditions are reset to the inactive low state after issuing the error reset command to Write Register 0.

End of Frame (D7). In the bit oriented modes, this bit indicates that a valid closing flag has been detected and the CRC Error and residue codes are now valid. The bit is reset by issuing an error reset command to WR0.

Read Register 2



Read Register 2 contains the interrupt vector and is read through channel B only. If the status affect vector bit is set (D2, Write Register 1), the register indicates the current interrupt service routine (if any) in operation, and is a copy of the interrupt vector supplied to the CPU. If no interrupts are pending, the vector is set to the condition for a special receive condition in channel B (see Write Register 1 description). If the status affect vector bit is not set (logic low), the register contains a copy of the vector written into Write Register 2 in channel B.

SERIAL INTERFACE

SIO Interrupt Sequence

The SIO incorporates an elaborate interrupt structure, which acts as an extension to the interrupt structure provided by the Interrupt Controller (PIC) on the System Board. A single interrupt output line ($\overline{INT}$) connects the SIO to the PIC.

Each channel within the SIO is able to generate an interrupt for a variety of conditions. All interrupting sources can be enabled or disabled (masked) by software, and are ordered on a priority basis. All sources within Channel A are assigned a higher priority than Channel B. Within each channel, the assigned priority is as detailed below:

1. Special receive condition (Highest).
2. Receive character available.
3. External/status interrupt.
4. Transmit data required (Lowest).

If a SIO interrupt is in service when a higher priority interrupt condition occurs, the higher priority condition is granted service. The SIO stores the lower priority condition and resumes the interrupt service routine on completion of the higher priority interrupt routine.

A special receive interrupt is generated when the SIO detects any of the following conditions in the receive data:

1. Parity errors.
2. Receiver overrun.
3. Framing errors (asynchronous modes).
4. End of Frame (SDLC/HDLC only).

Receive character available interrupts can be programmed to generate an interrupt on every receive character or for the first character only (synchronous modes).

An external/status interrupt can be caused by any of the following conditions:

1. Transitions on the modem control lines; $\overline{DCD}$, $\overline{CTS}$ and $\overline{SYNC}$.
2. A break in receive data (asynchronous modes).
3. An abort sequence in receive data (bit synchronous modes).
4. Transmit underrun/EOM condition in transmit data (synchronous modes).

The actual sequence of events performed on the occurrence of a SIO interrupt condition is as follows.

The SIO compares the new interrupt with any interrupt currently in service. If the new interrupt is of a higher priority than the current interrupt, the $\overline{INT}$ output is set active (logic low). If the new interrupt is of a lower priority, the interrupt condition is stored until it becomes the highest priority.

The $\overline{INT}$ output, after inversion, forms the interrupt request line (IR5) to the PIC. Providing IR5 is the highest priority unmasked interrupt request to the PIC, an interrupt is issued to the CPU. The CPU determines the device generating the interrupt by performing a PIC interrupt acknowledge sequence, which then provides an associated interrupt vector.

For the CPU to determine the SIO process generating the interrupt, the CPU has to perform a special SIO interrupt acknowledge sequence. This is achieved by issuing a "locked" read cycle, as defined by the following 8086 instruction:

```
LOCK IN AL,60H
```

This instruction accesses the SIO interrupt vector, which is a base address previously supplied to the SIO, modified according to the condition generating the interrupt. The SIO interrupt, thus allows the CPU to select the appropriate interrupt service routine. On completion of the SIO service routine, the CPU has to issue a return from interrupt

command (to WR0), to enable any lesser priority pending interrupts to generate an active interrupt output.

The SIO is able to produce eight different interrupt vectors according to the type of interrupt as detailed on the previous page (4 vectors for each channel). In the case where the interrupt can be caused by a number of different conditions (i.e. special receive and external/status), the actual cause can be detected by reading the appropriate status register.

KEYBOARD COMMUNICATIONS

General

Transfer of data between the keyboard and the processors is supplied via channel B of the SIO. The channel is programmed to support full duplex asynchronous communications at a fixed data rate (approximately 7800 bits per second). The format of the data over the serial link is as follows:

Transmit data to the keyboard	Receive data from the keyboard
8 bits per character.	8 bits per character.
1.5 stop bits.	1 stop bit.
No parity.	No parity.

No handshaking signals are provided between the Keyboard and the System Unit. Control of data flow over the serial link to the keyboard is controlled by character codes from the Keyboard. The Keyboard supplies a "transmit off" character to the System Unit when its receive buffer stack is nearly full. The SIO can then only transmit four more characters without the Keyboard receive buffer overflowing, resulting in the loss of data. When the Keyboard receive buffer stack becomes empty following the transmission of a "transmit off" character, the Keyboard transmits a "transmit on" character to inform the System Unit that data transmission can resume.

There is no control of data flow over the serial link from the Keyboard. To prevent loss of data, the processors have to read the keyboard data before it is overwritten in the receive FIFO stack of channel B. This requires the software to be capable of processing keyboard data stored in the SIO receive buffer at the maximum possible receive data rate of approximately 780 characters per second.

A description of all the possible character codes and their significance, supplied between the Keyboard and the system software is detailed in the Keyboard documentation.

In addition to carrying character codes, the data in line from the Keyboard also carries the system hardware reset. Activating the reset overrides the transmission of any data to the SIO by holding the data in line to the spacing condition. The line idle condition in both directions is indicated by continuous marking.

Keyboard Connection Detail

The data link between the Keyboard and the System Unit is provided via 9-pin D-type connectors and a 5-wire cable assembly. The cable supports the full duplex communications channel and also provides + 12V and –12V supply voltages to the Keyboard. The definition of each pin of the D-type connector is detailed in the table below.

The serial data is transferred over the keyboard link at V28 levels. These are defined as follows:

1. Line marking; Between –3V and –15V relative to 0V, (typically –11V).
2. Line spacing; Between +3V and +15V relative to 0V, (typically +11V).

SERIAL INTERFACE

Keyboard Connector Pin Definition

Pin	Description
1	+12V Out via a 4.7 ohm series resistor.
2	Serial Data Out.
3	Serial Data In.
4	0V.
5	N.C.
6	Frame Ground.
7	–12V Out via an 100 ohm series resistor.
8	0V.
9	N.C.

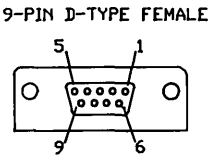


Figure 2. Keyboard Connector Detail

Channel B Programming Details

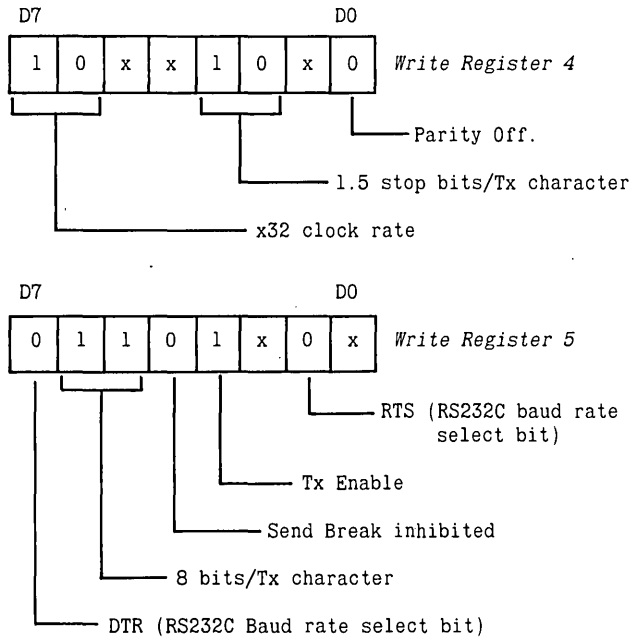
In addition to providing a series of Write Registers to configure the SIO to interface with the keyboard, a number of other parameters and control features are also programmed through channel B. These include:

(a) The base address for the SIO interrupt vector.
(b) Enabling and disabling of the Parallel Interface interrupt control input lines.
(c) Selection control for the baud rates for the RS232C channel.

The parameters issued to the Write Registers of channel B for interfacing the SIO to the Keyboard, and the interrupt vector base address are programmed during an initialisation routine by the BIOS. These are fixed and never require any modification by the software. The control bits for the Parallel Interface interrupts and selecting the baud rates for the RS232C channel are set to default states during

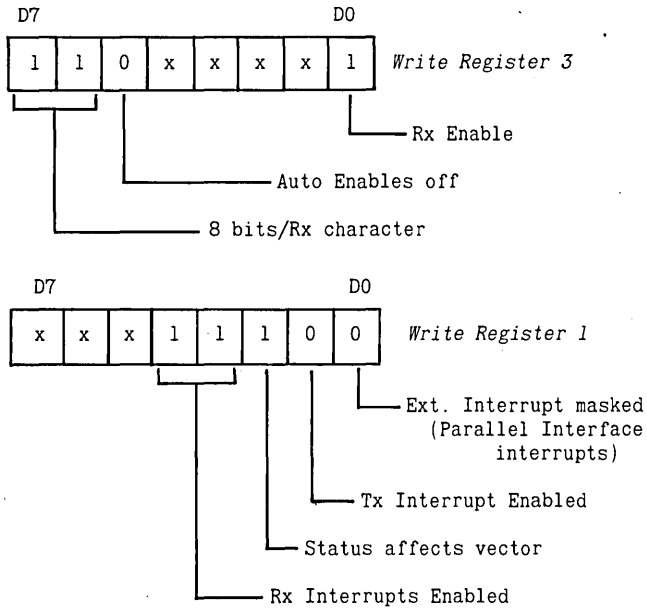
initialisation. Changing these under software control are described in subsequent sections.

Once channel B is initialised with the keyboard parameters and the base address for the interrupt vector (00H to Write Register 2), transfer of data between the Keyboard and the processors can proceed. The format of the command bytes supplied to the Write Registers of channel B during initialisation are detailed below. The address locations utilised by channel B are described in a previous section, under the heading "Processor Interface".



x = not applicable to mode and application, program to 0.

SERIAL INTERFACE



PARALLEL INTERFACE INTERRUPTS

Three printer control output lines from the Parallel Interface (Fault, Acknowledge and Busy) are wired to the SIO channel B modem control inputs (SYNCB, CTSB and DCDB respectively). These inputs can be programmed to generate an interrupt to the CPU, every time a transition occurs on any of the printer lines. The logic state on all three printer inputs are also indicated by a corresponding bit within Read Register 0.

The logic states of the bits in the register can only be regarded as reflecting the true state of the printer lines, if prior to reading the register, a reset external/status command is issued to Write Register 0. The bits within the register indicate the inverse of the logic state on the control input.

To enable transitions on the three inputs to generate an interrupt, requires the external/status interrupt bit originally set to logic 0 during channel B initialisation, to be updated. This is achieved by reprogramming bit D0 of Write Register 1 to logic 1.

The corresponding interrupt vector accessed by the CPU during an interrupt acknowledge cycle associated with the printer control lines, is the channel B ext/status interrupt vector (02H), as described in the section on the Write Registers. The other conditions, which are specified as being able to generate an external/status interrupt, are not enabled in channel B.

RS232C COMMUNICATIONS

General

The interface to the RS232C serial link is implemented by channel A of the SIO, which can be programmed to operate in either asynchronous or synchronous communications modes with transmit and receive baud rates determined either via the Programmable Interval Timer (TMR), or via the external data communications equipment.

The clock inputs to the SIO pins TxCA and RxCA, are used to determine the transmit and receive baud rates for the RS232C link and are supplied via the data selector. The select lines to the data selector are provided by two general purpose outputs (DTRB and RTSB) available through channel B, which are controlled by software. This allows the programmer to select:

(a) Different rates for transmit and receive, with the receive rate determined by the clock output OUT1 of the Interval Timer and the transmit rate, determined by OUT2.
(b) The same rate for transmit and receive as determined by OUT2 of the timer.
(c) The same rate for transmit and receive as determined by the external data communications equipment (RS232C connector pin, Rx Clock).

Details of the timer and the appropriate programming values to produce the majority of the commonly used baud rates are detailed in the TIMER chapter. These values apply to asynchronous communications only and require channel A to be programmed in the x16 clock rate mode to achieve

SERIAL INTERFACE

the correct baud rate. Three of the values detailed in the TIMER chapter produce a baud rate value which is marginally outside the +/– 5% tolerance normally allowed for communications over an RS232C link. These values, 3600, 7200 and 19200 bauds should therefore be used only for Apricot-to-Apricot communications.

The selected transmit rate clock (TxCA) is also supplied to the RS232C connector (connector pin, Tx clock), for use by the external data communications equipment.

Five of the most commonly used RS232C modem control lines are connected to various inputs and outputs of channel A of the SIO, available for coordinating data transfers over the serial link. These include the outputs RTS and DTR, and the inputs DSR, CTS and DCD. Use of each individual modem control line is dependent on compatible facilities within the external equipment. Generation of the output modem control lines and status monitoring of the input modem control lines is directly under program control. The SIO can be programmed to generate an interrupt to the CPU every time a transition occurs on any of the input control lines. DSR is not available in synchronous modes.

RS232C Connector Detail

The definition of the available RS232C connections provided by the 25-way D-type connector are detailed in the following table. The RS232C voltage levels are defined below. The line idle condition is indicated by continuous marking.

1. Line marking; Between –3V and –15V relative to 0V.
2. Line spacing; Between +3V and +15V relative to 0V.

RS232C Connector Pin Definition

Pin	Description	Pin	Description
1	Frame Ground.	9 to 14	N.C.
2	Tx Data.	15	Rx Clock.
3	Rx Data.	16 to 19	N.C
4	RTS.	20	DTR.
5	CTS.	21 to 23	N.C.
6	DSR.	24	Tx Clock
7	0V (Signal GND).	25	N.C.
8	DCD.		

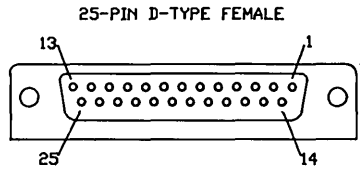


Figure 3. RS232C Connector Detail.

Programming Details

The following paragraphs detail the format of the Write Registers for setting channel A to operate in the asynchronous mode and also the various options available for controlling the transfer of data over the serial link. For details of programming the channel for synchronous communications, reference should be made to the appropriate Zilog documentation.

The address locations utilised by channel A and the method for writing data to the Write Registers are described in a previous section under the heading "Processor Interface".

SERIAL INTERFACE

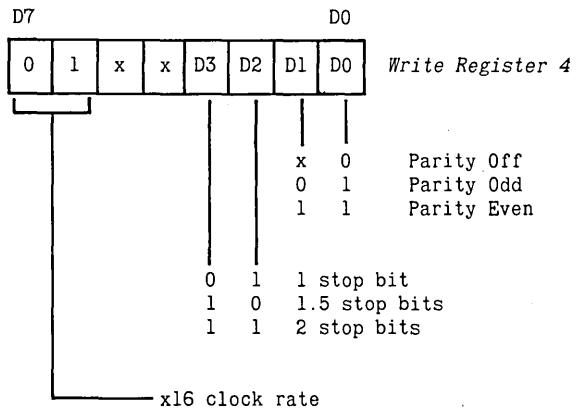
To receive and transmit data in the asynchronous mode, the following conditions have to be determined:

1. The transmit and receive baud rates.
2. The character length.
3. The number of stop bits in the transmit character.(1 stop bit is always expected in the receive data).
4. The sense of the parity, if any.
5. The interrupt mode.
6. The line protocol.

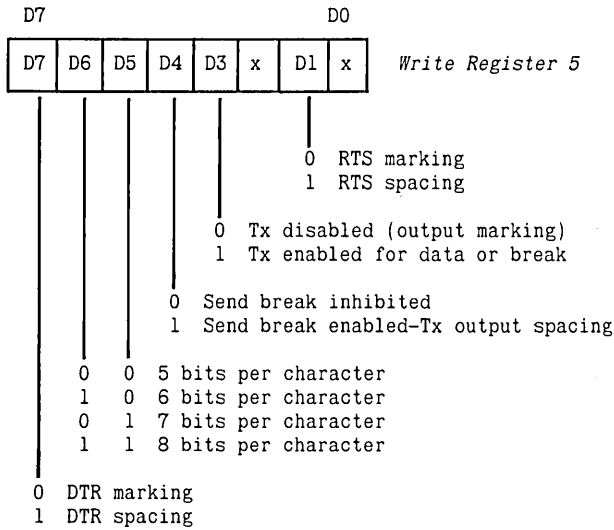
Baud Rate Selection. The clock source for determining the transmit and receive baud rates is selected according to the programmed state of the RTS and DTR bits of Write Register 5 in channel **B**, as follows.

RTS	DTR	Rx Clock	Tx Clock
0	0	OUT1 (TMR)	OUT2 (TMR)
0	1	External Clock	External Clock
1	0	OUT2 (TMR)	OUT2 (TMR)
1	1	0V	0V

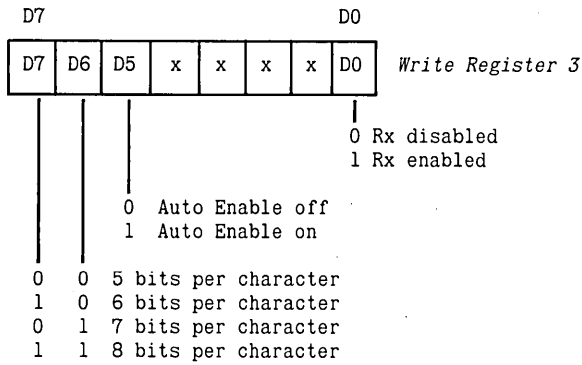
Write Register 4. This register determines, the number of stop bits in the transmit data, the sense of the parity (if any) in the transmit data and the expected parity in the receive data, the multiplier applied to both the transmit and receive input clock rates prior to setting the baud rates.



Write Register 5. This register determines the transmit character length, controls the modem outputs RTS and DTR, allows the programmer to enable/disable transmission, and also generate a break in transmission.

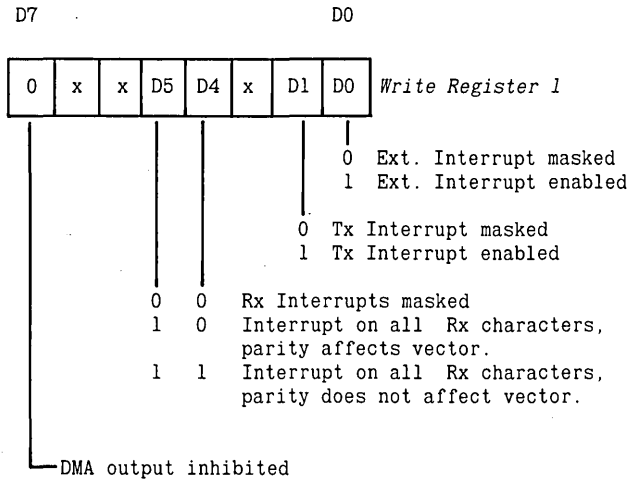


Write Register 3. This register determines the number of receive bits assembled to form a character, allows the programmer to enable/disable reception and also selects whether the modem control lines CTS and DCD control data transmission and reception over the serial link (Auto Enable).



SERIAL INTERFACE

Write Register 1. This register enables/disables the transmit, receive and external/status interrupts and also controls the DMA request output.



SIO PIN DETAIL

The two SIO communication channels are designated channel A (used for the RS232C link) and channel B (used for the keyboard link). On the block diagram at the beginning of the chapter, connections to channel A are denoted by the suffix A (e.g. TxDA), and to channel B by the suffix B (e.g. RxDB). All the remaining inputs and outputs of the SIO are connections to the processors. A description of each pin is detailed on the following pages.

System Connections

D0 to D7	Data bus, used to transfer data and commands between the processors and the SIO.
$\overline{IORQ}$	Input/Output Request. Control input, active state logic low, derived by combining the read and write commands from the system control bus ($\overline{AIOWC}$ and $\overline{IORC}$) using an AND gate. Used in conjunction with $\overline{RD}$ and the address inputs B/$\overline{A}$, C/$\overline{D}$ and $\overline{CE}$ to control the transfer of data and commands between the SIO and the procesors. An active state on $\overline{IORQ}$ indicates a transfer operation on the data bus; $\overline{RD}$ signifies the direction of data transfer (see below). $\overline{IORQ}$ also has another function, when set active at the same time as $\overline{M1}$ is set active. This condition acts as an acknowledgement of a SIO interrupt, causing the SIO to release an interrupt vector onto the data bus.
$\overline{M1}$	Machine Cycle 1. Control input, active state logic low. Function as described above.
$\overline{RD}$	Read. Control input. Used in conjunction with IORQ and the address inputs B/$\overline{A}$, C/$\overline{D}$ and $\overline{CE}$ to control the transfer of data and commands between the SIO and processing elements. If both $\overline{IORQ}$ and $\overline{RD}$ are set low and a valid address is set up, the direction of data/command transfer is from the SIO (read operation). If $\overline{IORQ}$ is set low and $\overline{RD}$ is set high, with a valid address set up, the direction of data/command transfer is to the SIO (write operation).
$\overline{CE}$	Chip Enable. Address input, active state logic low. When active, indicates that the SIO is selected for a data/command transfer operation.

continued ....

SERIAL INTERFACE

B/$\overline{A}$	Channel B/Channel A select. Address input. A logic low on B/$\overline{A}$ with $\overline{CE}$ also set low indicates that the data/command transfer operation involves channel A. A logic high on B/$\overline{A}$ with $\overline{CE}$ set low indicates that the data/command transfer operation involves channel B.
C/$\overline{D}$	Control/Data select. Address input. A logic low on C/$\overline{D}$ with $\overline{CE}$ also set low indicates that the information on the data bus is interpreted as data. A logic high on C/$\overline{D}$ with $\overline{CE}$ set low indicates that the information on the data bus is interpreted as commands.
CLK	System clock input. 2.5 MHz clock with a 50% duty cycle for internal timing within the SIO.
$\overline{RESET}$	System reset. Input, active low. When active, disables the receive and transmit sections of both channels and sets the transmit lines to the marking condition, requiring the registers within the SIO to be re-initialized before performing data transfer operations.
$\overline{INT}$	Interrupt request. Output, active low. Set active when an interrupt condition is detected internally within the SIO.

Channel A Connections

TxDA	Transmit Data. Serial data output to the RS232C interface via a line driver. A mark is indicated by logic high on TxDA and a space, by logic low.
RxDA	Receive Data. Serial data input from the RS232C interface via a line receiver. A mark is indicated by a logic high and a space, by logic low.

continued ....

TxCA	Transmitter Clock. Input used to determine the transmit baud rate of the RS232C channel. When operating in asynchronous mode, a facility exists within the SIO to divide the input clock frequency internally, prior to being used to set the baud rate. The divider is controlled by software and allows serial data to be transmitted at a rate of 1, 1/16th, 1/32nd or 1/64th of the rate supplied to TxCA. In synchronous modes, this facility is not available, so that TxCA corresponds directly to the transmit baud rate.
RxCA	Receiver Clock. Input used to set the receive baud rate of the internal data receiver circuitry. In asynchronous modes, the receiver clock is divided by the same divisor programmed for the transmitter clock. In synchronous modes, RxCA directly sets the receive baud rate of the data receiver circuitry.
$\overline{\text{DCDA}}$	Data Carrier Detect. Control input from the RS232C interface via a line receiver. When used with a modem, the active state (logic low) indicates that the modem has data to send. $\overline{\text{DCDA}}$ can be programmed to operate in one of two modes. The first mode allows $\overline{\text{DCDA}}$ to act as the enable control line to the receive section of channel A. The second mode sets a control bit within an internal register according to the state of $\overline{\text{DCDA}}$ and also has the facility to generate an interrupt request to the Interrupt Controller, every $\overline{\text{DCDA}}$ transition.
$\overline{\text{DTRA}}$	Data Terminal Ready. Control output to the RS232C interface via a line driver. When used with a modem, the active state (logic low) indicates that the SIO is ready to start handling data. $\overline{\text{DTRA}}$ is controlled by software, following the logic state of an associated control bit within an internal register.

continued ....

SERIAL INTERFACE

$\overline{\text{SYNCA}}$	Synchronization. Control input from the RS232C interface via a line driver. When used with a modem, the active state (logic low) indicates that the modem is ready to start handling data. This input is only available in the asynchronous mode. $\overline{\text{SYNCA}}$ sets an associated control bit within an internal register according to the state of SYNCA, and also has the facility to generate an interrupt request to the Interrupt Controller, every time a transition occurs on $\overline{\text{SYNCA}}$.
$\overline{\text{RTSA}}$	Request to send. Control output to the RS232C interface via a line driver. When used with a modem, the active state (logic low) indicates that the SIO has data ready to send. $\overline{\text{RTSA}}$ is controlled by software, following the logic state of an associated control bit within an internal register.
$\overline{\text{CTSA}}$	Clear to send. Control input from the RS232C interface via a line receiver. When used with a modem, the active state (logic low) indicates that the modem is ready to receive data. $\overline{\text{CTSA}}$ can be programmed to operate in one of two modes. The first mode allows $\overline{\text{CTSA}}$ to act as the enable control line to the transmit section of channel A. The second mode sets an associated control bit within an internal register according to the state of $\overline{\text{CTSA}}$ and also has the facility to generate an interrupt request to the Interrupt Controller, every time a transition occurs on $\overline{\text{CTSA}}$.
$\overline{\text{W/RDYA}}$	Wait/Ready A. Control output to the DMA request line of the Input Output Processor (IOP DRQ2). When active (logic low), indicates to the IOP that the SIO is ready for a DMA transfer operation to/from the SIO.

Channel B Connections

TxDB	Transmit Data. Serial data output to the Keyboard via a line driver. A mark is indicated by logic high on TxDB and a space, by logic low.
RxDB	Receive Data. Serial data input from the Keyboard via a line receiver. A mark is indicated by logic high on RxDB and a space, by logic low.
RxTxCB	Receiver/Transmitter Clock. Fixed frequency clock input of 0.25 MHz (50% duty cycle), used to determine the transmit baud rate of the data to the keyboard and the receive baud rate of the internal receiver circuitry. The clock frequency is divided internally by 32 to match the Keyboard baud rate of 7.8125 kHz.
$\overline{\text{RTSB}}$	Request to send. Used as a general purpose output in conjunction with $\overline{\text{DTRB}}$ to select the clock source for the transmit and receive clock inputs of channel A (TxCA and RxCA). The logic state on $\overline{\text{RTSB}}$ is controlled by software, following the state of an associated control bit within an internal register.
$\overline{\text{DTRB}}$	Data terminal ready. Used as a general purpose output in conjunction with $\overline{\text{RTSB}}$ to select the clock source for the transmit and receive clock inputs of channel A. The logic state on $\overline{\text{DTRB}}$ is controlled by software, following the state of an associated control bit within an internal register.

continued ....

SERIAL INTERFACE

$\overline{\text{SYNCB}}$	Synchronisation. Control input connected to the $\overline{\text{Fault}}$ control line of the Parallel Interface. Input programmed to set an associated control bit within an internal register according to the state of $\overline{\text{SYNCB}}$, and also has the facility to generate an interrupt request to the Interrupt Controller, every time a transition occurs on $\overline{\text{SYNCB}}$.
$\overline{\text{CTSB}}$	Clear to send. Control input connected to the $\overline{\text{ACK}}$ control line of the Parallel Interface. Input programmed in an identical manner to $\overline{\text{SYNCB}}$.
$\overline{\text{DCDB}}$	Data carrier detect. Control input connected to the Busy control line of the Parallel Interface. Input programmed in an identical manner to $\overline{\text{SYNCB}}$.

PARALLEL INTERFACE

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The Parallel Interface is located on the System Board and consists of the elements of circuitry as detailed on the block diagram Figure 1. The interface performs the following functions within the equipment:

(a) Provides the communications interface between the equipment and an external printer, via the Centronics connector.
(b) Allows the processors to detect the active display period on the screen of the Display Unit, by utilising a control output from the CRT control circuitry.
(c) Produces a series of control outputs to various areas of circuitry, under software control.

The control outputs are as follows:

(a) Two control outputs to determine the selection of floppy disk drive.
(b) An output to control the Read/Write head loading of the floppy disk drives.
(c) A control output to switch the Display Unit on and off.
(d) A control output to select graphics or alphanumeric mode.
(e) A control output to reset the CRT control circuits.

PARALLEL INTERFACE

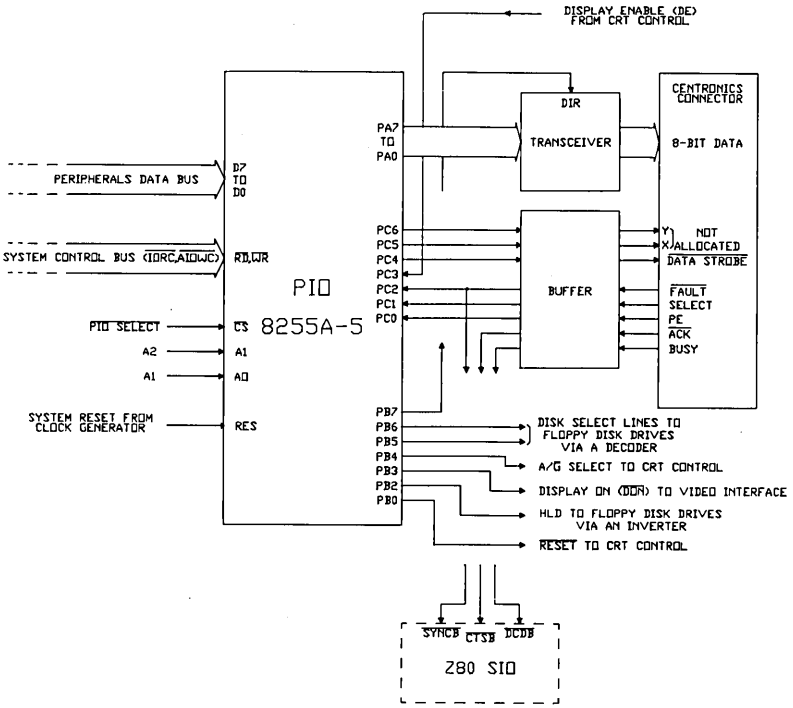


Figure 1.Parallel Interface block diagram

PIO Pin Definition

PA0 to PA7	Port A
PB0 to PB7	Port B
PC0 to PC7	Port C
D0 to D7	Data bus connection
$\overline{RD}$	Read control line
$\overline{WR}$	Write control line
$\overline{CS}$	Chip select input
A0,A1	System address bus inputs

DESCRIPTION

General

The Parallel Interface consists of; a programmable parallel input output port, an octal bus transceiver, an octal buffer and a Centronics connector. The buffer forms the interface for control signals between the board and the printer, and the transceiver forms the interface for data from the board to the printer.

The transceiver (see Figure 1) is bi-directional, with the direction of data flow controlled by the BIOS. Whilst used as a printer port, the flow of data is always in one direction only, out to the printer. However with the direction capability of the transceiver, the port can be more generally used for transferring data to and from the Apricot.

The three input control signals, Fault, Ack and Busy are connected, via the buffer, to input lines of the serial interface on the board (SIO). The three inputs are used to generate an interrupt to the CPU, utilising the interrupt capabilities of the SIO.

The Intel 8255A-5 Parallel Input Output Port (PIO) is organized internally as three input/output ports, with an associated control register. The control register determines the direction and mode of operation of each port.

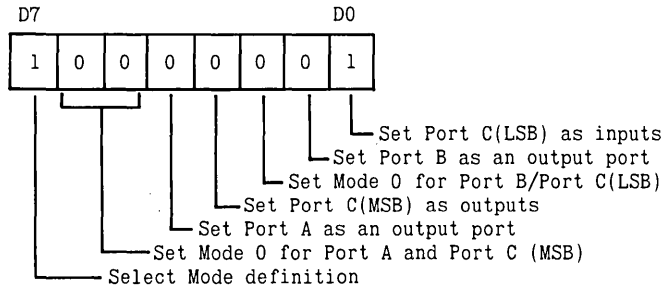
The system software views the three ports and the control register as an array of peripheral ports, with the port address locations defined by the $\overline{\text{PIO}}$ select and system address bus connections, as detailed below.

Peripheral Port	Address	Data
Port A	48H	Printer data
Port B	4AH	System control outputs
Port C	4CH	Printer control inputs/outputs plus a system control input.
Control register	4EH	Control word

PARALLEL INTERFACE

The constraints of the various hardware inputs and outputs to and from the PIO dictates the mode and direction of operation of the three ports. Port A, Port B and the 4 most significant bits (MSB) of Port C are configured as outputs and the 4 least significant bits (LSB) of Port C are configured as inputs, to provide the interface with a printer and the system connections.

The operating mode required is the basic input/output mode, defined as Mode 0. The control word to be written to the control register to set the three ports into the required operating configuration is thus as follows:



After the mode and direction of operation of the three ports is defined by the control word, the ports remain in the set operating condition unless redefined by a different control word or reset by the system reset. The system reset, when active (logic high), configures all lines of the three ports as inputs.

Reading data from and writing data to the ports of the PIO is performed by simple input/output operations, using the address locations detailed above. Data written to a port is latched into the PIO, whilst data read from the PIO is not latched.

Printer Interface

Eight bits of parallel data are supplied to the printer via Port A of the PIO and the transceiver. To set the transceiver to transmit data to the printer, the MSB of Port B (PB7) is set to logic low. The transceiver and the buffer for the printer control signals are TTL devices.

The definition of the control outputs to and from the printer and the details of the Centronics connector are provided below.

Centronics Connector Details

Pin	Description	Pin	Description
1	$\overline{\text{Data strobe}}$	19	0V
2	D0	20	0V
3	D1	21	0V
4	D2	22	0V
5	D3 (8-bit data output)	23	0V
6	D4	24	0V
7	D5	25	0V
8	D6	26	0V
9	D7	27	0V
10	$\overline{\text{Acknowledge}}$ ($\overline{\text{Ack}}$)	28	0V
11	Busy	29	0V
12	Paper Empty (PE)	30	0V
13	Select	31	$\overline{\text{N.C.}}$
14	0V	32	$\overline{\text{Fault}}$
15	Unallocated Output X	33	0V
16	0V	34	Unallocated output Y
17	Ground	35	N.C.
18	N.C.	36	N.C.

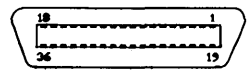


Figure 2. Centronics Connector pin detail

$\overline{\text{Data}}$ $\overline{\text{Strobe}}$	Output timing signal. Used to latch the data into the printer. Normally at logic high. Positive edge of logic low pulse indicates printer data is valid.
PE	Input signal, Paper Empty. Logic high state indicates that the printer is out of paper.

continued ....

PARALLEL INTERFACE

$\overline{\text{Fault}}$	Input signal. Logic low state indicates a printer fault condition.
Select	Input signal. Logic high state indicates the select status of printer; logic low the deselect status.
$\overline{\text{ACK}}$	Input signal, Acknowledge. Logic low state indicates that the printer is ready to receive more data.
Busy	Input signal. Logic high state indicates that the printer is unable to receive any data.

The three printer inputs, $\overline{\text{Fault}}$, $\overline{\text{Acknowledge}}$ and Busy can generate an interrupt request to the CPU by programming the SIO and interrupt controller. The SIO is also able to produce an associated interrupt vector internally, to indicate the cause of the interrupt. The system software can then determine the cause of the interrupt and the required service routine, by using data available from the SIO and data on the printer status lines.

System Control Interface

Port B and one input to Port C of the PIO form the system control interface between the processors and other areas of hardware on the System Board. The system control lines are defined below.

PB0	$\overline{\text{Reset}}$ to CRT control. Active state, logic low. When active, inhibits the production of the display address information and video timing signals, and clears the display address counters to zero count state.
PB2	HLD to floppy disk drives. Active state, logic high. When active, engages the read/write head of the selected disk drive against the disk.

continued ....

PB3	Display on ($\overline{DON}$) to video interface. A logic low on this control line allows the generated video signal through to the Display Unit. Conversely, a logic high prevents the video signal from reaching the Display Unit.
PB4	A/$\overline{G}$ select to CRT control. A logic high (alphanumeric) on this control line sets the CRT control circuitry to operate in the character display mode. A logic low ($\overline{graphics}$) sets the CRT control circuitry to operate in the graphics mode.
PB5, PB6	Disk select lines to the floppy disk drives. The selection of disk drive is determined as follows:

PB6	PB5	
0	0	Selects drive configured as drive 0.
0	1	No selection.
1	0	Selects drive configured as drive 1.
1	1	No selection.

PB7	Direction control for the transceiver of the parallel interface. A logic low on this control line enables data to be routed from Port A to the output. A logic high reverses the direction of data flow.
PC3	Display enable (DE) from CRT control circuitry. A logic high on this control line indicates the active line period of the Display Unit. A logic low indicates the flyback periods.

SOUND GENERATI0N

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The Sound Generator is located on the System Board and consists of the elements of circuitry as illustrated on the block diagram above. The Sound Generator can be programmed to produce audio frequency tones (of up to 15 kHz), audio noise, or synthesized sounds, and is utilised by the BIOS to generate:

(a) The keyboard keyclick.
(b) A bell tone to act as a general warning signal by a number of applications.

SOUND GENERATION

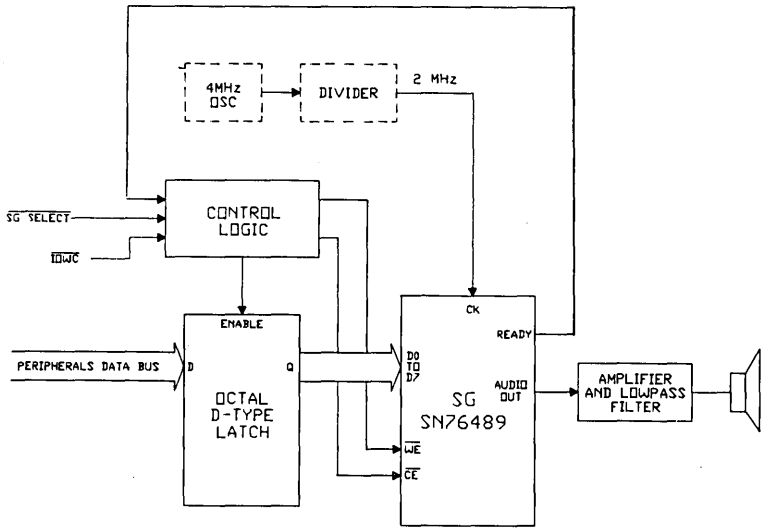


Figure 1. Sound Generator block diagram

SG Pin Definition

D0 to D7	Data bus connection
CK	2 MHz clock input
$\overline{CE}$	Chip Enable input
$\overline{WE}$	Write Enable input
READY	Ready status output
AUDIO OUT	Audio drive signal

DESCRIPTION

General

The Texas Instruments SN 76489 Programmable Sound Generator (SG) forms the basis of the sound generation circuitry. Internally the device consists of:

(a) Three programmable tone generators, each with associated control registers. (Tone generators 1, 2 , 3).
(b) A single programmable noise generator with associated control registers.

(c) An 8-bit parallel interface for transferring data from the data bus to the control registers.
(d) An audio output buffer stage.

Each generator consists of two cascaded sections; a section which produces the audio signal; and a programmable attenuator section which determines the audio signal output level and also provides the control for switching the generator off.

The audio output buffer is a summing network which automatically combines the audio signal outputs from the three tone generators and the noise generator.

The system software views the Sound Generator as a peripheral port, with an address location of 50H as defined by the $\overline{SG}$ select connection.

Due to the relatively slow data loading time of the SG device, consecutive bytes of data written to the Sound Generator should be normally separated by a delay of at least 32 clock cycles of the SG device input clock frequency (16 μs, i.e. approximately 80 processor clock cycles).

To match the slow data loading time of the Sound Generator to the speed of operation of the CPU without introducing processor wait states, the control data is not written directly to the SG device but latched through to the device inputs via an octal D-type latch. The SG then loads the control data stored at the outputs of the octal latch into the selected control register, by issuing a logic low state on the ready control output and utilising the control logic circuitry.

The logic low state on the ready control output, indicating that the SG is transferring control data into an internal register, is automatically generated on receipt of a logic low state at the CE input. The effect produced by the logic low state on the ready control output is to extend the SG input write cycle time, by holding the write enable input in the active low state for approximately 32 periods of the 2MHz input clock frequency. The ready output returns to logic high on completion of the internal data transfer process.

SOUND GENERATION

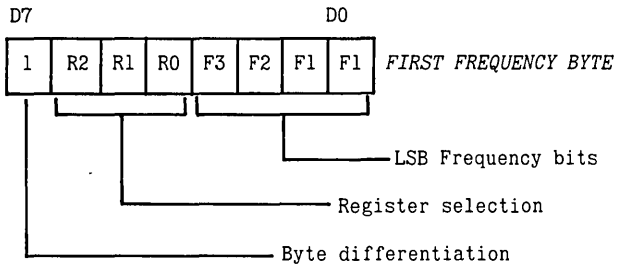
Tone Generation

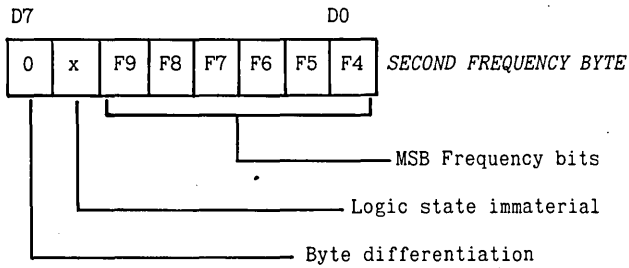
To generate a audio tone, initially requires the tone generator to be programmed with three bytes of information, **and the three other generators to be switched off.** The first two bytes determine the frequency of the audio tone and the third byte determines the level of the produced audio tone.

Once programmed with the frequency data, the tone generator can be switched on and off, using the level control byte alone.

The frequency of the audio tone is set by a 10-bit word which forms part of the first two bytes. The remaining bits of the frequency bytes select the tone generator register and differentiate between the byte containing selection bits and the byte containing frequency data only. The format of the frequency bytes and the expression used to calculate the tone frequency is as follows.

$F = 10^6 / 16n$ where F is the tone frequency and n is the 10-bit binary number in decimal format.

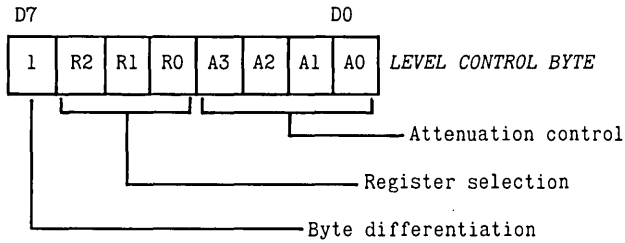




Register selection

R2	R1	R0	Register
0	0	0	Tone Generator 1 Frequency
0	0	1	Tone Generator 1 Attenuation
0	1	0	Tone Generator 2 Frequency
0	1	1	Tone Generator 2 Attenuation
1	0	0	Tone Generator 3 Frequency
1	0	1	Tone Generator 3 Attenuation
1	1	0	Noise Generator Control
1	1	1	Noise Generator Attenuation

The level of the audio tone is determined by a 4-bit word which forms part of the level control byte. Three of the remaining bits select the generator attenuator register (as detailed above), and the last remaining bit signifies that the byte contains generator register selection bits. The format of the level control byte is as follows.



The attenuation control bits allow 15 different levels of attenuation to be switched in from 0dB to 28dB in 2dB steps, and also allow the tone generator to be switched off. Each individual attenuation control bit introduces a different amount of attenuation as detailed below.

Attenuation Control

A3	A2	A1	A0	Attenuation
x	x	x	1	Introduces 2dB of attenuation
x	x	1	x	Introduces 4dB of attenuation
x	1	x	x	Introduces 8db of attenuation
1	x	x	x	Introduces 16dB of attenuation
1	1	1	1	Switches Generator OFF

SOUND GENERATION

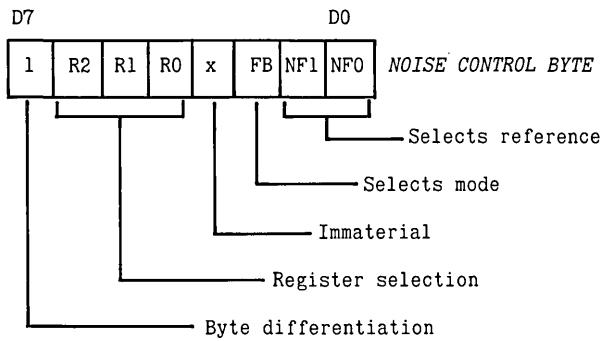
NOTE: Tone generator 1 is normally assigned to produce the bell tone. The frequency of the bell tone (800Hz) is set up during initialisation and then activated and inhibited using the level control byte.

Noise Generation

The noise generator is able to function in either one of two modes. The first mode produces "white noise" and the second mode produces "periodic noise".

The noise source for the generator is based on a feedback shift register network with the shift rate determined by a reference frequency. The reference frequency can be either one of three preset frequencies, or be determined by the audio signal frequency output from tone generator 3.

The choice of mode and reference frequency for the noise source is determined by a single noise control byte. Two bits of the noise control byte define the reference frequency and one bit defines the mode. Three of the remaining bits are used to select the noise generator control register (as defined by the register selection table in the tone generator section above), and the remaining bit signifies that the byte contains register selection bits. The format of the noise control byte is as follows.



Reference selection

NF1	NF0	Reference
0	0	3.90 kHz
0	1	1.95 kHz
1	0	975 Hz.
1	1	Tone Generator

Mode selection

FB	Mode
0	"Periodic noise"
1	"White noise"

The level of the noise signal is determined by a level control byte which allows a choice of 15 different levels of noise signal to be set, and also enables the noise generator to be switched off. The format of the level control byte is identical to the format of the level control byte described in the tone generator section above. The only difference between the level control bytes for a tone generator and the noise generator is the register selection bits.

NOTE: The noise generator is normally assigned to produce a keyclick, each time a key is pressed by updating both the noise generator register and the noise attenuator register.

EXPANSION SLOTS

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The two Expansion Slots are located on the System Board and provide an extension of the processing system for use by optional boards. The same system connections are wired to both Expansion Slots.

The extension connections wired to the Expansion Slots are:

(a) The 16-bit system data bus.
(b) The 20-bit system address bus.
(c) Various control and timing signals.
(d) Power supply outputs.

EXPANSION SLOTS

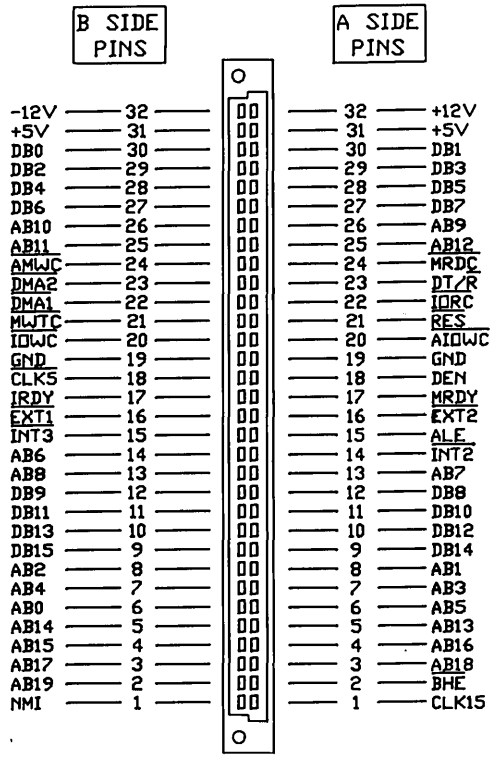


Figure 1. Expansion Connector

Pin Definition

Pin	Description	Direction
AB0 - AB19	20-bit system address bus	Output
DB0 - DB15	16-bit system data bus	Bi-directn
$\overline{\text{BHE}}$	Bus High Enable	Output
ALE	Address Latch Enable	Output
DEN	Data Enable	Output
DT/$\overline{\text{R}}$	Data Transmit/Receive	Output
$\overline{\text{AMWC}}$	Advanced Memory Write Command	Output
$\overline{\text{MWTC}}$	Memory Write Command	Output
$\overline{\text{AIOWC}}$	Advanced I/O Write Command	Output
$\overline{\text{IOWC}}$	I/O Write Command	Output

continued ....

Pin	Description	Direction
$\overline{\text{MRDC}}$	Memory Read Command	Output
$\overline{\text{IORC}}$	I/O Read Command	Output
MRDY	Memory Ready	Input
IORDY	I/O Ready	Input
$\overline{\text{RES}}$	System Reset	Output
CLK15	15MHz Clock signal	Output
$\overline{\text{CLK5}}$	5MHz Clock signal	Output
$\overline{\text{DMA1}}$	DMA Request for DMA Ch. 1	Input
$\overline{\text{EXT1}}$	External Terminate (DMA Ch. 1)	Input
$\overline{\text{DMA2}}$	DMA Request for DMA Ch. 2	Input
$\overline{\text{EXT2}}$	External Terminate (DMA Ch. 2)	Input
$\overline{\text{INT2}}$	Interrupt Request (Priority 2)	Input
$\overline{\text{INT3}}$	Interrupt Request (Priority 3)	Input
$\overline{\text{NMI}}$	Non-Maskable Interrupt	Input
+12V	System Board supply rail	Output
–12V	" "	Output
+5V	" "	Output

DESCRIPTION

Electrical specification

Current Consumption

Maximum allowed current consumption of a circuit board fitted into an expansion slot is:

0.5A from the +5V rail.
50mA from the +12V and –12V rails.

Signal Outputs

All signal outputs (data, address, control and clocks) have the capability to drive a maximum of 2 LS TTL loads,i.e.

Logic high state voltage (Voh);

2.0 < Voh < 5.25 with maximum high state output source current of 40 μA.

EXPANSION SLOTS

Logic low state voltage (Vol);

–0.5 < Vol < 0.8V with maximum low state output sink current of 0.8mA.

Signal Inputs

The signal inputs to the data bus require a tri-state driver stage meeting the following requirements.

Logic high state voltage (Voh);

2.4 < Voh < 5.25V with maximum high state output source current of 400 μA.

Logic low state voltage (Vol);

–0.5 < Vol < 0.5V with maximum low output state sink current of 8mA.

All the remaining inputs are control inputs and require to be driven by an open collector driver stage. The input control lines on the System Board are fitted with pull-up resistors (3.3k).

Pin Detail

Both Expansion Slots are 64-way connectors (DIN 41612, 2 by 32 female, with a type B housing) and are identical with regard to the connections to the system buses, as illustrated on the diagram of the Expansion Connector. A description of each connection to the System Board is detailed below.

DB0 – DB15	16-bit system data bus. Connected to the pair of transceivers which form the interface between the processors and the system data bus. DB0 is the LSB, DB15 the MSB. In effect, the bus is divided into two parts (the low order section DB0 to DB7, and the high order section DB8 to DB15), to support both 8-bit (byte) and 16-bit (word) data transfers. Byte transfers from/to even address locations are transferred on the bus lines DB0 to DB7 and byte transfers from/to odd address locations are transferred on bus lines DB8 to DB15. Word transfers from/to even addresses are transferred on bus lines DB0 to DB15 in a single operation. Word transfers from/to odd address locations are automatically transferred in two consecutive data byte transfer operations; the first operation uses bus lines DB8 to DB15 (odd address transfer), and the second operation use bus lines DB0 to DB7 (even address transfer).
AB0 – AB19	20-bit system address bus. Connected to the octal D-type latches which demultiplex the 20 address bits from the local bus of the multiprocessors (time multiplexed address/ data bus for the 16 LSB and the time multiplexed address/status bus for the 4MSB). AB0 is the LSB, AB19 the MSB. AB0 has a special function and is normally used in conjunction with the BHE signal to condition circuitry for byte or word data transfers.

continued ....

EXPANSION SLOTS

$\overline{BHE}$	Bus High Enable. Connected to a D-type latch which demultiplexes the $\overline{BHE}$ signal from the local bus of the processors. Normally used in conjunction with AB0 to enable circuitry for byte or word data transfers on the low or high order sections of the 16-bit data bus as follows;

BHE	AB0	Transfer operation
0	0	Whole Word
0	1	High order byte
1	0	Low order byte
1	1	None

ALE	Address Latch Enable. Connected to the 8288 Bus Controller. Negative edge of the active high pulse provides an indication of when the address is valid.
DEN	Data Enable. Output from the 8288 Bus Controller. Active high during memory and input/output data transfers.
DT/$\overline{R}$	Data Transmit/Receive. Output from the 8288 Bus Controller. Signifies direction of data flow. Logic high indicates data transmission from the processing system; logic low, data reception.
$\overline{AMWC}$	Advanced Memory Write Command. Output from the 8288 Bus Controller. Active low control signal which is set active before the Memory Write Command to provide memory-mapped devices an earlier indication of a write cycle.
$\overline{MWTC}$	Memory Write Command. Output from the 8288 Bus Controller. Active low write command for memory-mapped devices.

continued ....

$\overline{\text{AIOWC}}$	Advanced Input/Output Write Command. Output from the 8288 Bus Controller. Active low control signal which is set active before the Input/Output Write Command to provide input/output devices an earlier indication of a write cycle.
$\overline{\text{IOWC}}$	Input/Output Write Command. Output from the 8288 Bus Controller. Active low write command for input/output devices.
$\overline{\text{MRDC}}$	Memory Read Command. Output from the 8288 Bus Controller. Active low read command for memory-mapped devices.
$\overline{\text{IORC}}$	Input/Output Read Command. Output from the 8288 Bus Controller. Active low read command for input/output devices.
MRDY	Memory Ready. Input connected to the 8284A Clock Generator (RDY2) via an AND gate. Normally at logic high, but is set low to command the processing elements to extend the control transfer commands, by inserting wait states, until the selected memory-mapped device is ready for the data transfer operation. MRDY returning to logic high indicates that the selected memory-mapped device is ready for the data transfer operation (read or write).
IORDY	Input/Output Ready. Input connected to the 8284A Clock Generator (RDY1) via an AND gate. Normally at logic high, but is set low to command the processing elements to extend the control transfer commands, by inserting wait states, until the selected input/output device is ready for the data transfer operation. IORDY returning to logic high indicates that the selected input/output device is ready for the data transfer operation (read or write).

continued ....

EXPANSION SLOTS

$\overline{RES}$	System Reset. Output from the 8284A Clock Generator via an inverter. Active low state generated by the Clock Generator on receiving a hardware reset (power on reset or via the Hardware Reset button on the Keyboard Unit).
CLK15	15 MHz Clock signal. Buffered output from the 8284A Clock Generator with a 50% duty cycle.
$\overline{CLK5}$	5 MHz Clock signal. Output from the 8284A Clock Generator via an inverter. Inverted form of the clock signal supplied to the processing elements. 66% duty cycle.
$\overline{DMA1}$	DMA Request for DMA Channel 1. Input connected to the DMA request line (DRQ1) of the 8089 Input Output Processor (IOP) via an inverter and OR gate. A logic low on $\overline{DMA1}$ signifies to the IOP that the selected device is ready for a DMA transfer operation (read or write), using DMA Channel 1 of the IOP.
$\overline{EXT1}$	External Terminate for DMA Channel 1. Input connected to the External Terminate line (EXT1) of the 8089 IOP via an inverter and OR gate. A logic low on $\overline{EXT1}$ requests the IOP to terminate the current DMA transfer operation on DMA Channel 1.
$\overline{DMA2}$	DMA Request for DMA Channel 2. Input connected to the DMA Request line (DRQ2) of the 8089 IOP via an AND gate and inverter. A logic low on $\overline{DMA2}$ signifies to the IOP that the selected device is ready for a DMA transfer operation (read or write), using DMA Channel 2.

continued ....

$\overline{EXT2}$	External Terminate for DMA Channel 2. Input connected to the External Terminate line (EXT2) of the 8089 IOP via an inverter. A logic low on $\overline{EXT2}$ requests the IOP to terminate the current DMA transfer operation on DMA Channel 2.
$\overline{INT2}$	Interrupt Request (Priority 2). Input line connected to the Interrupt Request 2 input of the Programmable Interrupt Controller (PIC) via an inverter. The interrupt type number supplied to the 8086 processor on acknowledgement of the interrupt request is 52H. (The interrupt type number acts as a pointer to the interrupt service routine.)
$\overline{INT3}$	Interrupt Request (Priority 3). Input line connected to the Interrupt Request 3 input of the PIC via an inverter. The interrupt type number supplied to the 8086 processor on acknowledgement of the interrupt request is 53H. (The interrupt type number act as a pointer to the interrupt service routine.)
$\overline{NMI}$	Non-Maskable Interrupt. Input connected to the NMI input of the 8086 processor via an inverter. A logic high to low transition on NMI generates the predefined interrupt type number 2 internally within the 8086, which acts as a pointer to an interrupt service routine.

Address Allocation

The available address locations in the system memory space and input/output space allocated to the Expansion Slots are detailed below. 8-bit devices connected to the lower half of the data bus must be located on even address boundaries and 8-bit devices connected to the upper half, on odd address boundaries.

System memory: 40000H to EFFFFH.

System I/O: 80H to 1FFH excluding F8H to FFH.

EXPANSION SLOTS

The I/O space is divided into two ranges to cater for the varying access times of peripheral devices. In the lower address range 80H to F7H, 1 processor wait state is automatically inserted for any I/O read or write. In the upper address range 100H to 1FFH, no wait states are inserted.

Fast peripherals (access times of 350 ns or less) should therefore be located within the range 100H to 1FFH. Slower peripherals (access times from 350 to 550 ns) should be located within the range 80H to 7FH. Peripherals with even slower access times can be sited within either range, but must make use of the IRDY input to extend the processor cycle.

For memory-mapped devices, the input MRDY can be utilised to insert wait states to extend a processor memory cycle.

Expansion Board Layout Detail

The dimensions and layout details for an Expansion Board are detailed on the diagram on the next page. The uppermost view illustrates the overall board dimensions and the location of the connector. The middle projection provides a different view of the connector and details the maximum height available for components mounted on the board. The lower illustration details all the drilling requirements for the printed circuit board and the board area available.

NOTE: The 3.85 mm diameter holes located in three corners of the board are only required on expansion cards made up of two boards, in the form of a "sandwich "(i.e. a main board fitted with the expansion connector, and a piggyback board separated from the main board by spacers). In this situation, the three holes can be used as general purpose tooling holes and also as screw holes for the spacer fixings.

On single board expansion cards, the 3.85 mm diameter located in the upper right-hand corner is not required, thus providing a small extra area of board space. The two 3.85 mm holes on the left-hand side of the board together with the connector fixing hole in the lower right-hand corner can then be used as tooling holes, if required.

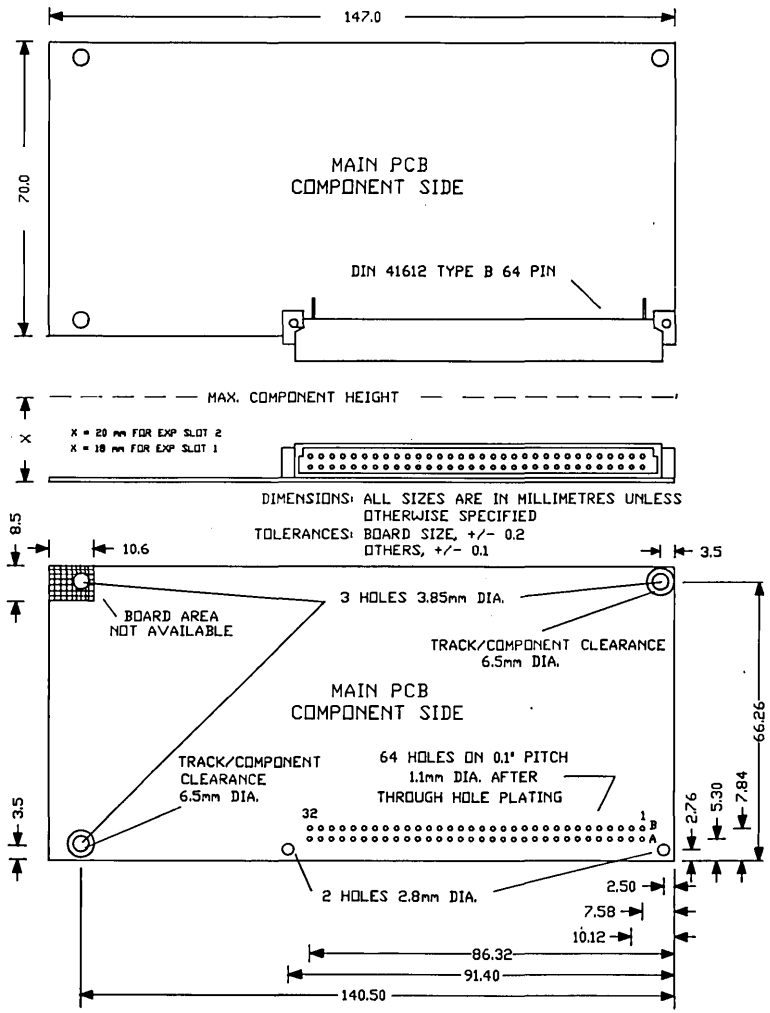


Figure 2. Expansion board detail

DISK DRIVE

List of Contents Page

List of Illustrations Figure

INTRODUCTION

This chapter provides information on the standard disk drive fitted within the Apricot, the Sony OA-D31V MicroFloppy Disk Drive, which is characterised by incorporating a single read/write head and utilising 70 track single-sided MicroFloppy disks.

The disk drive is mounted on a metal chassis assembly above the System Board in the System Unit and held in position by two pairs of screws in each of the chassis side plates.

DISK DRIVE

Connections from the disk drive controller, the Floppy Disk Interface on the System Board, are linked to the drive via a 26-way ribbon cable assembly. Power supply connections to the drive also originate from the System Board and are provided by a 4-wire cable assembly.

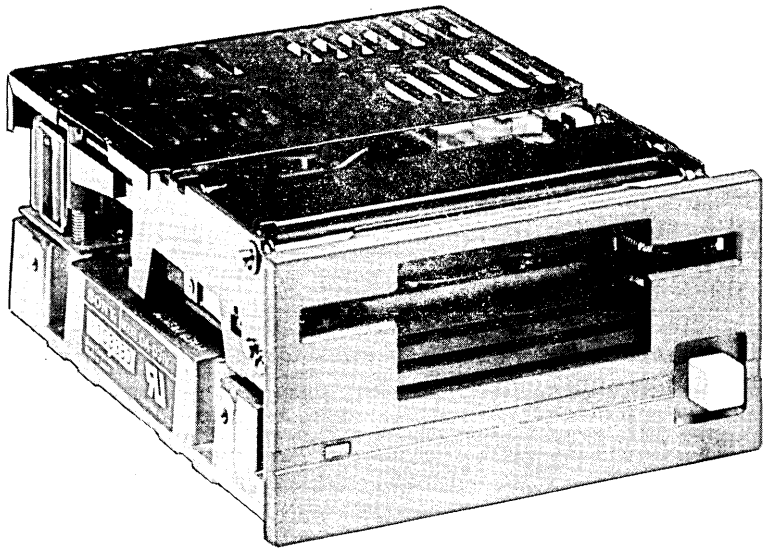


Figure 1. MicroFloppy Disk Drive

DESCRIPTION

General

The disk drive contains all the necessary electronics and mechanics for transferring MFM encoded serial data between the MicroFloppy disks and the System Board.

The electronics consist of:

(a) The interface to the disk controller (housed on a single printed circuit control board located at the base of the drive).
(b) A series of sensors for detecting various conditions within the drive (e.g. When the head is positioned over Track 0 of the disk, when a disk is in the drive, etc.).

(c) The read/write head tranducer circuitry for reading and writing data from/to the disk.

The mechanics consist of the disk drive mechanism, the disk loading/eject mechanism, and the mechanisms for positioning and engaging the read/write head.

Interface Connections

Connections between the disk drive and the System Board consist of four types; MFM encoded data signals, control input signals, status output signals and power supply lines. The latter is supplied via the four wire cable assembly; the remaining three types via the 26-way ribbon cable. All signals supplied via the 26-way ribbon cable are at TTL logic levels, apart from the Index Pulse, which is driven by an open collector.

The function of each connection is detailed in tabular format following the interface connection block diagram below.

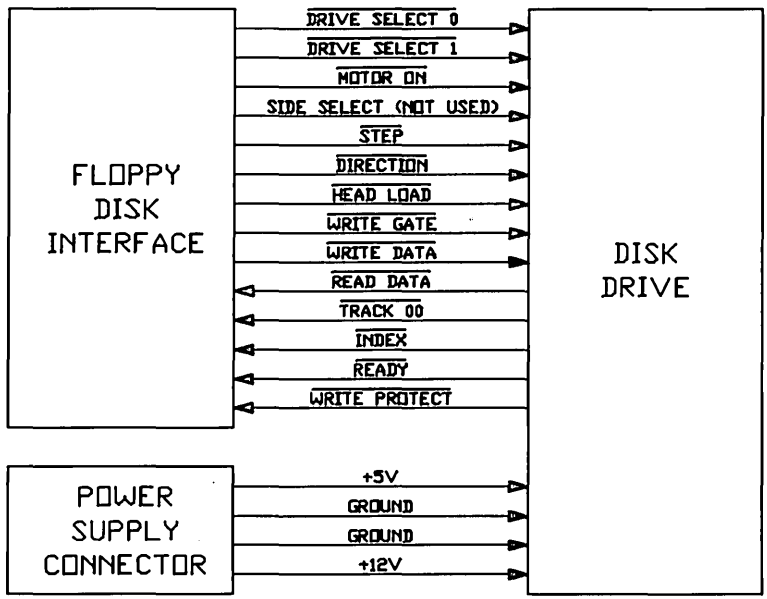


Figure 2. Interface block diagram

DISK DRIVE

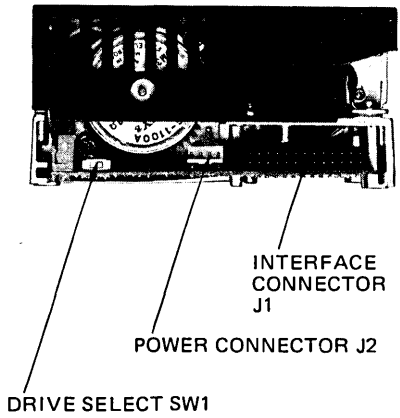


Figure 3. Connector Location

Interface Connections (Outputs)

$\overline{\text{Ready}}$	Active state, logic low. When active, indicates that a disk is within the drive, the drive is selected, and the drive motor is rotating at the normal operational speed (i.e. the drive is available for a data transfer operation). The time taken for $\overline{\text{Ready}}$ to be set active after receiving the input to select the drive with the two other conditions already met, is of the order of 0.5 μs. The time taken for $\overline{\text{Ready}}$ to be set active after inserting a disk into the drive with the drive already selected, is the order of one second.
$\overline{\text{Write Protect}}$	Active state, logic low. When active, indicates that the disk is write protected. If the drive is not selected, the output is set to logic high, regardless of the disk status.

continued ....

$\overline{\text{Index}}$	Index pulse which acts as the reference for the start of a track. Short duration negative going pulse (150 to 350 μs), generated once per revolution of the disk (i.e. every 100 ms at normal operational speed). If the drive is not selected, the output is held at logic high.
$\overline{\text{Track 00}}$	Active state, logic low. When active, indicates that the head is positioned over Track 0. If the drive is not selected, the output is set to logic high regardless of the position of the head.
$\overline{\text{Read Data}}$	MFM encoded serial data stream from the disk. If the drive is not selected, the output is forced to logic high.

Interface Connections (Inputs)

$\overline{\text{Drive Select 0}}$	Active state, logic low. When active, selects the drive for operation, if the drive is configured as drive 0 by a switch at the rear of the unit.
$\overline{\text{Drive Select 1}}$	Active state, logic low. When active, selects the drive for operation, if the drive is configured as drive 1 by a switch at the rear of the unit.
$\overline{\text{Motor On}}$	Active state, logic low. Hardwired to 0V on the System Board. The drive is configured by a switch on the underside of the unit so that the drive motor is switched on only when a disk is within the drive.
$\overline{\text{Step}}$	A negative going pulse generated by the disk drive controller which moves the read/write head, if the drive is selected. Each pulse causes the head to be moved to an adjacent track location, in the direction specified by the direction input.

continued ....

DISK DRIVE

$\overline{\text{Direction}}$	For each valid step pulse, the head moves in one track location towards track 69, if the direction control line is at logic low; and one track location towards track 0, if at logic high. If the head is already at either track 0 or track 69 and a step pulse is issued with the direction input set to move the head outside the normal track range, the head is held stationary.
$\overline{\text{Head Load}}$	Active state, logic low. When active, causes the disk to make contact with the drive head, if the drive is selected. An indication that the head is loaded is provided by the activity indicator on the front panel. If the drive is deselected, whilst the head load signal is still active, the head remains loaded.
$\overline{\text{Write Gate}}$	Active state, logic low. When active; enables the drive write control circuits to receive the write data from the disk controller; switches current through to the read/write head; and also enables the tunnel erase head. Set to the inactive high state during disk read and all head positioning operations.
$\overline{\text{Write Data}}$	MFM encoded serial data. Changes the polarity of the current flowing through the read/write head on each negative-going transition, providing the following conditions are met: 1. The drive is selected. 2. The $\overline{\text{Write Gate}}$ input is active. 3. A write unprotected disk is inserted. 4. The drive motor is rotating at operational speed. 5. The head has been loaded and is stationary.

Disk Drive Mechanism

The disk drive mechanism is a brushless direct drive motor, which employs a velocity servo control circuit to ensure that the disk rotates at a constant speed of 600 rpm. The drive is configured so that the motor rotates only when a disk is within the unit. Removal of the disk from the drive causes the motor to stop. The time taken for the motor to reach the normal operating speed, following the insertion of a disk, is the order of one second.

The servo control circuit also generates the index pulse once per revolution of the disk.

Read/Write Head

The head consists of a read/write element and a pair of tunnel erase heads which provide guard bands for adjacent tracks. Current is supplied to the read/write element on receipt of an active Write Gate signal from the disk controller, which also activates the tunnel erase section.

Head Positioning Mechanism

The head positioning mechanism uses a stepping motor and a guide arm controlled by a needle screw to precisely position the read/write head over the tracks on the disk. Control of the movement of the head is supplied by the Step and Direction inputs from the disk controller. On application of the power supplies, the drive automatically generates control signals to position the head over Track 0.

Head Load Mechanism

Head loading is controlled by the Head Load signal from the System Board. When the signal is active, a plunger causes a pressure pad to press the disk against the head. The activity indicator on the front panel of the unit remains illuminated, as long as the head remains loaded.

DISK DRIVE

Sensors and Detectors

A series of photo-sensors and associated detector circuits are fitted in the drive. These generate status output signals to the disk controller, on detecting the conditions detailed below.

(a) A disk is within the drive ($\overline{\text{Ready}}$).
(b) The disk is write protected ($\overline{\text{Write Protect}}$).
(c) The head is positioned over Track 0 ($\overline{\text{Track 00}}$).
(d) The start of each track ($\overline{\text{Index Pulse}}$).

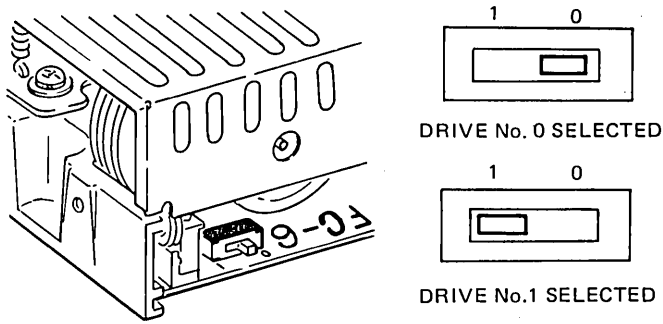
Drive Switch Settings

Two switches are located on the drive, which are set according to the application of the drive within the system. One switch (SW1) determines which of the two drive select input signals switch the drive to an operational condition. The second switch (SW2) determines the method of switching on the disk drive motor.

The drive select switch is located at the rear of the unit (see Figure 4). In single disk drive systems, the switch is set to drive position 0. In dual drive systems, the left hand drive (as viewed from the front of the System Unit) is configured as Drive 0, and the right hand drive is configured as Drive 1.

The motor control switch is located on the circuit board in the base of the unit, just behind the front panel (see Figure 4). The switch must be set to position B, so the disk motor rotates only when a disk is within the drive. Setting the switch to position A, will cause the disk motor to rotate, regardless of whether a disk is within the unit or not.

Drive Select Control



Drive Motor Control

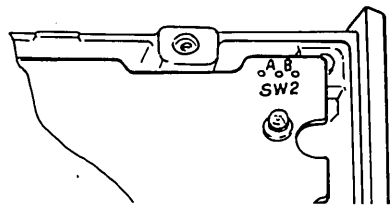


Figure 4. Disk Drive Switch Settings

Drive Specification

Media	3.5 inch 70 track single-sided MicroFloppy Disks. Soft-sectored with 9 sectors per track, 512 bytes per sector, producing a total formatted data capacity of 315 Kbytes per disk. Encoded with double density MFM data.
Data Transfer Rate	500 kbits/s.
Media Life	More than 10^6passes/track.
Track density	135 tracks per inch.
Track-to-track access time	15 ms.

DISK DRIVE

Head Load Time 60 ms.

Head Settling Time 15 ms.

Rotational Speed 600 rpm (power-up time approx. 1 second after insertion of the disk).

Dimensions

Height 2.0 inches (51mm).
Width 4.0 inches (102mm).
Depth 5.1 inches (130mm).
Weight 1.5 lbs. (700g).

Current Consumption +12V supply: Typically 0.4A (max. 1.5A)
+5V supply: Typically 0.6A (max. 0.8A)

Environmental Conditions

1. Operating

Temperature 40F to 115F (5C to 45C).

Humidity 20% to 80% relative humidity, with a wet bulb temperature of 85F (29C) and no condensation.

2. Storage

Temperature –40F to 140F (–40C to 60C)

Humidity 5% to 95% relative humidity,(no condensation).

DISKS

General

Note: To ensure complete compatibility with the Apricot, only ACT approved MicroFloppy disks should be used with

the disk drive. Using non-ACT disks, not manufactured to the same high standard, may result in intermittent read and write errors, rendering information stored on the disk totally unintelligible.

The Sony OA-D31V disk drive uses 3.5 inch 70 track single-sided MicroFloppy disks. The disks are encased in a rigid plastic shell and feature an automatic shutter and a metal centering hub.

Two versions of MicroFloppy disks exist, which are directly compatible and interchangeable, the only differences being in minor mechanical details. The auto shutter of the earlier versions of the disks can be latched open, exposing a portion of the disk surface. The latch is released by squeezing the corner of the disk, marked by the arrow labelled "PINCH". On later versions of the disk, the shutter cannot be latched open.

The shutter protects the disk media from contamination by dust, dirt or fingerprints, allowing the disk to be easily handled without affecting the integrity of stored data. When inserted into the drive, the shutter automatically slides open, allowing the read/write head of the drive access to the recording media. Removal from the drive automatically closes the shutter.

The metal centering hub ensures that the disk is accurately positioned on the disk drive motor spindle.

Disk Precautions

The same precautions apply to the MicroFloppy disks as any other magnetic recording media. **Do not:**

(a) Touch the disk surface.
(b) Allow the disk to be placed in the proximity of other magnetic materials or other sources of magnetic fields.
(c) Expose the disks to heat or direct sunlight.
(d) Attempt to clean the disk surface.

DISK DRIVE

Disk Insertion/Removal

Insert the disk into the drive shutter side first, with the metal centering hub facing downwards. The disk should slide into the drive with the minimum degree of force. Immediately the disk is inserted, the head is momentarily loaded to seat the disk properly onto the drive spindle. This causes the front panel activity indicator to be briefly illuminated. Improper insertion results in the disk not being accepted by the drive.

Remove the disk by pressing the front panel disk eject button. The disk eject button should not be pressed, whilst the activity indicator is illuminated.

Write Protecting

The method of write protecting the disk is different on the two versions of disk. The earlier versions feature a Write Protect tab, located on the rear of the disk in the lower right corner. Breaking off the tab prevents any further data being written onto the disk. The tab then should be inserted into the bottom of the recess at right angles to its original orientation (i.e. as close as possible to the lower edge of the disk). Write protected disks can be rewritten on, by moving the tab to the top of the recess.

On the later versions of the disk, write protecting the disk is achieved by sliding the tab to the lower position, creating a window in the lower left corner of the disk. To allow the disk to be rewritten onto, slide the tab back over the window.

Disk Format

Each of the 70 tracks on the disk is divided into sectors under software control (i.e. soft sectored), with the beginning of each track indicated by the index pulse generated by the drive motor assembly. The software format chosen for the disks is a derivation of the IBM system 34 format for 8 inch disks. This uses double density MFM encoded data, with 9 sectors per track and 512 bytes per sector. The total storage capacity of the 70 track disk is thus 315 Kbytes of formatted data (9x70x512 bytes).

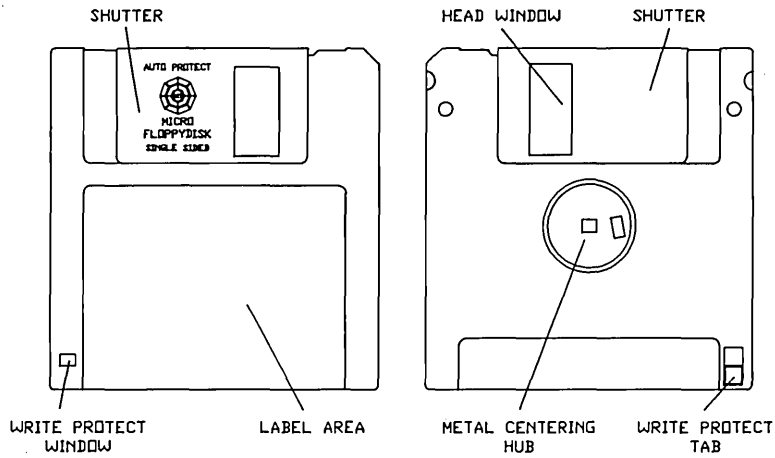


Figure 5. Microfloppy Disks

KEYBOARD

List of Contents Page

List of Illustrations Figure

INTRODUCTION

The Apricot Keyboard is rather more sophisticated than the usual computer Keyboard. As well as the keyswitch array there is the MicroScreen, which is a liquid crystal display with two lines of forty characters. A battery backed-up time and date chip, a membrane keypad with integral LED's and a port to connect a mouse are all incorporated in the Keyboard. Later sections of this chapter describe each of the above areas in detail and also outline the operation of the Keyboard firmware.

The central element in the Keyboard is the processor which is a single chip device, type number 6301. This device has 4K bytes of code space, 128 bytes of RAM, 29 parallel bidirectional I/O lines, a 16 bit timer, and a UART all in the one device together with a vectored interrupt controller to facilitate the use of these facilities. As a result of this high level of integration very little additional logic is required to connect the peripheral devices to the processor.

KEYBOARD

A list of character codes and their meaning, and the MicroScreen character display codes are contained in tabular format at the end of the section.

SERIAL DATA LINK CONTROL AND FORMAT

The data link passes asynchronous serial date at V28 levels between the Keyboard and the System Unit. The format is NRZ, with the "O" start bit followed by eight data bits, followed by "1" stop bit, no parity information is transmitted. The line idle condition is a continuous string of "1s". The data transmission rate is approximately 7800 bit/s

There is no control of data flow from the Keyboard to the System Unit so to avoid the loss of data the System Unit has to be capable of receiving and processing data at the maximum data rate i.e. 780 cps.

Data flow from the System Unit is controlled in one of two ways. The first method is when the Keyboard buffer is full, it will then transmit an X-off character to the System Unit. When the System Unit receives the X-off character from the Keyboard rather than the mouse port it must not physically send more than four further characters in order to prevent overflow at the Keyboard. The System Unit will receive the X-on character from the Keyboard when the Keyboard buffer is empty, it can resume transmission. With the exception of data from the mouse port, which would be preceded by a prefix code, the X-on and X-off characters will not appear in any other context.

The second method is when data is being received from the mouse port. In this case the data constitutes an implicit X-off condition and no more than four characters can be physically sent to the Keyboard. At the end of data reception from the mouse port transmission to the Keyboard may resume if an X-off character had not previously been received from the Keyboard.

The program is constructed so that data for the time and date will not have key data interleaved but it is possible that data from the mouse port may be transmitted. In order to

avoid confusion all time and date information will be in the range F0H to F9H where the lower nibble contains the data.

KEYSWITCH OPERATION

The keyswitch array consists of 90 capacitive keyswitches and 6 membrane keys. The capacitive keys are organised in a matrix with twelve rows of eight keys, with some omissions, which are accessed through the two Keytronic proprietary chips connected to ports 1 and 3 of the 6301. The membrane key positions are read through bits 0 to 5 of port 4 on the 6301.

The key array is scanned by latching the appropriate row address (in the range 0 to 11) into the driver IC and then reading the output of the detector IC to determine which keys are depressed. The detector has eight inputs, thus knowing which of the rows has been strobed the identity of the depressed keys can be determined. The detector has two sensitivities, which are selected by latching row address 14 or 15 into the driver, thus providing hysteresis eliminating teasing of the keys. If a key state has changed then the row is scanned again 250 us later at the other sensitivity to give hysteresis and immunity to external interference. If the key is detected as being in the same state by both scans and this state is different from the previous state then the key position has changed.

When a key is detected as having been depressed the appropriate character is transmitted to the System Unit (one of the make codes) and the same character with 80H added is transmitted when the key is released. The actual key scan codes used by the Keyboard handler of the BIOS are illustrated on Figure 2. The corresponding make codes in decimal are also detailed in a table ("Keyboard to System Unit Codes") at the end of the section.

The capacitive keyswitch array and the membrane keyboard are scanned for changes in key position every 10 ms. Keys that are newly down have their make code placed in the UART transmit buffer and also in the delay stack, which is eight bytes long. If a key is found to be newly up then its code is compared with those on the delay stack. If it

is present in the delay stack then it is ignored, if it is not present then the break code of the key is inserted in the transmit buffer. When all the keys have been scanned the oldest character on the delay stack is removed. The scan is performed twice for each row of keys that has changed

The purpose of this arrangement is to eliminate key bounce. The consequence is that if a single key is operated in each 10 ms interval the time between its downcode and its upcode being transmitted will be between 70 and 80 ms creating a maximum manual repetition of 12.5 cps. However each additional key pressed within the 10 ms interval will reduce the delay time by 10 ms. If more than eight keys are depressed within the 10 ms interval only the first eight are detected, the remainder are ignored until the next scan. This is caused by the limitation on data transmission due to the bit rate of about 10 characters per 10 ms.

The membrane keys are debounced for 30 ms on both make and break to remove bounce. The key codes are placed on the delay stock in the same way as the ordinary key codes.

MICROSCREEN OPERATION

The MicroScreen is a liquid crystal display (LCD) with two lines of forty characters. These characters are formed on a 7x5 matrix. Integral with the function keys are six LED's which together with the CAPS LOCK and STOP LED's are controlled by the shift register connected to bits 6 and 7 of port 4 on the 6301.

The MicroScreen data lines are connected to port 3 of the 6301 which is switched to output mode when data is transferred to it. Data transfer to it is enabled by bit 7 of port 1 and the output strobe line from the 6301 latches the data in. The LCD is connected as a write-only device and synchronisation of the data output with the LCD internal operation is achieved by software timing. The LCD has two registers, which are selected by bit 0 of port 1. The register selected when this bit is 0 provides the control functions such as clear screen, cursor movement etc. The other

register is used either for selecting characters to be displayed or writing the data for the character shapes that are held in RAM rather than in the character generator ROM.

The following functions are available for the MicroScreen:

1. Text display in two lines of forty characters.
2. Absolute cursor addressing.
3. Move cursor left (non-destructive).
4. Move cursor right (non-destructive).
5. Clear display.
6. Display on/off.
7. Cursor on/off.
8. Display time, day and date.

The character set is shown in a table at the end of the chapter, the codes shown are those that are actually sent to the Keyboard and do not correspond to those that are sent to the MicroScreen device in MS-DOS.

The cursor is a single non blinking underline that is displayed (when enabled) where the next character sent will be displayed.

The physical cursor address for line one of the display are 0 to 27H and for line 2 40H to 67H. These are mapped into the range 80H to CFH (i.e. line 1 80H - A7H, line 2 A8H - CFH).

The date and time are read out of the clock IC and displayed in the top right side of the MicroScreen when the appropriate command is received by the Keyboard.The time is updated every second while it is displayed without the cursor position being disturbed. The format on the screen for the date and time is:

DAY DD MMM YY HH:MM:SS

The LED's are controlled by sending two characters to the Keyboard, the LED prefix followed by a data character in the range 00 to FF. A bit set indicates that the LED will be switched on and a bit clear that it will be extinguished.

KEYBOARD

CLOCK OPERATION

The clock IC, an OKI MSM 5832, has 13 internal registers containing the time (including seconds), the day of the week,the date, the month and the year. The data is stored in 4 bit BCD nibbles. The IC can accommodate leap years if appropriately programmed and can work in 12 hour or 24 hour mode although only 24 hour mode is used by the Apricot. The clock frequency can be adjusted by means of a variable capacitor on the Keyboard PCB.

The clock IC continues to keep time when the system is not powered. This is because the IC then draws its power from the PP3 battery contained in the compartment on the underside of the Keyboard case (see Figure 3).

This software provides three basic functions, passing the time and date to the System Unit, displaying the time and date on the LCD and programming the clock with data supplied by the System Unit.

The System Unit requests the time by sending the appropriate command to the Keyboard. The Keyboard responds by sending the time prefix character followed by thirteen data bytes, the lower nibble contains the BCD data and the high nibble is all ones. The order of the data is: year, month, date, day of the week, hours, minutes, seconds, with tens first and units seconds.

The date and time are displayed when the appropriate code is sent to the Keyboard. The format on the screen is shown in the section on the MicroScreen. The display of the time and date stops when the clear screen command is received.

The time or date are programmed by sending the appropriate prefix followed by BCD data in the same format and order as sent from the Keyboard. Other commands can be interleaved with the clock data. However if date and time program or LED commands are nested the resultant contents of the clock chip are unpredictable. The clock is programmed for leap years by setting bit 2 of the most significant date digit.

MOUSE PORT OPERATION

The interface to the mouse is basically a hardware multiplexing arrangement, the data line for transmitting from the Keyboard to the System Unit is also used by the Mouse. This data line is also shared with the reset switch which overrides any data transmission.

The mouse port provides 12 volts continuously to the mouse and has two control lines, used to implement the hand-shaking protocol and the data line used to transmit data to the Keyboard from the mouse.

The two control lines are Bus Request and Bus Grant. Bus Request is asserted by the mouse when it has data to transmit, then if the Apricot CPU has sent a mouse enable command to the Keyboard and the Keyboard has finished transmitting the current character, the Keyboard will assert the Bus Grant line thus permitting the mouse to transmit data. This data is sent to the CPU along the same wire as transmitted data from the Keyboard. The mouse must not transmit data while Bus Grant is not asserted as this will corrupt any data currently being sent by the Keyboard.

The Keyboard also has a Mouse Disable command. This immediately sets the Bus Grant line inactive and prevents further bus requests being granted. The Keyboard does not restart transmission until the mouse sets the Bus Request line inactive. After power-up the mouse is initially disabled.

The Keyboard Mouse Enable command allows the assertion of the Bus Request line to be recognised by the Keyboard and the bus granted as in the above description.

The data from the mouse port is preceded by a prefix code (sent by the mouse) this is an implicit X-off character and the System Unit must not send more than four further characters to the Keyboard. At the end of the data transmission from the mouse port, may resume if an X-off had not previously been received from the Keyboard.

KEYBOARD

KEYBOARD FIRMWARE

The Keyboard firmware is a real-time interrupt driven operating system. The main scheduler is split into two parts, one which is continuously executed and the other which is only executed every 10ms. Each part carries out several operations, the first part scans the receive character buffer for characters received and processes any that have not previously been dealt with, it also sends X-on when appropriate and polls the mouse port for the mouse releasing the data link.

The second part is executed after the timer interrupt routine has been executed (every 10 ms). If the time and date are being displayed then it is updated if necessary. After this the Keyboard is scanned for any changes to the key positions. If the Keyboard has not received a reset acknowledge character since power up or switching into diagnostic mode a reset request character is transmitted if 100 ms has elapsed since the last one was sent. In this phase all other characters are ignored.

This main routine is interrupted by several events. The first is reception of characters from the System Unit, the second is the UART transmitter being empty, when there are characters to send to the System Unit. Other sources are the 10 ms timer maturing, or the mouse requesting the bus, if it is enabled.

The processing of characters received from the System Unit is split into two parts. The first is all the commands that do not directly affect the LCD. These are of the form ExH and are processed by means of a jump table, their effect is shown in the System Unit to Keyboard table and the relevant section. When the appropriate code is received the processor diagnostics are executed. There are two tests, a ROM checksum and a RAM self address test.

If both of the tests are passed then FBH is returned. If the ROM test failed then E9H is returned. If the RAM test passed but the ROM test failed the sequence E9H, FBH is returned. If the ROM test passes but the RAM test fails then 80H is sent, if both fail then E9H, 80H is sent.

In all cases any data in the RAM is destroyed i.e. all record of make codes that have been transmitted is lost and the Keyboard is left in a state where the clock chip is in a reference signal output mode, i.e. data line 0 oscillates at 1024 Hz with a 50% duty cycle. While in this state the Keyboard sends a reset request every 100 ms.

The Keyboard is extracted from this state by sending a reset command after which it is in its normal mode of operation. If a reset character is received at any other time all variables will be initialised so that any make codes that have been transmitted will not have corresponding break codes transmitted and the initialisation routine will be called. An acknowledge character (FB) is transmitted when the Keyboard is ready for further operation.

The reset character has immediate effect, all characters that were in the buffer are lost. Synchronisation is achieved by using the query command and waiting for the acknowledge indicating that the buffer is now empty (provided no more characters have been sent to the Keyboard in the intervening time) so that a reset can be sent.

If the command is in the range D0H to D7H then the LCD subroutine is called. This controls all direct access to the LCD, for example the subroutine that displays the time and date on the LCD reads the contents of the clock IC and then calls this routine. This in turn calls a subroutine that actually outputs the data to the LCD and inserts the appropriate delay to synchronize with the LCD operation. A record is kept of the current cursor position, whether the display is on or off and whether the cursor is on or off in this routine

KEYBOARD

Keyboard to System Unit Character Codes

00H	unused
01H-60H	Make codes, these correspond to the switch codes on the PCB for switches 1 to 8, switches 9 to 0EH are the membrane keys. Switches 0FH to 60H correspond to the switch codes plus 6.
61H-68H	Reserved
69H-6FH	Not used by Keyboard
70H-7FH	Mouse Codes
80H	RAM test failed
81H-E0H	Break codes, corresponding to the make codes + 80H
E1H-E8H	Reserved
E9H	ROM test failed
EAH	X-on
EBH	X-off
ECH	Reset request
EDH	Time prefix
EEH	Date prefix
EFH	Mouse Header
F0H-F9H	BCD data
FAH	Spare returned for invalid clock data
FBH	Acknowledge firmware version/or processor diagnostics.
FCH-FEH	Reserved for future firmware version
FFH	Unused

System Unit to Keyboard Character Codes

00H	Unused
01H-FFH	Character codes for MicroScreen
80H-CFH	Cursor address
D0H	Clear screen stop time/date display
D1H	Cursor left
D2H	Cursor right
D3H	Cursor on
D4H	Cursor off
D5H	Display on
D6H	Display off - display date is not lost
D7H-DFH	Spare
E0H	Query
ElH	Time and date request EDH+13 bytes of BCD data returned
E8H	Keyboard reset or acknowledge reset request after power up.
E2H	Display time and date on MicroScreen
E3H	Set LED prefix
E4H	Set time and date prefix followed by 13 bytes of the form FxH.
E5H	Mouse enable.
E6H	Mouse disable

KEYBOARD

E7H	Execute processor diagnostics.
E9H - EFH	Spare
F0H - F9H	BCD data
FAH	Returned if any on the clock data bytes are invalid
F8H - FEH	Spare
FFH	Unused

	0	1	2	3	4	5	6	7
0	not used	÷	sp	0	@	P	`	p
1	↑	¢	!	1	A	Q	a	q
2	↓	■	"	2	B	R	b	r
3	±	°	#	3	C	S	c	s
4	£	□	$	4	D	T	d	t
5	\	sp	%	5	E	U	e	u
6	▌	sp	&	6	F	V	f	v
7	g	sp	'	7	G	W	g	w
8	~	sp	(	8	H	X	h	x
9	↑	sp	)	9	I	Y	i	y
A	↓	sp	*	:	J	Z	j	z
B	±	sp	+	;	K	[	k	{
C	£	sp	,	<	L	¥	l	|
D	\	sp	-	=	M	]	m	}
E	▌	sp	.	>	N	^	n	→
F	g	sp	/	?	□	_	o	←

sp = space

Figure 1. MicroScreen Display Codes

KEYBOARD

Keyboard Code Table

Keyboard Code	Key Legend	Keyboard Code	Key Legend
1	HELP	49	7(num. pad)
2	UNDO	50	8(num. pad)
3	REPEAT	51	9(num. pad)
4	CALC	52	CAPS LOCK
5	PRINT	53	A
6	INTR	54	S
7	MENU	55	D
8	FINISH	56	F
9	Function 1	57	G
10	Function 2	58	H
11	Function 3	59	J
12	Function 4	60	K
13	Function 5	61	L
14	Function 6	62	;
15	Backslash	63	'
16	1	64	Return
17	2	65	INSERT
18	3	66	DELETE
19	4	67	4(num. pad)
20	5	68	5(num. pad)
21	6	69	6(num. pad)
22	7	70	SHIFT(L)
23	8	71	Z
24	9	72	X
25	0	73	C
26	–	74	V
27	=	75	B
28	Backspace	76	N
29	%	77	M
30	Multiply	78	,
31	Divide	79	.
32	–(num. pad)	80	/
33	+(num. pad)	81	SHIFT(R)
34	Tab	82	Cursor up
35	Q	83	SCROLL
36	W	84	1(On num. pad)

continued ....

Keyboard Code	Key Legend	Keyboard Code	Key Legend
37	E	85	2(On num. pad)
38	R	86	3(On num. pad)
39	T	87	ESC
40	Y	88	CONTROL
41	U	89	Space bar
42	I	90	STOP
43	O	91	Cursor left
44	P	92	Cursor down
45	[	93	Cursor right
46	]	94	0(num. pad)
47	HOME	95	.(num. pad)
48	CLEAR	96	ENTER

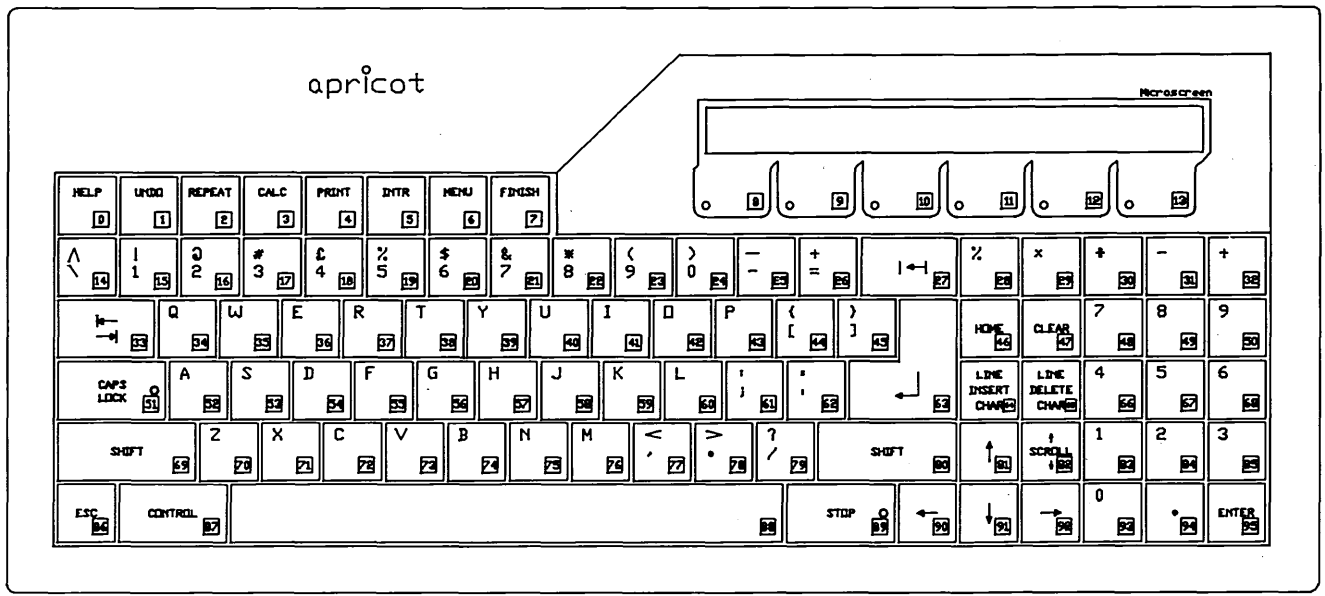
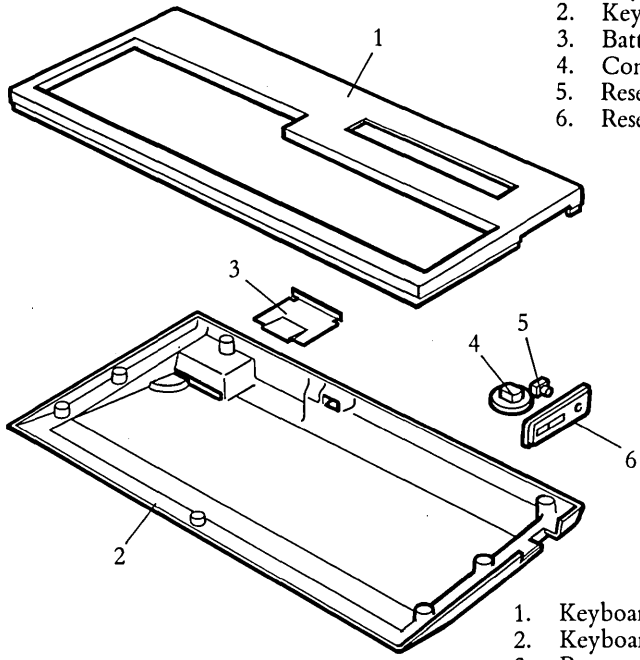


Figure 2. Keyboard Codes Used by the BIOS

1. Keyboard Bezel
2. Keyboard Base
3. Battery Cover
4. Contrast Wheel
5. Reset Button
6. Reset Plate



1. Keyboard Assembly
2. Keyboard Cable
3. Battery Connector
4. Battery
5. Reset Switch
6. Brightness Control
7. LCD Microscreen

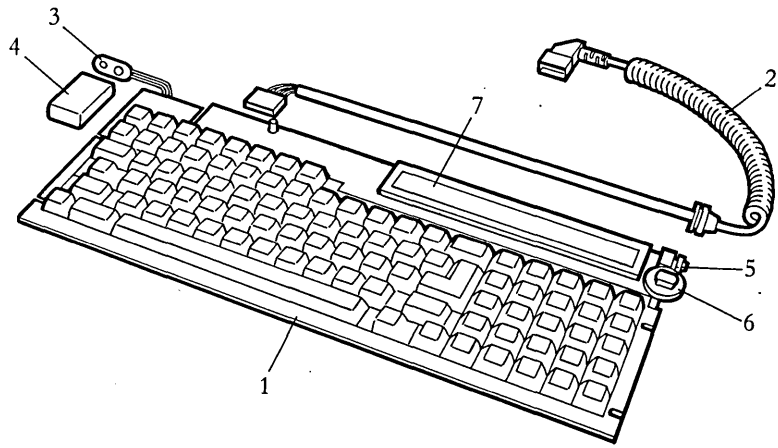


Figure 3. Keyboard Constructional Details

DISPLAY UNIT

INTRODUCTION

The standard Display Unit is a 9 inch high resolution (400lines) monochrome (green on black) display monitor, which fits onto the shallow recess on the top of the System Unit. The unit tilts and swivels, enabling the operator to position the screen to obtain the optimum viewing angle.

To minimise the possibility of operator eye fatigue:

(a) The screen incorporates a long persistence phosphor (P39) to reduce display flicker.
(b) An anti-reflective mesh is fitted over the screen to reduce glare.

Display brightness is adjusted by a manual control located next to the recessed carrying handle on the rear of the unit.

DESCRIPTION

General

The display is controlled by the CRT circuitry on the System Board, which generates a serial video signal and two sync signals (HSync and VSync to control the movement of the electron beam accross the screen). The amplitude of the video signal is approximately 0.3 V when operating in normal intensity, 0.4 V in high intensity. The sync signals are positive TTL.

The +12V supply voltage for powering the monitor is also supplied from the System Unit. A 1.0 amp fuse is located internally within the Display Unit.

The CRT control circuit is programmed to produce an interlaced picture on the screen operating at a field rate of approximately 72 Hz and a line scan rate of 15.79 kHz.

The Display Unit is connected to the System Unit via a 6-wire cable assembly terminating in a 9-pin female D-type connector.

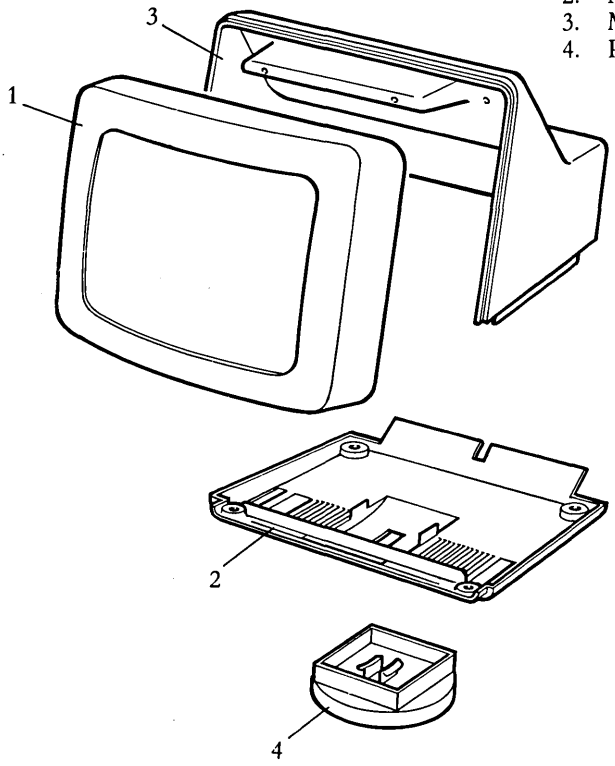
DISPLAY UNIT

Physical Dimensions

Max. height:	8.5 inches
Max. width:	10.5 inches
Max. depth:	10.0 inches
Weight:	9.1 lbs

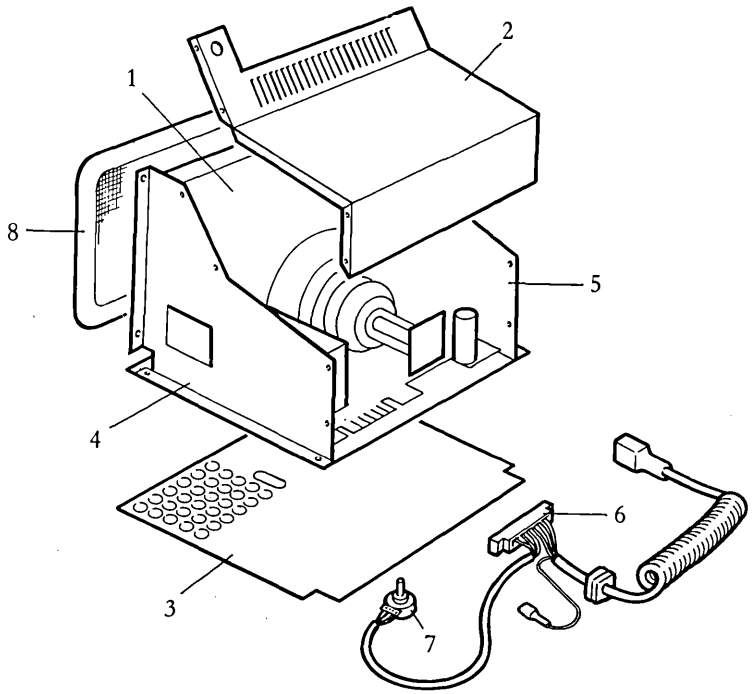
DISPLAY UNIT External

1. Monitor Bezel
2. Monitor Base
3. Monitor Cover
4. Pedestal



DISPLAY UNIT Internal

1. Wire Frame Monitor
2. Top Screen
3. Bottom Screen
4. LH Monitor Bracket
5. RH Monitor Bracket
6. Video Cable
7. Brightness Control
8. Sunflex Screen



SECTION 3:
SOFTWARE DETAIL

Software Detail

About This Section

This section of the Technical Reference manual deals with the Software for the Apricot computer.Its contents are primarily intended for software writers who wish to use the Apricot's advanced features from within their own programs. The functional operation of the machine can also be appreciated to a greater extent once a basic understanding of the software and hardware integration is achieved.

The first chapter is a basic overview to the Software system in the Apricot. There then follow a number of chapters which detail each of the "device drivers", which are pieces of software designed to handle various items of hardware, such as the Keyboard or MicroScreen.

Each device driver chapters is sub-divided into three individual portions, titled "INTRODUCTION", "APPLICATIONS INTEREST" and "SYSTEMS INTEREST".

"INTRODUCTION is intended to be read by anyone with a basic understanding of computer concepts, and gives a functional specification of what the related software actually does.

"APPLICATIONS INTEREST" essentially details how to use a device through the existing code, it is this section which will be of most interest to software developers.

"SYSTEMS INTEREST" describes what the software is and how it actually works, it also gives an insight into the way that the hardware and software combine together to make the Apricot.

PROGRAMMERS GUIDE TO THE APRICOT BIOS

INTRODUCTION

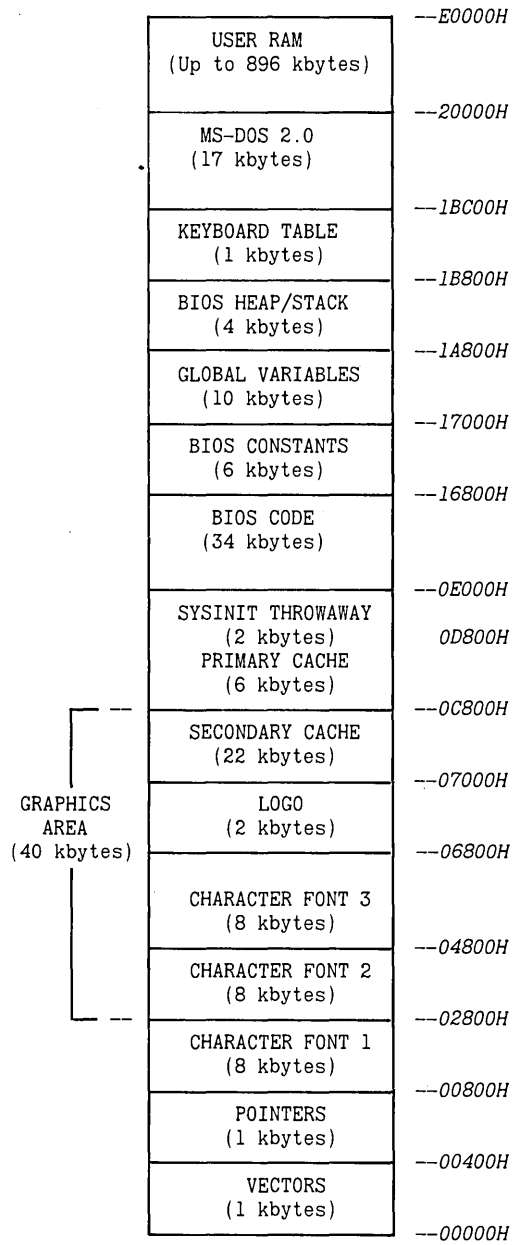
The Apricot BIOS has been written by ACT to make maximum use of the hardware facilities provided by the Apricot. It incorporates a number of unique facilities which are not provided by other microcomputers. The following sections describe how various aspects of the BIOS can be changed or accessed from within applications software. Where necessary programming examples are given. These are in MicroSoft Basic, which is included with all Apricots. For other languages, the routines will have to be changed accordingly.

Bios Internal Structure – Memory Map

All the BIOS, disk operating system and allocation of memory for graphics, disk cache, character fonts and keyboard tables exist in the first 128K bytes of RAM. Table 1 is a detailed Memory Map showing locations of the relevant parts of the lower 128k.

GUIDE TO THE BIOS

Table 1 -- Apricot Memory Map



Below is a description of all the sections illustrated in the memory map:

VECTORS

Main hardware and software Interrupt and Jump vectors. Used for service call requests from both hardware and software.

POINTERS

1K of double and single word pointers, jump tables and constants, most of which are used by the BIOS, but can also be used or changed by applications software. Table 2 shows all relevant addresses, with their corresponding functions:

Table 2 -- Pointer/constant area

HEX	ADDRESS CONTENTS	Type*
0400-04FF	BOOTSTRAP ROM WORKING AND COMMUNICATION AREA	
0402-0403	Apricot memory size in paragraphs	S
0408-0409	Drive number that the system was booted off	S
040A-040D	Pointer to configuration table in bootstrap	D
040E-0411	Pointer to BPB image in bootstrap	D
0500-05FF	IOP (8089) WORKING AND COMMMUNICATION AREA	
0500-0505	Channel control block (CCB), channel 1	
0508-051D	Channel control block (CCB), channel 2	
0600-06FF	SPECIAL JUMP TABLE	
0600-0603	Long Control Device jump-table	T
0610-067F	Reserved	
0700-07FF	BIOS CONFIGURATION AND COMMUNICATION VARIABLES	
0700-0703	Pointer to main configuration table	D
0704-0705	Length of configuration block in bytes	C
0706-0709	Pointer to active character font	D
070A-070B	Length of font in bytes	C
070C-070F	Pointer to Apricot logo in the bootprom	D
0710-0711	Length of logo in bytes	C
0712-0715	Pointer to active keyboard tables	D
0716-0717	Length of keyboard tables in bytes	C
0718-071B	Pointer to the default (boot) keyboard table	D
071C-071D	Length of default table in bytes	C

* S=single word, D=double word, T=table, C=constant

Single word values are two bytes long, the low byte containing the value required.

Double word pointers are four bytes long, and conform to the standard Intel addressing foramts. The first two bytes are the offset from the start of a segment that the pointer is to reference, and the second two bytes are the segment address.

Tables are Jump tables, used by the BIOS to quickly pass control to various parts of itself without having to make multiple conparisons which would use memory and slow down operations considerably.

A constant is two bytes long, the low byte containing the relevant data.

Care should be taken when altering any of these values as many of them are used frequently by the BIOS (the keyboard pointers are accessed a number of times on each key depression).

Here is an example method of using the pointers from Basic:

```
10 DEF SEG=0:CS=PEEK(&H0706)+256*PEEK(&H0707)
20 DEF SEG=PEEK(&H0708)+256*PEEK(&H0709)
```

In this case, line 10 sets the variable CS to point to the beginning of the character set table in RAM, and the current active segment is set to the start of the segment containing the character set.

CHARACTER FONTS 1, 2 and 3

The BIOS uses a character font to define the shapes of the characters as they appear on the screen. It consists of 256 16-word data elements – one for each character, making a total of 8K bytes. For more details see the Character Set section. There is always one default character set in RAM, which starts at 0800H. After this there is 16K reserved for another two sets if required. When the cache or graphics is used, however, there is only room for the default character set.

SECONDARY CACHE

The secondary disk cache is an extension of the primary cache, that can be used when there is no graphics RAM required. The Boot ROM character set is also placed here at boot-up, so characters such as the Apricot logo can be displayed on the screen.

PRIMARY CACHE

This is used as the main disk cache memory. Utilising the now-vacant SYSINIT space, 6K of primary cache is available. See the section on Disk Caching for more information.

SYSINIT

This module is provided by Microsoft to initialize the MS-DOS operating system. It is only used once, each time the system is booted up. Resident in the BIOS, it is "thrown away" when it has loaded and initialized the operating system and the memory space it occupied becomes vacant and is used for various system purposes (see above).

BIOS CODE

The actual BIOS program. This section of memory contains all the BIOS that the Apricot MS-DOS 2.0 implementaion uses. There are some internal tables (such as the keyboard tables and jump tables), all of which are accesed by the pointers. For assembly language programmers, this code is the primary BIOS Code Segment.

BIOS CONSTANTS

An area of code containing the constant numeric values which are established by the BIOS at boot-up time, and remain unchanged until the system is re-booted. For example, one constant is the number of disk drives that the particular Apricot configuration has, and is accsesed by MS-DOS and the BIOS.

GUIDE TO THE BIOS

GLOBAL DATA AREA

This area stores changing data values that can be accessed and changed by any part of the BIOS or operating system. Initial default values are placed into this area at boot-up time, some of them remain unchanged as pseudo-constants, but others are constantly changing to dictate the current state of the machine software and hardware. For example, the default disk drive number is stored here which has to be accessed by the BIOS and MS-DOS, but can be changed at any time. This is also the flags area, containing things such as the "Is the calculator switched on?" flag.

BIOS HEAP AND STACK

This area should never be changed by the user, as it contains the work space that the BIOS uses in every operation. The main program stack is here, along with the heap workspace. It should be noted that the BIOS Stack Segment and Data Segment are located in this area.

MS-DOS 2.0

This area of memory is reserved for MS-DOS. It contains the code, stack and all the constant area that MS-DOS 2.0 uses. This block of memory should under no circumstances be changed by the user. No documentation is (or will be) available as to its contents or layout.

USER RAM

This is where all the user programs are loaded and executed from. The memory is upward vectored, and up to the maximum Apricot address space of 896K may be used. The program area can be divided by the user into a self-contained Data, Stack and Code segments. Essentially, this area can be used at the user's discretion, and the BIOS places no limits on (memory permitting) how much code and/or data is stored here.

SCREEN DRIVER

INTRODUCTION

The Apricot Screen driver is an advanced module that is designed to handle all communications between the CRT screen and the MS-DOS 2.0 Operating system.

The software consists of a number of modules, and includes the following specific areas:

1. The CRTC interface, which initializes the display hardware on power-up. This software is included in the Boot PROM as well as the BIOS.
2. Low-level routines to display a character in a specified position on the screen.
3. Cursor management routines, to keep track of the cursor position and cursor mode.
4. Control code management functions, to handle escape sequences and Control codes.

APPLICATIONS INTEREST

Escape sequence Management functions.

The Screen driver supports 73 different Escape sequences, not including the ANSI codes. When a character 27 decimal or 1B hex is received by the Screen Driver, control is transfered (via a Jump Table) which branches to a specific piece of code depending on what the next character to be displayed is. For example, to clear the VDU screen the sequence ESCAPE E has to be sent to the screen driver. On receiving the ESCAPE code, the BIOS will wait until the next character is sent to be displayed. When it is received (in this case an "E"), control will be transferred to the section of code which deals with clearing the screen. When the screen is cleared, control is passed back to the screen driver, which can either display the next character or return to the DOS for another function to be executed. Table 1 is an annotated list of all the Apricot Escape codes.

SCREEN DRIVER

Table 1 – List of all the Apricot Escape Codes.

*** - Not Sirius compatible**

*	23	#	Transmit page. This code sends each character on the screen to the MS-DOS input buffer queue. So after printing ESC #, the keyboard buffer will contain the entire contents of the page.
	24	$	Sends the character under the cursor into the keyboard buffer.
*	25	%	Transmits the line that the cursor is on into the keyboard buffer.
*	26	&	Prints the entire screen on a connected line printer, after sending a Form Feed character.
*	27	'	Prints the line that the cursor is on on the line printer - no Form Feed is sent first.
	28	(	Set high intensity mode. All characters sent to the VDU from now on will be displayed brighter than the others.
	29	)	Cancel the high intensity mode, so all characters now sent to the VDU will appear at equal brightness.
*	2A	*	Change to the second character font.
	2B	+	Clear all the characters that are displayed on the screen in High Intensity (see code 28).
*	2C	,	Set the size of the video screen bounds. Four arguments are passed: 1-Top line 2-Bottom line 3-Left column 4-Right column

Note that 31 must be added to these values. For example, suppose you wanted to make the screen a 20 character square in the top left-hand corner. You would do this:

```
PRINT CHR$(27)+","+CHR$(1+31)+
CHR$(20+31)+CHR$(1+31)+CHR$(20+31)
```

*	2D	-	Clear all the characters displayed in normal, low intensity (see code 29).
*	2E	.	Reset the size of the video screen, revokes the parameters passed using code 2C.
*	2F	/	Set the membrane LED's – see the section on the MicroScreen for more information.
	30	0	Set the underline attribute on all characters printed from now on, so now everything printed will be underlined.
	31	1	Reset the underline status so that characters will not be underlined when printed.
	32	2	Set the blink attribute for the block or underline cursor, so it will flash on and off approximately 3 times each second.
	33	3	Switches the cursor blink off.
	34	4	Change the value of a key on the keyboard. Three arguments are passed:

1. The mode. From 1 to 3.

   1 is normal
   2 is shifted
   3 is control

2. The key number. An ASCII value in the range 0 to 95

SCREEN DRIVER

3. The new character for the key – An ASCII character from 0 to 255. e.g.

```
PRINT CHR$(27)+"43HC"
```

will disable the control C key (H is the ASCII for 72, which is the number of the C key).

	38	8	Set the LITERAL TEST MODE, where characters with an ASCII value of less than 20 hex will be displayed instead of obeyed.
*	39	9	Set the strikeout mode on: all characters printed from now on will have a horizontal line through the middle.
*	3A	:	Set strikeout off - see code 39.
*	3B	;	Move cursor to the bottom line (line 25).
*	3C	<	Display Time And Date On Mscreen.
	3F	?	Switches on the Internal calculator.
	40	@	Enter the INSERT mode - now whenever characters are printed, all text to the right-hand on the same line is shifted one place to the right.
	41	A	Cursor UP, moves the cursor up one line.
	42	B	Cursor DOWN, moves the cursor down one line.
	43	C	Moves the cursor RIGHT one position.
	44	D	Moves the cursor back one position.
	45	E	Clear the VDU screen and home the cursor. Line 25 is left intact.

46	F	Enter graphic mode, where block graphics characters appear instead of lower case letters.
47	G	Exit graphic mode (see code 46).
48	H	Home the cursor to the top left-hand corner.
49	I	Move the cursor up one line, and if it is at the top of the screen, scroll the page down.
4A	J	Erase all characters on the screen from the cursor position to the end of the page.
4B	K	Delete to end of line function, all the characters to the right of the cursor, and on the same line are deleted.
4C	L	Insert a line at the cursor position, all the lines below the cursor are scrolled down one line.
4D	M	Delete the line that the cursor is on, moving all the lines under the cursor up one to fill the gap.
4E	N	Delete the character at the cursor.
4F	O	Exit insert character mode (see code 40).
50	P	Insert a single character at the cursor position.
* 51	Q	Scroll left n characters, e.g.

```
PRINT CHR$(27) + "Q" + CHR$(31+5)
```

moves the entire screen contents left 5 characters.

* 52	R	Scroll right n characters.
* 53	S	Scroll up n characters.
* 54	T	Scroll down n characters.

*	55	U	Enable dual ouput to the MicroScreen.
*	56	V	Disable dual MicroScreen output.
*	57	W	Output all text to the MicroScreen but disable the screen. Code 56 re-enables the screen. Note that this sequence will only work after ESC U has been invoked.
	58	X	Exchange the line that the cursor is on with the last line that the cursor visited.
	59	Y	Main cursor addressing lead-in function. Two additional bytes have to be supplied: the line and the column. The top left-hand corner of the screen is position 31,31. To move the cursor to Line 7, column 41, it would be necessary to do this: `PRINT CHR$(27) + "Y" + CHR$(31+7) + CHR$(31+41)` If either value is out of range, no movement will take place.
	5A	Z	After this code is sent, the keyboard buffer is filled with 3 characters: ESCAPE / K. This can be used as a test to see if software is running on the Apricot.
*	5B	[	ANSI lead-in character (see section on ANSI).
	62	b	Erase all characters from the cursor position to the beginning of the screen.
	63	c	Disables MicroScreen scrolling.
	64	d	Enables MicroScreen scrolling (see code 63).
	65	e	When the MicroScreen is ON, this will switch the underline cursor on the MicroScreen on.

66	f	When the MicroScreen is ON, this code will switch the underline cursor on the MicroScreen off.
67	g	Disable time & date display on MicroScreen.
68	h	Reverse Tab, moves the cursor back in a modulo of 8 positions, just as Tab moves the cursor forward.
69	i	Returns the BIOS number as three characters, e.g. "2.3".
6A	j	Save the current cursor position.
6B	k	Returns the cursor to a previously saved position (see code 6A).
6C	l	Erase the entire line that the cursor is on.
6E	n	Return the current cursor position in the keyboard buffer as ESCAPE Y line, column.
6F	o	Erase from the cursor position to the beginning of the current line.
70	p	Enter the reverse video mode, all text printed from now on will appear as black characters on a green background.
71	q	Exit the reverse video mode described above.
72	r	MicroScreen echo enable, clears the MicroScreen, switches the cursor ON and the DATE display OFF.
73	s	MicroScreen echo disable, clears the MicroScreen, switches the cursor OFF and the DATE display ON.
74	t	Engage Shift Lock.

75	u	Dis-engage Shift Lock.
76	v	Wrap characters around at the end of the line (that is move on to the next line).
77	w	Remain at the end of a line when text is printed there.
78	x	Set environment flags: see below.
79	y	Reset environment flags: see below.
7A	z	Reset all modes. Any commands set with any of the above escape sequences will be revoked: see below.

Environment Flags

For escape codes 78 and 79, a single byte argument is required. This byte is an ASCII number from 1 to 9. Each number performs a specific function:

1	Line 25	: enable or disable
* 2	Key click	: on or off
3	Hold Mode	: on or off
4	Cursor type	: block or underscore
5	Cursor status	: on or off
* 6	Mouse port	: on or off
* 7	Margin bell	: on or off
8	Auto LF	: yes or no
9	Auto CR	: yes or no

So to switch the cursor OFF, this would need to be done:

```
PRINT CHR$(27)+"x5"
```

Or to switch it on:

```
PRINT CHR$(27)+"y5"
```

When ESCAPE z is used, the following action is taken:

1. The screen is cleared (including the 25th line) and the cursor is homed.
2. Any modes (ie reverse, underline, intensity) are switched OFF.
3. The cursor is switched ON, and set to BLOCK, non-flashing.
4. The Mouse Port, Margin Bell, 25th Line, Hold Mode, Auto CR and Auto LF are all disabled.

ANSI Escape Sequences

Many third-generation machines support the ANSI (American National Standards Institute) set of escape sequences. So as to retain compatibility, many of the ANSI codes are incorporated on the Apricot:

CU : Cursor Movement Functions

MNEMONIC	PARAMETER	CODE	FUNCTION
CUU	n	A	Moves the Cursor UP n lines
CUD	n	B	Moves the Cursor DOWN n lines
CUF	n	C	Moves the Cursor FORWARD n columns
CUB	n	D	Moves the Cursor BACK n columns
CUP	Y;X	H	Positions the Cursor at Y,X
HVP	X;Y	f	Same as CUP, except parameters are reversed

ER : Text Erase Functions

MNEMONIC	PARAMETER	CODE	FUNCTION
* ED	0	J	Erase from cursor to end of screen
* ED	1	J	Erase from start of screen to cursor
ED	2	J	Clear screen, but don't home the cursor
EL	0	K	Erase from cursor to end of line
* EL	1	K	Erase from cursor to start of line
* EL	2	K	Erase the entire line that cursor is on

CP : Report Functions

MNEMONIC	PARAMETER	CODE	FUNCTION
CPR	1;c	R	Put cursor position into keyboard buffer

SCREEN DRIVER

SM : Set Mode Functions

MNEMONIC	PARAMETER	CODE	FUNCTION
LNM	20	h	Auto CR (on reciept of LF)
DECOM	6	h	Origin Mode ON (see note below)
DECAWM	7	h	Auto-wrap at end of line

RM : Reset Modes

MNEMONIC	PARAMETER	CODE	FUNCTION
LNM	20	l	No Auto CR (on reciept of LF)
DECOM	6	l	Origin Mode OFF (see note below)
DECAWM	7	l	No Auto-wrap at end of line

TAC : Text Attribute Change

MNEMONIC	PARAMETER	CODE	FUNCTION
* SGR	0	m	All attributes off
* SGR	1	m	Bold ON
* SGR	4	m	Underline ON
* SGR	7	m	Reverse video ON

SCR : Screen size/mode modification

MNEMONIC	PARAMETER	CODE	FUNCTION
DSR	6	n	Device Status Report - Generates CPR
* DECSTBM	T;B	r	Set T (top) and B (bottom) lines
SCP		s	Save cursor position
RCP		u	Restore cursor position

* = Code is not supported on the IBM PC

Programming Example

The following program produces some text on the Apricot Screen, using the ANSI escape codes:

```
10 an$=chr$(27)+"["    ' an$ is now the ANSI lead-in string
20 print an$ "2J"      ' clear the screen
30 print an$ "10;15H Now at line 10, column 15 !!"
          ' write message
40 print an$ "5A Now at line 5, column 1 !!"
           ' another
50 print an$ "10;14r Screen is now 5 lines high !!";
           'set screen size
60 end
```

Note On the ORIGIN MODE:

When set, line 1 is the top line of the current scrolling region; the cursor cannot be moved (by the ANSI sequences) outside of this area. When reset, line 1 is the top line and the cursor can then move freely around the whole screen. The cursor is homed whenever the origin mode is reset, or the scrolling margins are set.

The screen driver also handles a number of the standard ASCII control codes, which have values less than 31 (20 hex).

The follwing codes are not acted upon, and no characters are output:

0 – NUL	20 DC4
1 – SOH	21 NAK
2 – STX	22 SYN
3 – ETX	23 ETB
4 – EOT	25 EMM
5 – ENQ	26 SUB
6 – ACK	28 FS
16 – DLE	29 GS
17 – DC1	30 RS
18 – DC2	31 US
19 – DC3	32 DEL

These codes, however, perform specific functions:

07 – BEL	Sounds the internal bleeper
08 – BS	Moves the cursor back one space
09 – TAB	Moves the cursor RIGHT in modulos of eight
10 – LF	Move the cursor down one line
11 – VT	As above (LF)
12 – FF	As above (LF)

13 – CR Moves the cursor to the beginning of the current line

14 – SO Switches in the second character set

15 – SI Selects the default character set

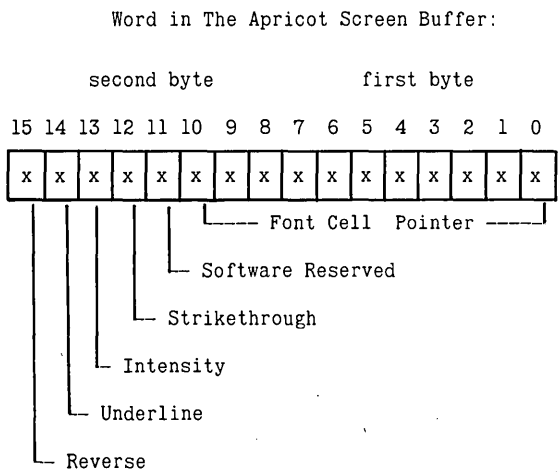
24 – CAN Cancels any escape sequence currently in operation

27 – ESC Herald the start of an escape sequence

Physical Screen Layout

The Apricot screen has a 4K byte block of memory which acts as a screen buffer. This memory is located at F0000 hex and is utilised as follows:

For every character on the screen, there is one word. The first word in the screen buffer is for the first top left-hand character on the screen, the last word is for the bottom right-hand character.



The diagram above shows a graphical representation of a word in the screen RAM. It should be noted that the diagram is in high-low order, so the second byte shown actually comes before first one shown. The attribute bits have the following functions:

1. Reverse Field Video.

When this bit is set, the display hardware will produce black characters on a green background.

2. High/Low intensity

When this bit is set, its corresponding character will be displayed in high intensity (or enhanced, foreground) mode.

3. Underline

If this bit is set, a horizontal line one pixel high will be printed along the bottom of the character. The position in the font cell of the Underline bit determines where the line is to go; thus, underlining is programmable.

4. Strikethrough

This bit determines whether a horizontal line appears through the middle of the character displayed or not.

5. Software reserved

This bit is currently not used by the display harware or software, however future BIOS releases may use it.

SCREEN DRIVER

The 11-bit font cell pointer can point to anywhere within the first 32k (on a WORD only boundary) in the bottom segment of memory. (for example, if the value contained here were to be 0, the character font cell pointed to would be at 0000:0000 in memory) A font cell consists of 16 words or 32 bytes, 1 word for each line of the character:

A character font-cell

```
          0 1 2 3 4 5 6 7 8 9 A B C D E F
        0 . . . . . . . . . . . . . . . .
        1 . . . . . . . . . . . . . . . .
        2 . 1 1 1 1 1 1 1 . . . . . . . .
        3 . . 1 1 . . . 1 1 . . . . . . .
1       4 . . 1 1 . . . 1 1 . . . . . . .
6       5 . . 1 1 . . . 1 1 . . . . . . .
        6 . . 1 1 . . . 1 1 . . . . . . .
w       7 . . 1 1 1 1 1 1 . . . . . x . .  - strikethrough
o       8 . . 1 1 . . . 1 1 . . . . . . .
r       9 . . 1 1 . . . 1 1 . . . . . . .
d       A . . 1 1 . . . 1 1 . . . . . . .
s       B . . 1 1 . . . 1 1 . . . . . . .
        C . 1 1 1 1 1 1 1 . . . . . . . .
        D . . . . . . . . . . . . . . . .
        E . . . . . . . . . . . . . . . .  x - underline bit
        F . . . . . . . . . . . . . . . .

                                  not used
                                   in the
                                  text mode
```

The default character set table starts at 800H in the lower 64k memory segment. The first word's Least significant bit denotes the status of the pixel in the top left-hand corner of the font cell on the screen: if a bit is set, the pixel will be on, if a bit is not set, the pixel will be off.

Here is a program to demonstrate direct screen accessing from Basic:

```
10 PRINT CHR$(27);"z";CHR$(27);"x5"     'Clear & cursor off
20 DEF SEG=&HF000:FOR P=50 TO 255  'Character loop
30 FOR M=1 TO 4000 STEP P      'Screen loop
40 POKE M*2,P  'Put character on screen
50 NEXT M,P:END      'Loop till done
```

SYSTEMS INTEREST

1. CRTC initialization.

The Motorola MC6845 CRTC chip contains 18 internal registers which have to be initialized before any screen I/O can take place. Information on the register functions and the actual data which has to be placed in them to perform various operations can be found in the CRTC description in the Hardware section. Once this information is established, the port addresses used are 68H for the Address register, and 6AH for the control registers.

2. Low level display routines.

These routine runs closely in conjunction with the cursor management code. First of all, the current cursor position is established, the cursor switched off, and the character to be printed on the screen is moved into the position in the screen memory that the cursor occupied. Then the attribute byte (see later) is set. Various tests are then made before the cursor is restored, for example, does the screen need to be scrolled? When any actions have been taken, a short delay (which CAN be interrupted) is executed. This is called the cursor TIMEOUT and is used to stop the cursor appearing when large volumes of text (such as a directory) are being output. When the timeout has expired, the cursor position is updated, and the cursor is switched back on.

3. Cursor management routines.

A record must always be kept of where the cursor is on the screen, in order that the character display routines know where to put the next character, and certain console functions such as report cursor position can work. The cursor is managed by a set of routines which can move it around the screen. The initial cursor mode is known, so the actual position on the screen has to be calculated. Once this is done, the cursor position is sent to the CRTC which then moves its internal cursor reference accordingly.

SCREEN DRIVER

4. How a character appears on the screen

The CRTC controller chip will generate a word address in the screen buffer memory. A part of each word contains the attributes, which are stripped off and passed to the video control logic to be used later. The whole word at this address is then passed into the video shift register, which passes the word (as serial information) to the video output controller, where it is re-united with the (now processed) attribute data and then converted to a series of raw on/off pulses ready for the CRT – after merging them with horizontal & vertical sync signals.

MICROSCREEN DRIVER

INTRODUCTION

The MicroScreen is treated by the BIOS as a serially-accessed device. Not only text output is supported, but the LED's associated with the membrane keys can be switched on and off. It accepts characters sent to it, some of which are special controls (such as clear screen, position cursor etc.) The BIOS has an internal table of the Sirius-compatible escape codes, which operate as an "interpreter" for the user, so no actual machine-level direct access to the MicroScreen is required.

APPLICATIONS INTEREST

Within the BIOS and MS-DOS device table, the ASCII string "MSCREEN" is used as a device name so that it can be referenced when a file is opened, and subsequent output to that file sent to the MicroScreen. Here is a programming example:

```
10 OPEN "0",1,"MSCREEN"
20 PRINT#1,"This Is The MICROSCREEN!"
30 CLOSE 1
```

It will be noted that the output will not appear on the MicroScreen until the file is CLOSEd. This is due to MS-DOS, which saves all serial output of this kind until its internal buffers are full, then it sends the data to the screen. For alternative ways of accessing the MicroScreen, refer to the Device I/O - Control Device section. Table 1 is a list of the escape codes that the MicroScreen recognises. If an un-recognisable escape sequence (including ANSI) is recieved, the cursor position remains unchanged, and no action is taken.

MICROSCREEN

Table 1 – MicroScreen Escape sequences.

ESC /n	Set membrane key LED's where n is a binary number which denotes the LED to be switched on. The last six bits of the byte are one for each LED, the first two bits are arbitrary. For example, this number: (Binary) 00000001 will switch on the left-hand LED, thisnumber: (Binary) 00100000 will switch on the right-hand LED. Character 0 switches OFF all the LED's, character 255 switches ON all LED's. (See Program example).
ESC <	Switch on time and date display
ESC A	Moves the cursor UP one line
ESC B	Moves the cursor DOWN one line
ESC C	Moves the cursor RIGHT one position
ESC D	Moves the cursor BACK one position
ESC E	Clear the screen,homes the cursor
ESC H	Homes the cursor only
ESC J	Erase from cursor to end of MicroScreen
ESC K	Erase from cursor to end of line
ESC U	Enable LCD screen echo
ESC V	Disable screen echo to MicroScreen
ESC W	Send output to the MicroScreen only

ESC Y y x	Send cursor column number (x). (y) is ignored
ESC b	Erase from cursor to start of MicroScreen
ESC c	Disable scrolling - when the cursor reaches the bottom right-hand corner, it is returned to the top-left, and no MicroScreen scrolling takes place.
ESC d	Enable MicroScreen scrolling
ESC e	Cursor ON
ESC f	Cursor OFF
ESC j	Save cursor position
ESC k	Restore cursor to previously saved position
ESC l	Erase entire line, leave cursor
ESC o	Erase from cursor to beginning of line
ESC r	Echo Enable
ESC s	Echo Disable
ESC z	Clear Mscreen & enable screen output.

Program example:

This program presents an easy way of selecting which LED on the membrane pad is illuminated. Place the required pattern of LED's into LED$ as 0's and 1's, and GOSUB 2000. The example below illuminates LED 1 and LED 6.

```
  10 led$="100001":gosub 2000
2000 m=0:for k=1 to 6:m=m+val(mid$(led$,k,1))*2^(k-1):next
2010 print chr$(27)+"/"+chr$(m or 192);
```

MICROSCREEN

Some of the standard ASCII control codes are also supported:

Backspace – Character 8

Move the cursor BACK one position

Linefeed – Character 10

If the cursor is on the first line, it is moved down to the next line, otherwise a scroll UP takes place, and the bottom line is cleared.

Vertab – Character 11

As linefeed

Formfeed – Character 12

As linefeed

Return – Character 13

Move the cursor to the start of the current line.

A standard 256 character set is supplied, containing all the standard ASCII, and some extra graphics characters.

Characters 0 to 31	Non printing control codes
Characters 32 to 127	Standard ASCII character set
Characters 128 to 159	Special graphics characters
Characters 160 to 255	Repeat of the ASCII set

SYSTEMS INTEREST

The MicroScreen is accessed by the BIOS as a serial device. The keyboard queue buffer is used for all communication to the MicroScreen, which is an output device only.

The MicroScreen Character set

	0	1	2	3	4	5	6	7
0	not used	÷	sp	0	@	P	`	p
1	↑	¢	!	1	A	Q	a	q
2	↓	▒	"	2	B	R	b	r
3	±	°	#	3	C	S	c	s
4	£	□	$	4	D	T	d	t
5	\	sp	%	5	E	U	e	u
6	▌	sp	&	6	F	V	f	v
7	g	sp	'	7	G	W	g	w
8	~	sp	(	8	H	X	h	x
9	↑	sp	)	9	I	Y	i	y
A	↓	sp	*	:	J	Z	j	z
B	±	sp	+	;	K	[	k	{
C	£	sp	,	<	L	¥	l	|
D	\	sp	-	=	M	]	m	}
E	▌	sp	.	>	N	^	n	→
F	g	sp	/	?	□	_	o	←

sp = space

KEYBOARD DRIVER

INTRODUCTION

The keyboard software is designed to handle all communications between the serial keyboard, real-time clock, mouse and the MS-DOS 2.0 Operating system.

APPLICATIONS INTEREST

Keyboard Lookup Table

The keyboard lookup table is used by the BIOS to obtain the code(s) for each key on the keyboard. The first 104 words (208 bytes) is the data for the unshifted mode, the next 104 words is the data for the shifted mode, the next 104 words is the data for the control mode. After this there are 400 bytes of string space which is dynamically allocated for the string keys.

The table in all consists of 568 words.

Each word in the table provides the following information:

`Bit 15`	`=0`	Key produces a string of characters, in this case, bits 0 – 14 provide a 15-bit string pointer to a string held in memory.
`Bit 15`	`=1`	Key is a normal non-string or special key and is handled in the normal way.
`Bit 14`	`=0`	Key is not a special key

KEYBOARD DRIVER

`Bit 14 =1` Key is a special key, in this case, bits 0 – 7 provide a special key number which defines the following key types:

00 – NULL key – produces no output
01 – CAPS LOCK key
02 – RIGHT-HAND SHIFT key
03 – LEFT-HAND SHIFT key
04 – CONTROL key
05 – STOP key
06 – CALC key
07 – Mscreen Echo toggle
08 – FF are undefined

`Bit 13 =0` Auto repeat is not allowed on the key

`Bit 13 =1` Auto repeat is allowed

`Bit 12 =0` Key is not affected by shift-lock

`Bit 12 =1` Key is affected by shift-lock

`Bit 11 =0` Key is not affected by caps Lock

`Bit 11 =1` Key is affected by caps lock

`Bit 10 =0` Key is not a local key

`Bit 10 =1` Key is local – output goes direct to the screen

`Bits 9 & 8` Prefix type

00 – No prefix
01 – "ESC" prefix
10 – "ESC [" prefix
11 – "ESC O" prefix

`Bits 7 — 0` Key code – usually 7-bit ASCII, with bit 7 = 0

For a string key the data pointed to by the 15-bit string pointer has the following format:-

Bytes 1 and 2 = String type and character count word.

Bits 0 – 7 is the count of characters in the string (zero is a null string)
Bits 8 – 13 correspond to the above types.
Bits 14 – 15 are not used

Byte 3 ..... Byte n = Character 1 to Last character.

Program 5 allows a multi-string key to be set up from Basic. The key numbers are held in DATA, and the routine is called by GOSUB.

```
  10 GOSUB 50000:STOP
 999 REM--SOME SAMPLE KEY DATA
1000 DATA 4              'NUMBER OF KEYS
1010 DATA "0:8:RUN-":        'MEMBRANE KEY 1 IS RUN WITH A CR
1020 DATA "1:13:LIST-":      'SHIFT MEMBRANE KEY 6 IS LIST CR
1030 DATA "0:2:DIR/W-":           'REPEAT IS DIR/W CR
1040 DATA "0:0:HELP":             'HELP IS HELP, NO CR
1050 REM--THE DATA IS HELD AS FOLLOWS:
        MODE:KEYNUM:STRING [-]
    MODE   - 0=UNSHIFTED, 1=SHIFTED, 2=CONTROL
    KEYNUM - THE NUMBER OF THE KEY TO BE RE-DEFINED FROM 1 TO 96
    STRING - THE NEW STRING TO APPEAR ON THE KEY, IF A "-" IS
             APPENDED, THEN A CARRIAGE RETURN IS ADDED TO THE CODE
50000 READ N:FOR E=1 TO N:READ KEY$
50010 IF RIGHT$(KEY$,1)="-" THEN MID$(KEY$,LEN(KEY$))=CHR$(13)
50020 GOSUB 50030:NEXT E:RETURN
50030 DEF SEG=0:ST=PEEK(&H0712)+256*PEEK(&H0713)
50040 TL=PEEK(&H0716)+256*PEEK(&H0717)-1
50050 KB=PEEK(&H0714)+256*PEEK(&H0715):DEF SEG=KB
50060 M=VAL(KEY$):Y=VAL(MID$(KEY$,INSTR(INSTR(KEY$,":")+1))
50070 O$=MID$(KEY$,INSTR(INSTR(KEY$,":")+1,KEY$,":")+1):SL=LEN(O$)
50080 L=(ST+(M*208)+Y*2):FOR K=ST+TL TO ST+623 STEP -1:
             IF PEEK(K)=0 THEN 50090 ELSE 50100
50090 NEXT K:IF K=ST+623 THEN K=K+2
50100 K=K+1:POKE K,SL:POKE K+1,0:FOR P=1 TO SL:
              POKE P+K+1,ASC(MID$(O$,P)):NEXT
50110 DEF SEG:G%=K:P1=PEEK(VARPTR(G%)):P2=PEEK(VARPTR(G%)+1)
50120 DEF SEG=KB:POKE L,P1:POKE L+1,P2:RETURN
```

KEYBOARD DRIVER

SYSTEMS INTEREST

The software consists of a number of modules, and includes the following specific areas:

1. Keyboard Unit to Apricot serial data handler
2. Key make code (down-code) handler
3. Key break code (up-code) handler
4. Keyboard Data Translation Handler
5. Auto-repeat handler
6. Non-make/break code interpreter
7. Queue handlers for data flow to MS-DOS, screen, and keyboard

1. Keyboard Unit to Apricot - serial data handler

Serial data from the keyboard unit is received into port B of the Z80 SIO chip, which causes a receive interrupt routine to be activated.

This routine, in the SIO handler module, filters out valid codes from the keyboard and sends these to the main keyboard handler routine.

The handler routine in turn filters out all codes not relating to key make or break operations (up or down codes), and passes these on the the non-make/break code interpreter.

2. Key MAKE CODE (down-code) handler.

This sub-module first converts the raw down-code to a number in the range 0 - 103 (for the possible 104 key positions on the keyboard - 96 of which are currently used).

It then looks up the key make code in the keyboard lookup table, which define the key outputs in the unshifted, shifted or control (alt) modes; and the type of key which it is. A key can have any valid combination of the folloing attributes for each of the three modes:

Key produces a string of characters (a STRING key)
Key is allowed to Auto-repeat
Key is affected by Shift Lock
Key is affected by CAPS lock
Key is "local" – output goes direct to the screen driver
Key has "ESC" prefix (used for escape sequences – usually local)
Key has "ESC [" prefix (used with the ANSII driver)
Key has "ESC O" prefix (also for ANSII)
Key is valid for use by the calculator
Key is special, i.e. SHIFT, CONTROL, CAPS LOCK, STOP, CALC, NULL

Special keys are filtered-off to a special key subroutine where the following actions are taken:-

NULL Ignored – no output is sent to any queue.

CAPS LOCK If the keyboard is already in shift lock or caps lock then this mode is cleared, and the CAPS LOCK LED is extinguished. If the keyboard is in control mode then the shift-lock status is activated, otherwise the CAPS lock status is activated, and the CAPS lock LED is illuminated.

SHIFT Right or left - the SHIFT flag is set, so the keyboard is now in the SHIFT mode.

CONTROL The CONTROL flag is set, so the keyboard is now in the immediate CONTROL mode.

KEYBOARD DRIVER

STOP If the keyboard is in the STOP mode, then this is released and the stop LED is extinguished, otherwise the stop LED is illuminated, and the STOP mode is asserted.

CALC If the calculator is OFF, a "wake-up" signal is sent to the internal BIOS calculator which then sets up the MicroScreen display (saving the original contents) and sets the CALCULATOR ON flag. If the key is pressed when the calculator is on, a RESET signal is sent to the internal calculator, which then re-initialises itself.

For normal or string keys, the routine then checks whether the keyboard is in the STOP mode, if it is, then the STOP mode is released, and the STOP key LED is extinguished. This means that any key depression (exept for a 'special' key) can be used to release the STOP mode.

The key number is placed in a 16-byte down-buffer where it is held until the key is released, if the buffer is full then the number is simply forgotten - in either case, the count of keys held down is incremented by 1.

If the number of keys held down is one, then the AUTO-REPEAT lead-in delay count is initialised.

The CALCULATOR ON flag is then checked to see if we are in the CALCULATOR MODE. If so, then the key is checked via a special bit-map to see whether it should be sent to the calculator module, if it is, then the key number (raw down-code minus 1) is sent to the calculator module. The module performs all error checking, MICROSCREEN output etc.

The key is then checked for "local" type – if it is then the internal destination flag is set to screen output, rather than MS-DOS output.

The prefix-bits of the key type are then checked and the appropriate ESC prefix (ESC, ESC [or ESC O) codes are sent to the output queue.

The key type is then checked for a string key – if it is then the string is extracted from memory, and sent, a character at a time, to the output queue – if there is room. If there is not enough room to fit the string in, then the output is aborted and the internal bell is sounded to serve as an alarm.

Otherwise the key code is sent to the output queue.

3. Key Break code (up-code) handler

Raw up-codes are first converted by this routine to the range 0-103, they are then converted via the keyboard lookup tables to give the key type and data (as in the down-code handler).

Special keys are filtered out and handled as follows:

CAPS LOCK,STOP,CALC keys

ignored (since these are toggling keys, only the make code needs to be acted upon).

NULL key

ignored

CONTROL key

The CONTROL KEY flag is reset – the keyboard is now in the normal mode.

LEFT or RIGHT SHIFT

The SHIFT KEY flag is reset - the keyboard is now in the normal mode.

A normal, or string key, has its entry in the down-code buffer removed (if it is there) and the count of keys held down is decremented by one.

If the down-count is now one, the lead-in Auto Repeat delay count is intialised.

KEYBOARD DRIVER

4. Translation of data routines

These routines use the keyboard lookup tables to establish the data and attributes of a key.

5. Auto Repeat handler

The auto repeat handler is called by the timer interrupt routine every 20ms, the routine first checks to see if the number of keys down is one – if not it aborts.

If the count is 1 then the clock rate is divided to give the basic auto repeat period (e.g. 10cps = 100ms per char – 5 20ms periods).

If an auto-repeat rate period has been reached, then the current mode is checked to see if the keyboard is in the STOP mode - if so then the routine does not repeat.

The lead-in delay count is then decremented, if it is not zero then the routine aborts, otherwise an "auto-repeat" operation is performed:

The CALCULATOR ON flag is checked – if the calculator is on, or the key is a special key then no auto repeat is done.

If all is okay, the previous key sequence is repeated, then the Auto Repeat module is re-entered. This continues until the key is released (up-code).

6. Non make/break code interpreter

This part of the keyboard software is used as a junction for other keyboard-related operations such as date/time, mouse codes and Mscreen feedback. In each case control is passed to the relevant device driver.

7. Queue handlers for data flow to MS-DOS or screen.

The keyboard software contains 3 ring buffers, or queues, to pass data between various asynchronous software modules.

These are:

a. Queued data from the keyboard handler to MS-DOS
b. Queued data to the screen handler (local keys from the keyboard)
c. Queued data from the MicroScreen driver, LED driver, Time and Date handler.

a. The MS-DOS queue:

This consists of three routines, a queue filler, a look-ahead character checker, and a queue reader.

The queue filler is called from the main keyboard handler and places the character in the 80-byte buffer.

The caller also passes a count of how many characters it wishes to place sequentially in the buffer, if there is not enough room for the entire string in the buffer, then the queue filler does not accept the character, sounds the bell, and returns an error status.

The character look-ahead routine is called from the MS-DOS BIOS interface routines to provide a keyboard look-ahead, and simply returns the next output character in the queue, without removing it from the queue (if no character is available it returns a null code).

The queue reader is called by MS-DOS with an address where a character from the queue should be placed, if no character is available, the routine waits for one.

b. Local queue:

The local queue filler is called by the keyboard handler to pass characters to the screen driver, and operates in much the same way as the MS-DOS queue handler.

KEYBOARD DRIVER

c. Keyboard Unit queue:

This queue is used to pipe data from the Apricot to the keyboard unit. The queue filler places a single character in the buffer, if there is no room, the character is ignored, otherwise the character is transmitted to the keyboard, where it is either displayed on the Microscreen or, if it is a control character, is executed.

CONTROL DEVICE DRIVER

INTRODUCTION

The Apricot's devices are contained in both the BIOS and MS-DOS device lists. One example that has been seen already is the MicroScreen, that is opened as a normal file from within Basic in MS-DOS. The Apricot has many more external hardware devices than most other machines, and because of the way they are incorporated within the BIOS, extra hardware systems can be accessed far more easily than on other systems. In fact, the same routine can be used from within a program to access everything from the Screen Driver circuitry to the Mouse, and data can be passed both to and from these devices.

At present, there are nine devices catered for within the Device I/O handling, and more hardware and software devices are under development at ACT.

APPLICATIONS INTEREST

In the BIOS overview the "BIOS control device" was mentioned. This is a routine which starts at 600H (within the Pointer area) and can be called from an applications program. Parameters are passed to either change the way that the BIOS handles certain devices (such as the communications ports) or access these devices for Input/ Output operations.

Calling The Control Device

The routine requires (in all) three items of data to be supplied to it, and it returns one:

Parameters Supplied:

1. **DEV – Device Number**
2. **COM – Command**
3. **DAT – Data to be sent**

Parameters Returned:

1. **RET – Variable for Data/Status return**

CONTROL DEVICE

When the routine is called as an absolute jump to the location 0000:0600H, the first three items above have to be pushed (as words) onto the stack. The routine can also be accessed through INTERRUPT 0FCH, which machine language programmers will find more convienient. In this case, the parameters are passed in the registers like so:

BX – Device number (DEV)
CX – Command (COM)
DX – Data (DAT)

And the status is returned as:

AX – Status/Data (RET)

All the 8086 registers & the flags are saved

The three items of data are all numeric values. The device number is one of the following:

ASCII	HEX	DEC	DEVICE NAME
1	31H	49	Screen driver
2	32H	50	Keyboard driver
3	33H	51	MicroScreen driver
4	34H	52	Serial Input/Ouput driver
5	35H	53	Parallel Input/Output driver
6	36H	54	Mouse driver
7	37H	55	Clock driver
8	38H	56	Sound generator/driver
9	39H	57	Floppy Disk Drivers
B	42H	66	Cache/Graphics/IBM config

The command, data and return status all vary depending on which device is being accessed.

Specification Of The Apricot Control Device

a. Screen Driver

The Screen driver can easily be accessed by using this configuration call:

DEV – 49
COM – 0 to 2
DAT – 0 or 1

COM = 0:

Return the current status of the video display – RET returns 1 if no display is connected, or 0 if it is.

COM = 1:

Get or set text and graphics mode. If DAT = 0, then the Text mode is selected, or if DAT = 1, the graphics mode is selected.

COM = 2:

Switch the screen display on or off. If DAT = 1, the screen display will be switched off, or if DAT = 0, the screen display will be switched on.

b. The keyboard driver

This option allows the programmer to set the Auto Repeat Key status, as defined under the Keyboard section.

DEV – 50
COM – 0 to 5
DAT – variable

COM = 0:

The keyboard status will be returned in RET; 0 means that the keyboard is connected AND is a valid ACT Apricot Keyboard, 1 means that there is no keyboard connected.

CONTROL DEVICE

COM = 1:

The Auto Repeat feature on some keys can be switched on or off. Note that Auto Repeat is only applicable to those keys which have an auto repeat attribute.

Value In DAT	Desired Auto Repeat Status
0	Disable
1	Enable

COM = 2:

The Auto Repeat Lead-in delay can be set, this is the ammount of time that a key has to be held down before it will auto-repeat. DAT is from 1 to 255, each value adds another 20 milliseconds onto the repetition rate.

COM = 3:

The Auto-Repeat-Rate can be set, this is the ammount of time it takes in between repetitions of the key when it is held down, ie. the Repeat Speed. DAT is from 1 to 255, each value adds another 20 milliseconds onto the Lead-in Time.

COM = 4:

The raw downcodes and upcodes are sent to MS-DOS instead of the values of the keys. For example, UNDO, which is key 2 on the keyboard would generate a character 2 (CHR$(2)) in the keyboard buffer, and when it is released it will generate a character 2 plus 80H (128 decimal) this can be used to check if a key is being held down, for applications such as games. To return to normal, either set DAT to 1 and re-call or press the HELP key.

COM = 5:

DAT is ignored and the internal keyboard buffer is cleared and reset.

c. MicroScreen Driver

This option allows the programmer to bypass the normal method of accessing the MicroScreen, and send characters to the driver directly.

DEV – 51
COM – Either 0 or 1
DAT – From 0 to 255

COM = 0:

DAT is ignored and RET will either contain 0, which means that the MicroScreen is ready to receive characters, (ie. is online and working,) or –1 which means that the MicroScreen is in some way out of action.

COM = 1:

The character held in DAT is printed to the MicroScreen. As DAT is two bytes long, the character to be printed is held in the low-byte. All the MicroScreen-specific escape sequences can be used as normal.

Programming Example:

```
 10 REM -- Program to write ACT on the MicroScreen --
 20 DEF SEG=&H60    'Set default segment
 30 IO=0   'IO indexes start of the Device I/O routine in memory
 40 DEV%=51     'Select the device number to be the MicroScreen
 50 COM%=1      'Command is 1, so we want to OUTPUT
 60 RET%=0      'Dummy variable to hold Returned status
 70 ' Send an A
 80 DAT%=ASC("A")    'Set up data in DAT%
 90 CALL IO(DEV%,COM%,DAT%,RET%)    'Call I/O routine to print character
100 ' Send a  C
110 DAT%=ASC("C")    'Set up data in DAT% again
120 CALL IO(DEV%,COM%,DAT%,RET%)    'Call to I/O
130 ' Send a  T
140 DAT%=ASC("T")    'Set up a "T" in DAT%
150 CALL IO(DEV%,COM%,DAT%,RET%)    'Call to I/O
160 DEF SEG:END     'End program
```

CONTROL DEVICE

d. Serial Port Driver

This is the most powerful of all the Device Modification routines, this option allows the programmer to send and receive characters and change the SIO (Serial I/O) settings (Both Transmit AND Recieve BAUD rates, Stop Bits, Parity, Transmit Bits, Recieve Bits, in addition to control line status)

DEV – 52
COM – From 1 to 16
DAT – 0 to 255 for character to send

COM = 0:

The SIO chip is polled to establish as to wether the port is ready to send or receive, and the result is returned in RET – 0 the port is ready, –1 Not ready.

COM = 1:

The low byte of DAT is transmitted.

COM = 2:

DAT is ignored, and RET returns the ASCII code of the character just received. If, however, RET returns –1 (0FFFFH) then there is no character to receive.

Programming Example

```
 10 REM -- Program to emulate a teletype terminal. Half Duplex Mode
 20 REM -- Any character received is printed on the screen,
 30 REM -- Any key pressed is echoed on the screen AND sent.
 40 '     Setup default variables:
 50 DEF SEG=&H60     'Segment for IO Routine
 60 IO=0  'IO now points to device I/O routine
 70 DEV%=52     'Setup the device number
 80 RET%=0      'Setup an empty return variable
 90 '     Start
100 PRINT CHR$(27) "z" CHR$(27) "2"    'Clear screen & flash cursor
110 GOSUB 800 'Get CHAR
120 IF CHAR<>-1 THEN PRINT CHR$(CHAR);    'Print it?
130 IN$=INKEY$: IF IN$="" THEN 120        'Is there a key ?
140 CHAR=ASC(IN$):GOSUB 900:PRINT IN$;:GOTO 110 'Send and Print key
800 REM -- Routine to GET a character in CHAR
810 REM -- Returns CHAR=-1 if no character is available
820 COM%=2:DAT%=0:CALL IO(DEV%,COM%,DAT%,RET%)     'Get character
830 CHAR=RET%:RETURN 'Back to caller
900 REM -- Routine to SEND a character in CHAR
910 COM%=1:DAT%=CHAR:CALL IO(DEV%,COM%,DAT%,RET%)     'Send character
920 CHAR=RET%:RETURN 'Back to callerD
```

COM = 3:

Update the SIO with the parameters that are set using codes 4 to 14.

COM = 4:

Set the TRANSMIT BAUD RATE to the code held in DAT (1 to 15)

CONTROL DEVICE

COM = 5:

Set the RECEIVE BAUD RATE to the code held in DAT (1 to 15)

Value in DAT	BAUD rate
01	50
02	75
03	110
04	134.5
05	150
06	300
07	600
08	1200
09	1800
10	2400
11	3600
12	4800
13	7200
14	9600
15	19200

COM = 6:

Set the TRANSMIT BITS PER CHARACTER to the code held in DAT (DAT is between 5 and 8 inclusive)

COM = 7:

Set the RECEIVE BITS PER CHARACTER to the code held in DAT (DAT MUST be between 5 and 8 inclusive)

NOTE: The Number In DAT denotes the required Number of Bits

COM = 8:

Set the number of STOP BITS PER CHARACTER to the code in DAT (1 to 3)

Value in DAT	Desired Number of STOP BITS
1	ONE
2	ONE AND A HALF
3	TWO

COM = 9:

Set the PARITY TYPE to the code in DAT (0 to 2)

Value in DAT	Desired Parity Type
0	No Parity
1	Odd Parity
2	Even Parity

COM = 10:

The current XON/XOFF TRANSMIT protocol is set according to a code held in DAT.

COM = 11:

The current XON/XOFF RECEIVE protocol is set according to a code held in DAT.

Value in DAT	Desired Protocol Status
0	OFF
1	ON

CONTROL DEVICE

COM = 12:

The RTS line can be set or reset according to a code held in DAT:

Value in DAT	Desired RTS line status
0	reset
1	set

COM = 13:

The DTR line can be set or reset according to a code held in DAT:

Value in DAT	Desired DTR line status
0	reset
1	set

COM = 14:

The Transmit Feature can be enabled or disabled:

Value in DAT	Desired Transmit Status
0	disable
1	enable

COM = 15:

The Receive Feature can be enabled or disabled:

Value in DAT	Desired Receive Status
0	disable
1	enable

COM = 16:

DAT is ignored and RET gives the current CTS line status:

Value returned in RET	CTS Line Status
0	reset
1	set

COM = 17:

DAT is ignored and RET returns the current DCD line status:

Value returned in RET	DCD Line Status
0	reset
1	set

COM = 18:

The DSR status is returned in RET, and DAT is ignored:

Value returned in RET	DSR Line Status
0	reset
1	set

IMPORTANT NOTE:

Note that any alteration made by codes 4 to 16 will not be invoked until code 3 is used. Codes 4 to 16 will return the value that was sent along with them to confirm successful alteration. If, however, the data is out of range (Ideally –1) then the current status will be returned in RET and no alteration will take place.

CONTROL DEVICE

Program Example

```
 10 REM -- Program to re-configure the Apricot serial port
 20 '     Set up numeric constants
 30 DEF SEG=&H60     'Active segment to where control block is
 40 DEV%=52     'Setup device number
 50 RET%=0:DAT%=0    'Reset RET and DAT
 60 IO=0         'IO is now where routine is in memory
 70 '     Get current default status
 80 DAT%=-1     'Rogue value to return status
 90 COM%=4      'Get current Xmit BAUD Rate
100 CALL IO(DEV%,COM%,DAT%,RET%)    'Call IO
110 XBAUD=RET% 'Update Variable
120 COM%=5      'Get current Recv BAUD Rate
130 CALL IO(DEV%,COM%,DAT%,RET%)    'Call IO, remember DAT%
140 RBAUD=RET% 'Update Variable
150 COM%=6      'Get Xmit Bits/Char
160 CALL IO(DEV%,COM%,DAT%,RET%)    'Call IO
170 XBITS=RET%
180 COM%=7      'Get Recv Bits/Char
190 CALL IO(DEV%,COM%,DAT%,RET%)    'Call IO
200 RBITS=RET%
210 ' We won't bother with Stop bits or Parity
220 ' So now, XBAUD & RBAUD = Xmit and Recv BAUD RATES
230 '         XBITS & RBITS = Xmit and Recv BITS
240 PRINT"Current Values :"
250 PRINT"Transmit BAUD = " XBAUD " Receive BAUD = " RBAUD
260 PRINT"Transmit BITS = " XBITS " Recieve BITS = " RBITS
270 INPUT"Enter New Transmit BAUD :" ; XBAUD
280 INPUT"Enter New Recieve  BAUD :" ; RBAUD
290 INPUT"Enter New Transmit BITS :" ; XBITS
300 INPUT"Enter New Recieve  BITS :" ; RBITS
310 '     Update BAUD
320 COM%=4:DAT%=XBAUD
330 CALL IO(DEV%,COM%,DAT%,RET%)    'Update Transmit BAUD
340 COM%=5:DAT%=XBAUD
350 CALL IO(DEV%,COM%,DAT%,RET%)    'Update Recieve BAUD
360 '     Update BITS
370 COM%=6:DAT%=XBITS
380 CALL IO(DEV%,COM%,DAT%,RET%)    'Update Transmit BITS
390 COM%=7:DAT%=RBITS
400 CALL IO(DEV%,COM%,DAT%,RET%)    'Update Recieve BITS
410 '     Intermediate values now set, so go and update the SIO
420 COM%=3      'Value for SIO update
430 CALL IO(DEV%,COM%,DAT%,RET%)    'Update the SIO
440 '     Everything done
450 PRINT "DONE":DEF SEG:END        'End
```

e. Parallel I/O Driver

DEV – 5
COM – 0 to 6
DAT – Not Used

COM = 0

Return the Parallel status.

COM = 1

This option will return the amount of free bytes left in the print buffer is returned in RET.

COM = 2

Set the FAULT line status:

DAT = 0 – Disable
DAT = 1 – Enable

COM = 3

Set the SELECT line status detect:

DAT = 0 – Disable
DAT = 1 – Enable

COM = 4

Set the PAPER ERROR line status detect:

DAT = 0 – Disable
DAT = 1 – Enable

COM = 5

Set the Auto LF after CR enable:

DAT = 0 – Disable
DAT = 1 – Enable

CONTROL DEVICE

COM = 6

Set the default printer type:

DAT = 0 – Parallel
DAT = 1 – Serial

f. Sound Generator

This option allows the programmer to vary the volume of the sound generator

DEV – 56
COM – 0, 1 or 2
DAT – 0 to 15

COM = 0:

RET gives the status (0 = sound generator working, –1 = sound generator not working)..

COM = 1:

DAT gives the new volume level for the key click. DAT can be from 0 (maximum volume) to 15 (minimum volume). If DAT is out of range (ideally –1) then the current volume level of the feedback click is returned in RET:

Program example

```
10 rem - program to increase the key click volume
20 def seg=&h60:io=0:ret%=0
30 dev%=56                      'device number
40 com%=1                       'command number 1
50 dat%=1                       'highest level
60 call io(dev%,com%,dat%,ret%)     'do change
```

COM = 2:

DAT gives the new volume level for the bell. DAT can be from 0 (maximum volume) to 15 (minimum volume). If DAT is out of range (ideally –1) then the current volume level of the bell is returned in RET.

g. Floppy Disk Drivers:

DEV – 39H
DAT – 0 – 65535
COM – 0 – 4

COM = 0

Return the floppy disk status.

COM = 1

Set the drive number for Format:

DAT = 0 – Drive 0
DAT = 1 – Drive 1

COM = 2

DAT contains the segment of the track image to be placed on the disk

COM = 3

DAT contains the offset of the track image from the segment supplied using call 2.

CONTROL DEVICE

COM = 4

Format the disk using the parameters set up above.

h. Cache/Graphics/IBM configuration

This choice selects how the memory area used for Cache, Graphics, or IBM PC emulation is to be used by the BIOS.

DEV – 42H
COM – 0 to 3
DAT – 0 to 2

COM = 0

Return the status in DAT. If 0 is returned, then the user is free to change the usage factor of the memory. IF 0FFFH is returned the user is still free to change the usage factor, however, the BIOS may still be using the memory.

COM = 1

Switch the Disk CACHE on or off:

DAT = 0 – On
DAT = 1 – Off

COM = 2

Set the Graphics on/off flag in the parallel driver:

DAT = 0 – On
DAT = 1 – Off

COM = 3

Set/Reset the IBM PC emulation flag:

DAT = 0 – On
DAT = 1 – Off

SYSTEMS INTEREST

The Control Device is essentially a tree-structured set of jump tables. When program execution is first passed to the control device entry point, DEV is checked for legality, and then Jumped upon to the next branch in the tree, where COM is checked and then jumped upon to the appropriate configuration procedure, where DAT is then jumped upon again passing control to the actual routine required, which could be anything from setting a flag to configuring the Z80 SIO.

DISK INPUT/OUTPUT SYSTEM

INTRODUCTION

The BIOS acts in conjunction with MS-DOS 2.0 where disk I/O is concerned. MS-DOS will call a routine within the BIOS, passing parameters such as where on the disk to write the data, how much data to write and where the data is held in memory. The BIOS will then inform the floppy disk controller of what has to be done, which will then perform the actual write, by moving the head to the correct place on the disk and then sending the actual data to the disk controller which will write it to the disk.

APPLICATIONS INTEREST

All applications level programming should only use the disk handling instructions provided by the language that is being used or, in exceptional circumstances, the MS-DOS interrupts can be used to read and write files. Some external utilities such as DEBUG will allow the user to write absolute disk sectors. It should be stressed that under no circumstances must any sectors be changed unless there is a full knowledge of what is being done. The actual way that the Apricot uses the disk controllers is sufficient for most software.

SYSTEMS INTEREST

Most input/output to the disk controller is performed by the 8089. The 8089 uses a parameter block (located in memory) which is set up by the BIOS to contain various pieces of information relating to the relevant disk operations. Then a signal is sent to the 8089 which reads the parameter block and takes action on its contents.

It will become apparent if any experimentation is carried out that all the physical details of how the disk is written or read are handled by the floppy disk controller. The BIOS passes commands and parameters to the Floppy disk controller and then invokes the 8089 (as described above). The 8089 is used in only two of its operation modes, and

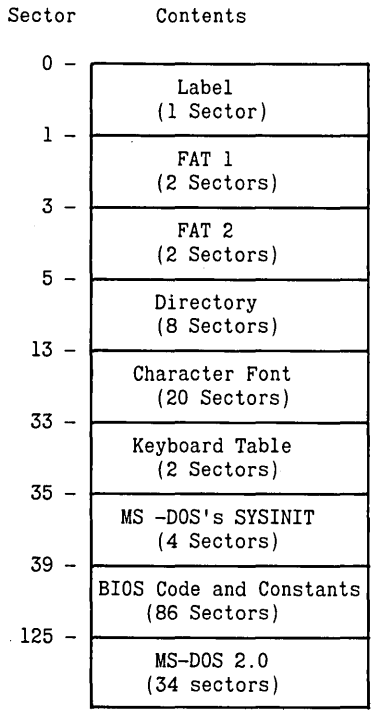
that is transferring data from port to memory (on READ) and memory to port (on WRITE).

A background "demon" is used, which makes regular checks in conjunction with the clock interrupt on the floppy disk drives to ensure that the user has not swapped disks. The demon will be activated whenever a disk has not been accessed or checked for more than 1.5 seconds - theory has it that it takes at least 2 seconds to change a disk. If a disk has been removed, the cache is flushed of all sector blocks referring to that disk.

Disk Formats

The first 158 sectors (80896 bytes) on each Apricot System disk are reserved for the operating system.

Apricot Disk Sector map of the First 158 Sectors.



The Label Sector

The Apricot disk Label sector is the first 512 bytes on the disk (1 sector). Only the first 128 bytes are relevant for the boot ROM locating where the operating system is on the disk, where to load it in RAM, and where to execute it from when loaded. These 128 bytes are divided as follows:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
LBLform_vers	WORD	0,0,0,0	Version of FORMAT that created disk
LBLop_sys	BYTE	1	0=invalid 1=MSDOS 2.0 2=p-system 3=CP/M-86 4=Concurrent CP/M 5=BOS 6=UNIX
LBLsw_prot	BYTE	0	Software Write Protect 1=disk is protected 0=disk is not
LBLcopy_prot	BYTE	0	Not used
LBLboot_disk	BYTE *	1	Must be 1 if a valid boot disk (0 on data only disks)
LBLmulti_region	BYTE	0	Not Multi-regioned
LBLwinchester	BYTE	0	Not a Winchester Disk
LBLSec_size	WORD	512	Sector Size
LBLSec_track	WORD	9	Sectors per track
LBLtracks_side	WORD	70,0	Tracks per side
LBLsides	BYTE	1	Number of sides
LBLinterleave	BYTE	1	Interleave factor
LBLskew	WORD	5	Skew factor
LBLboot_locn	WORD *	29,0	Sector of boot image

DISKS

LBLboot_size	WORD *	112	Number of bootstrap sectors
LBLboot_addr	WORD *	00H,0D80H	boot load address (0D80:0000H)
LBLboot_start	WORD *	29H,0E00H	boot start address (0E00:0029H)
LBLdata_locn	WORD	13,0	Sector No.of first data block
LBLgeneration	WORD	0	Generation number
LBLcopy_count	WORD	0	Copy count
LBLcopy_max	WORD	FFFFH	Maximum number of copies
LBLserial_id	WORD	0,0,0,0	Serial Number
LBLpart_id	WORD	0,0,0,0	Part Number
LBLcopyrite	ASCII	(C) ACT	Copyright or Information data

Those data items marked * are vital to the operation of the current Boot ROM, the rest may be used by future BIOS releases.

MS-DOS AND BIOS UTILISATION OF THE DISKS

FAT's 1 and 2

The FAT (File Allocation Table) is what MS-DOS uses to remember where on the disk each file will go.

Directory

The directory is MS-DOS's list of contents on a disk. Pointing to the first FAT entry for a file. Other information such as file size and creation date is kept in the directory.

Character Font

A memory image of the character font which will be directly loaded into memory. The last 2kbytes of the character font is the Apricot Logo which is loaded into its respective place in memory.

Keyboard Table

This is the keyboard table image.

SYSINIT

The SYSINIT code used to initialize MS-DOS 2.0.

BIOS code

Another direct memory image containing the BIOS program.

MS-DOS 2.0

MS-DOS 2.0 itself, a totally self-contained module.

The Configuration data:

For MS-DOS, the next data is used at initialisation time:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
LBLBPBsctr_sz	WORD	512	Sector size in bytes
LBLBPBclu_sz	BYTE	1	Cluster size in sectors
LBLBPBrsvd_sct	BYTE	1	Number of reserved sectors
LBLBPBn_fats	BYTE	2	Number of FATs
LBLBPB_dir_ent	WORD	128	Number of dir entry(s)
LBLBPBn_sec	WORD	630	Number of sectors on disk
LBLBPBmedia_id	BYTE	0FCH	Media number/identification
LBLBPBn_fat_s	WORD	2	Number of sectors in a FAT
	BYTE	?	Not used
	WORD	?	Not used
LBL_font_name	ASCII	'FONT=BRIT02'	Font Name
LBL_keys_name	ASCII	'KEYS=ACT001'	Keyboard name

DISKS

Next in the label sector is the configuration data, which is changed by the system configuration package. The sample data given below is for a typical System Disk:

System Unit:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
CNF_drive0	BYTE	1	70 track drive
CNF_drive1	BYTE	1	70 track drive
CNF_diagflg	BYTE	0	Diagnostics on or off?
CNF_lst_dev	BYTE	0	Parallel LST: device
CNF_bell_vol	BYTE	4	Default Bell volume
CNF_cache_on	BYTE	1	Primary of secondary Cache?
CNF_graph_on	BYTE	0	Graphics on or off?
	BYTE	0	Not used
	BYTE	0,0,0,0,0,0,0,0	Not used

Keyboard:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
CNF_Click_vol	BYTE	8	KB click volume
CNF_rept_en	BYTE	1	Auto-repeat master enable
CNF_rept_dly	BYTE	25	Lead-in delay, 25*20ms = 500ms
CNF_rept_int	BYTE	5	Repeat interval rate
CNF_Mscrn_mode	BYTE	0	MScreen echo
	BYTE	0,0,0	Not used
	BYTE	0,0,0,0,0,0,0,0	Not used

Screen:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
	BYTE	0,0,0,0,0,0,0,0	Not used
	BYTE	0,0,0,0,0,0,0,0	Not used

Serial or AUX communications:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
CNF_Tx_brate	BYTE	14	Tx = 9600 baud
CNF_Rx_brate	BYTE	14	Rx = 9600 baud
CNF_Tx_bits	BYTE	8	8 bits/char
CNF_Rx_bits	BYTE	8	"
CNF_STOP_bits	BYTE	2	1.5 stop bits
CNF_paritychk	BYTE	0	Parity check off
CNF_paritytyp	BYTE	0	No Parity
CNF_tx_xonof	BYTE	0	Tx XON protocol
CNF_rx_xonof	BYTE	0	Rx XOFF protocol
CNF_rx_x_limit	WORD	30	XON/XOFF Rx buffer limit
CNF_dtr_dsr	BYTE	0	No DTR/DSR protocol
CNF_cts_rts	BYTE	0	No CTS/RTS protocol
CNF_CR_Null	BYTE	0	Number of Nulls after CR
CNF_FF_Null	BYTE	0	nulls * 10 to send after FF
	BYTE	0,0,0,0,0,0,0	Not used
	BYTE	0,0,0,0,0,0,0,0	Not used

Parallel Comms:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
CNF_p_CR_LF	BYTE	0	No CR after LF
CNF_select	BYTE	1	Select Line supported
CNF_pe	BYTE	1	Paper Empty supported
CNF_Fault	BYTE	1	Fault Line supported
	BYTE	0,0,0,0,0,0	Not used
	BYTE	0,0,0,0,0,0,0	Not used

Winchester Disk:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA

RAM Disk:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA

Spare:

MNEMONIC	TYPE	SPECIMEN VALUE	DESCRIPTION
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA
	BYTE	0,0,0,0,0,0,0,0,0,0	NYA

BOOT ROM

INTRODUCTION

The Apricot's Internal Boot ROM contains the software which is executed first of all when the machine is switched on. The ROM contains a calculator, and a number of software and hardware tests. When a disk is inserted and the conditions for booting are satisfactory, the boot ROM will load in the operating system and execute it. Below is a list of the boot ROM's operation.

1. Initialize Hardware.
2. Print logo and "Testing" message on the screen.
3. Perform Software tests on hardware.
4. Display the results of the tests.
5. Cycle in an enless loop testing for the Calculator key or a disk being inserted.
6. When a disk is inserted, check for validity as a system disk and read in then execute the Bios.

APPLICATIONS INTEREST

Very little use can be made of the boot ROM from applications software. One of the features it has is an ability to "cold-boot" the machine. The ROM entry point (At 0FFFF hex) has to be jumped to.

Programming Example : Software Reset

```
5000 rem -- "GOTO 5000" will have the same effect as pressing reset.
5010 def seg=&hffff:boot=0:call boot
```

The Logo that is loaded from disk can also be displayed, if the standard Apricot logo is used, it will be a copy of the one in the Boot ROM

Programming example : Logo Display Program

This routine will display the Apricot Logo in the centre of the screen at the top:

```
10 def seg=&Hf000:o=-2:for l=195 to 207
20 o=o+1:poke l*2+160,114+o:poke l*2+161,67
30 poke l*2,93+o:poke l*2+1,67:next
40 poke 252,82:poke 253,67:poke 242,77:poke 243,67
```

CALCULATOR

INTRODUCTION

The Apricot's Internal Calculator is designed to be used either before the system is booted-up, without the need to insert a disk, or from within an application where calculations can be performed and the results (either intermediate or final) can be sent to the MS-DOS input queue as if they had been typed in from the keyboard.

APPLICATIONS INTEREST

The Calculator can be operational in one of two modes: Before boot-up in isolation or when the BIOS is loaded and an application is running.

The Calculator code itself resides permanently in the boot ROM, and when the pre boot-up sequence is in operation, the keyboard is scanned once every 20ms to check if the CALC key is depressed. If so, the calculator function is engaged, the MicroScreen is set up and calculations can begin.

Within the calculator code, mathematics is performed upon ASCII numbers using the standard Intel 8086 floating point arithmetic routines.

If the calculator is used from within the BIOS then slightly different Microscreen setup routines are used (for example, the SEND legend is displayed on the MicroScreen, and the second LED on the membrane pad is illuminated). Also, an extra feature is incorporated whereby if Membrane Key 2 is pressed then the current contents of the Microscreen, starting from the top-left and continuing right until anything but a valid number or a decimal point is encountered gets sent (a character at a time) into the MS-DOS input queue.

CALCULATOR

SYSTEMS INTEREST

Each time a key is pressed, the BIOS consults the CALCULATOR ON flag to see if the calculator is switched on. If so, then the Calculator bit-map (see the Keyboard Tables section for more information) is checked against the number of the key held down. If there is a "1" in the bit map, then the key is designated to be calculator specific so the Auto repeat routine is bypassed and the raw downcode is sent to the calculator. It sould be noted that the calculator specific keys can be defined in the table as anything, but the calculator will still interpret the down-code and produce the correct action, the only key that must not be re-defined is the CALC key, which has to produce the special "I am the CALC key" code.

SOUND GENERATOR

INTRODUCTION

The Apricot incorporates the SN 76489 sound generator chip, which has four sound channels. Two of these are used by the bell and the keyboard click, leaving two for the user. The BIOS controls the software aspect of the sound, including both initialization and execution. The Keyboard click is achieved through the white-noise channel of the chip, while the bell uses one of the square wave channels.

APPLICATIONS INTEREST

The Sound Generator is accessed from Port 50H, the program below will play a short four-note tune:

```
10 DATA 9,128,15,149,168,7,182,192,5,215          'C
20 DATA 9,136,12,149,164,6,182,195,4,215          'D
30 DATA 9,128,10,149,160,5,182,197,3,215          'E
40 DATA 9,136,07,149,172,3,182,200,2,215          'F
50 DATA 4,159,191,223,255,255,255,255,255,255     'no note
60 FOR K=1 TO 5
70 FOR L=1 TO 10:READ A
80 OUT &H50,A:NEXT
90 FOR L=0 TO 200:NEXT   'Delay
100 NEXT
```

AUXILIARY DEVICE/SERIAL PORT

INTRODUCTION

The Apricot has, in addition to a parallel port, a serial RS-232 data port which can be connected to a wide variety of external hardware add-ons. The hardware that controls the port is the Z80A SIO (Serial Input/Output) chip. The Apricot is capable of transmitting at a rate of up to 19200 BAUD (Bits per second).

APPLICATIONS INTEREST

Most communication with the ports and thus with the outside world should take place through the particular applications language being used. However, the Apricot Control Device is an easy and powerful way of configuring and using the port. For more information see the section concerning the control device.

SYSTEMS INTEREST

The serial communications driver manages 1 RS-232C port on the Apricot computer. The driver provides for only asynchronous data transfer at speeds from 75 baud to 19200 baud. An interrupt driven interface is provided as well as two dynamic length queues to provide character buffering whilst the interface or application is otherwise occupied.

The serial port is located in port A of the Z80-SIO and utilises the following control/status signals

RTS – Request To Send (pin 4)
CTS – Clear To Send (pin 5)
DCD – Data Carrier Detect (pin 6)
DTR – Data Terminal Ready (pin 20)

SERIAL PORT

The Clock source for the serial port is as follows (selection is via DTR and RTS bits of channel B):

DTR, RTS

0	Output clock 1 from 8253 to RxC & output clock 2 to TxC
1	RxC and TxC driven from line clock
2	RxC and TxC driven from output clock of 8253 timer
3	RxC and TxC at 0v

Modem Control Through The AUX Device

When Modem Control is selected, the user has control over the following:

RTS
DTR
Transmit Enable
Receive Enable

And can monitor the state of

RTS
DTR
Transmit Enable
Receive Enable
DSR
CTS
DCD

GRAPHICS SYSTEMS EXTENSION SYSTEM

INTRODUCTION

The Apricot has, as standard, an 800 by 400 pixel high-resolution screen. This gives the user 320 thousand individual points which, in the high-resolution mode, can be individualy switched on or off as required.

The Graphics Systems Extension module, from Digital Research, allows the user to write programs which can use the full high-resolution screen. Programs written in high-level languages (such as Pascal and Basic) which use Graphics Systems Extension can be compiled on the Apricot and then executed.

APPLICATIONS INTEREST

Graphics Systems Extension Functions

All graphics devices were not created equal. Terminals, plotters and printers all draw lines, fill in areas and produce text differently.

All computer graphics are displayed on a coordinate system of one sort or another. Graphics Systems Extensions job is to make sure the coordinate system that one device uses matches the coordinate system used by another. For example, with GSX an applications program produces the same graphics image on an Apricot as it does on an IBM PC: the linetypes are the same, the character size the same, etc.

How To Use Graphics Systems Extension

Applications programs work with Graphics Systems Extension through a standard calling sequence. GSX translates the parameters passed by these standard calls to fit the peculiarities of each graphics device. It is this translation process that makes applications programs device-independent.

GSX SYSTEM

What is Graphics Systems Extension?

Each graphics device is mechanically and electrically different, and requires a special program to run it. This program is called a Device Driver. Graphics Systems Extension consists of a Device Driver written by ACT (called the GIOS - or Graphics Input/Output System) and a User Interface supplied by Digital Research called the GDOS (Graphics Device Operating System).

For details on how to install Graphics Systems Extension and the relevant program functions to use it, see the GSX-86 Programmer's manual.

SYSTEMS INTEREST

The Graphics Systems Extension Apricot Graphics Device driver consists of three files:

1 – GRAPHICS.EXE
2 – ASSIGN.SYS
3 – DDACRT.SYS

1. GRAPHICS.EXE

This is the file which contains the Digital Research GDOS module. When loaded, it will first allocate about 32K of user RAM for itself and the GIOS. Then it will open ASSIGN. SYS and read the Device number and the name of the file containing the GIOS device driver from it. After doing this it will open the GIOS file and load it into RAM so that it is within the 32K that GDOS has allocated. Interrupt 223 is set up to point to the GDOS command structure entry point, then the program terminates with an MS-DOS "Stay Resident" call thus moving the start of the TPA up by 32K.

2. ASSIGN.SYS

As described above, this file contains the number and name of the logical Device Driver that the Apricot uses.

When TYPEd it should display the following:

```
A>TYPE ASSIGN.SYS
01 ddacrt.sys

A>_
```

The 01 indicates that the driver is a CRT device driver, and is the primary one that is used for output.

The name of the file that contains the Device Driver is next.

3. DDACRT.SYS

This file is the ACT Graphics Input Output System. It is what enables GDOS to draw lines, plot points and fill areas. The Device Driver was written in 8086 Assembler and Pascal.

THE ACT GRAPHICS SYSTEM – DDACRT.SYS

On the Apricot, the following functions are provided that are classed as "Machine Dependent" in the Graphics Systems Extension Programmer's guide:

▷ **Line Attributes:**

a. 7 Line Styles, 1 to 7
b. 5 Line Widths, 1, 3, 5, 7, and 9 (In Pixels)

▷ **Marker Attributes:**

a. 5 Markers, 1 to 5
b. 16 Sizes, 1, 3, 5 .. 31 (In Pixels)

GSX SYSTEM

▷ **Text Attributes**

a. 1 character font
b. 4 heights 14, 28, 42, 56 (In Pixels)
c. Rotation, with the smallest angle 1 degree

▷ **Fill Attributes**

a. 4 interior styles:

0 – Hollow
1 – Solid
2 – Pattern
3 – Hatch

b. 6 style indices for Pattern and Hatch interior styles

▷ **Input Locator**

One device is available, using a cross-hair cursor:

The keyboard, using the standard cursor keys or the numeric pad. The 5 key or the Clear key acts as a toggle for varying the cursor speed.

▷ **Input Valuator**

One device is available here. The device is given an initial value which the user may increment or decrement by using the '+' or '–' keys. The default increment is 1, but this can be changed by typing a numeric key. Any non-numeric key apart from + and – will terminate the input.

▷ **Input Choice**

The user may enter a number between 0 and 99. If a single digit is entered, it should be followed by any non-numeric key to terminate the input. The 6 membrane keys constitute the second input choice device.

▷ **Input String**

Using the keyboard. Strings are terminated by typing carriage return.

Elements and facilities not implemented in the first release of the graphics system are as follows:

1. Colour
2. Inquire cell array
3. General Drawing Primitives:

    (a) Arc
    (b) Pie Slice
    (c) Circle
    (d) Graphics characters for use withprinters.

4. Escape Op-codes:

    (a) Tablet Status
    (b) Hard Copy

APPENDICES

APPENDIX A

CIRCUIT DIAGRAMS

Figure 1. SYSTEM BOARD CPU SECTION
Figure 2. SYSTEM BOARD DRAM AND CRTC LOGIC
Figure 3. 128k DRAM SECTION
Figure 4. EXPANSION SLOTS
Figure 5. KEYBOARD

The circuit diagrams listed above can be found in the wallet on the inside of the back cover of the manual binder.

APPENDIX B

BOOT PROM DIAGNOSTICS

After a power-on reset, or following a keyboard reset prior to the insertion of a system disk, the boot PROM performs a series of diagnostic checks on the system hardware. After the first 8 system diagnostic checks, the screen is initialised to display the Apricot logo and the words "**Testing**". If all the diagnostic checks pass, "**Testing**" is replaced by "**System OK**", the RAM size and the prompt for a system disk (disk symbol and flashing arrow).

If any of the diagnostic checks fail, the screen displays the RAM size, the prompt for the system disk and an Error number. The Error number indicates which diagnostic test failed, as detailed in the table on the next page. If more than one of the tests fail, the first failure detected is indicated by the error code.

If all the diagnostic checks pass and then the system is unable to boot a system disk, due to detecting an invalid/ corrupted disk or disk drive failure, the arrow stops flashing and an Error number is displayed according to the fault. These are detailed in the table below. Removal of the disk causes the disk prompt to be resumed, but leaves the error code displayed.

FAILURE TO BOOT ERROR CODES

Error Number	Error Condition	Possible Cause
02	Disk Drive Not Ready	Drive power failure, drive motor failure, drive not able to be selected, disk removed during boot.
04	CRC Error	Corrupt disk data.
06	Seek Error	Unformatted disk, bad track No. in disk ID field, disk drive fault.
07	Bad Media	Corrupt disk media block.
08	Sector Not Found	Unformatted disk, bad sector No. in disk ID field, disk drive fault.
11	Bad Read	Corrupt data field on disk.
12	Read Fault	8089 processor, floppy disk controller (FDC), disk drive fault.
99	Invalid System Disk	Invalid header on disk.

DIAGNOSTIC TEST ERROR CODES

Error Number	Test	Possible Cause
20	PROM Checksum	Boot PROM Fault
22	SIO read/write	Serial input/output controller fault.
23	CRTC read/write	CRT Controller fault.
24	Screen RAM read/write	Screen RAM fault.
25	System RAM read/write	System RAM fault.
26	PIO transfer	Parallel Port fault.
27	PIC read/write	Interrupt Controller fault.
28	FDC read/write/seek	Disk Controller, disk drive fault.
29	TMR read/write	Programmable Timer fault.
30	RS232C transfer	Z80 SIO channel A fault.
31	Keyboard initialisation	Keyboard fault or Keyboard disconnnected.
32	TMR accuracy.	Programmable Timer fault.
33	TMR/PIC interaction	Programmable Timer or Interrupt Controller fault.
34	IOP initialisation/ memory transfer	8089 processor.

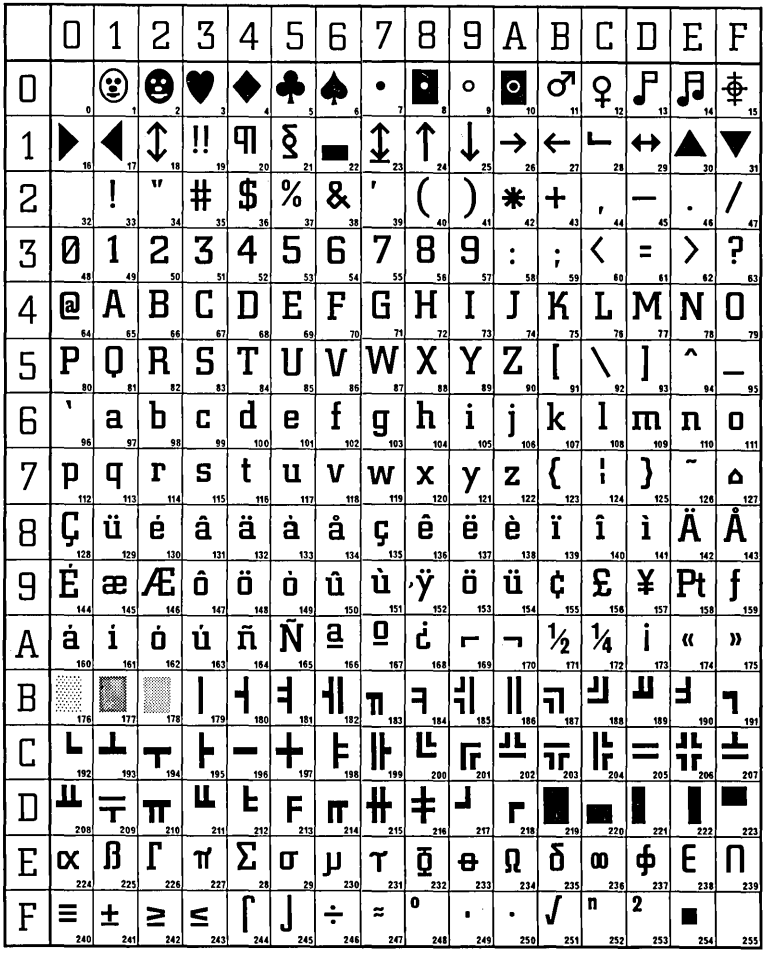
APPENDIX C

DEFAULT CHARACTER FONT

The default character font loaded by the BIOS into the System RAM (for UK use) is illustrated on the following page. This is the ASCII character set, SINTL 01 which is allocated 8 Kbytes of System memory from address location 0800H to 27FFH. Each of the 256 characters is stored in an area of memory defined as a font cell, which consists of 32 contiguous bytes in memory, organised as 16 consecutive 16-bit words. For further details of the concept of the font cell and details of the method of displaying characters on the screen, refer to the Screen Driver and CRT Control chapters.

Note: The Screen Driver interprets the first 32 ASCII codes of the character set (00H to 1FH), as control codes, unless the escape sequence ESC 8 is sent to the driver.

ASCII CHARACTER SET



APPENDIX D

HEXADECIMAL TO DECIMAL CONVERSION

INTRODUCTION

This appendix provides a simple method of converting:

(a) Any hexadecimal value from 00000H to FFFFFH to a decimal number.
(b) Any decimal number from 0 to 1,048,575 to a hexadecimal value.

This method of converting between the two different bases makes use of the look-up tables overleaf and generally requires a simple arithmetical calculation to be carried out. Two examples are shown below using arbitrary numbers to illustrate the principles involved.

Example 1. Conversion of 7FEA3H to decimal.

Step 1. Partition the hexadecimal number into the following segments

```
70000H+   ...x
0FE00H+   ...y
000A3H    ...z
7FEA3H
```

Step 2. Use look-up table 1 to find the value of x. i.e. 458,752

Step 3. Use look-up table 2 to find the value of y. i.e. 65,024

Step 4. Use look-up table 3 to find the value of z. i.e. 163

Step 5. Add the three decimal numbers together to arrive at the required result.

```
458,752+
 65,024+
     163
523,939 = 7FEA3H
```

Example 2. Converting 8,322 to hexadecimal.

Step 1. Search through the look-up tables to find the number which is identical to the decimal value; or if this is not possible (as in most cases), the value which is closest, but lower than the decimal value.

e.g. 8,192 = 2000H from look-up table 2.

Step 2. Subtract this value from the original value.

```
8,322 -
8,192 -
  130
```

Step 3. Repeat the procedure as detailed in Step 1 using the result of the subtraction process.

e.g. 130 = 82H from look-up table 3.

Step 4. Add the heaxadecimal values obtained to arrive at the required result.

```
2000H +
  82H +
2082H = 8,322 decimal
```

Table 1

Hex	Decimal
00000	0
10000	65,536
20000	131,072
30000	196,608
40000	262,144
50000	327,680
60000	393,216
70000	458,752
80000	524,288
90000	589,824
A0000	655,360
B0000	720,896
C0000	786,432
D0000	851,968
E0000	917,504
F0000	983,040

Table 2

Hex	000	100	200	300	400	500	600	700	800	900	A00	B00	C00	D00	E00	F00
0	0	256	512	768	1,024	1,280	1,536	1,792	2,048	2,304	2,560	2,816	3,072	3,328	3,584	3,840
1	4,096	4,352	4,608	4,864	5,120	5,376	5,632	5,888	6,144	6,400	6,656	6,912	7,168	7,424	7,680	7,936
2	8,192	8,448	8,704	8,960	9,216	9,472	9,728	9,984	10,240	10,496	10,752	11,008	11,264	11,520	11,776	12,032
3	12,288	12,544	12,800	13,056	13,312	13,568	13,824	14,080	14,336	14,592	14,848	15,104	15,360	15,616	15,872	16,128
4	16,384	16,640	16,896	17,152	17,408	17,664	17,920	18,176	18,432	18,688	18,944	19,200	19,456	19,712	19,968	20,224
5	20,480	20,736	20,992	21,248	21,504	21,760	22,016	22,272	22,528	22,784	23,040	23,296	23,552	23,808	24,064	24,320
6	24,576	24,832	25,088	25,344	25,600	25,856	26,112	26,368	26,624	26,880	27,136	27,392	27,648	27,904	28,160	28,416
7	28,672	28,928	29,184	29,440	29,696	29,952	30,208	30,464	30,720	30,976	31,232	31,488	31,744	32,000	32,256	32,512
8	32,768	33,024	33,280	33,536	33,792	34,048	34,304	34,560	34,816	35,072	35,328	35,584	35,840	36,096	36,352	36,608
9	36,864	37,120	37,376	37,376	37,888	38,144	38,400	38,656	38,912	39,168	39,424	39,680	39,936	40,192	40,448	40,704
A	40,960	41,216	41,472	41,728	41,984	42,240	42,496	42,752	43,008	43,264	43,520	43,776	44,032	44,288	44,544	44,800
B	45,056	45,312	45,568	45,824	46,080	46,336	46,592	46,848	47,104	47,360	47,616	47,872	48,128	48,384	48,640	48,896
C	49,152	49,408	49,664	49,920	50,176	50,432	50,688	50,944	51,200	51,456	51,712	51,968	52,224	52,480	52,736	52,992
D	53,248	53,504	53,760	54,016	54,272	54,528	54,784	55,040	55,296	55,552	55,808	56,064	56,320	56,576	56,832	57,088
E	57,344	57,600	57,856	58,112	58,368	58,624	58,880	59,136	59,392	59,648	59,904	60,160	60,416	60,672	60,928	61,184
F	61,440	61,696	61,952	62,208	62,464	62,720	62,976	63,232	63,488	63,744	64,000	64,256	64,512	64,768	65,024	65,280

Table 3

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

APPENDIX E

DESIGN CRITERIA FOR APRICOT EXPANSION BOARDS

INTRODUCTION

This appendix acts as a supplement to the information provided in the Expansion Slots chapter in the HARDWARE section, and defines the guidelines and constraints applied to the design of both the hardware and software for Expansion Boards in-house within ACT (Advanced Technology).

Any third party vendor who wishes to design Expansion Cards which are/will be compatible with ACT's existing and future products must strictly adhere to the information provided below.

The guidelines are divided into a number of different areas, which can be broadly summarized as follows:

(a) Using interrupts.
(b) Using the DMA facilities.
(c) I/O address allocation.
(d) Mechanical Constraints.
(e) Miscellaneous Criteria.

INTERRUPTS

Of the two interrupt request lines to the Interrupt Controller on the System Board, INT2 and INT3, INT2 is reserved by ACT solely for use by the ACT Winchester Controller Board. All other Expansion Boards must use INT3 only.

To account for the possibility of two boards both requiring the use of the single available interrupt request line:

(a) All boards using INT3 must contain a status register.
(b) The software device driver must operate in the manner described in the paragraphs below. The use of the status register will also become apparent.

Device Driver Specification

Installing the device driver at system boot.

One of the routines executed during initialisation must read the four vectors stored in memory specified by the interrupt number 53H, and then substitute its own vectors to point to the appropriate device driver service routine.

Run-time operation.

The first task undertaken by the service routine must be to read the status register to check that the device is the actual cause of the interrupt. If not, the device driver immediately relinquishes control to the service routine specified by the vectors read previously during initialisation.

If the interrupt is caused by its own device as indicated by the status register, the driver naturally performs the appropriate service routine. At the end of the routine, the device driver must relinquish control to the routine specified by the vectors read during initialisation.

This effect, where multiple device drivers are installed all using the same interrupt line, is similar to the technique of "daisy chaining" interrupt lines and acts as a logical extension to the MS-DOS 2.0 installable device driver philosophy.

Notes

1. At boot-time the BIOS assigns in effect a null device driver to interrupt number 53H, which performs two functions:

(a) Supplies the end of interrupt command to the Interrupt Controller.
(b) Returns control back to the program point of interruption.

The "null device driver" is always the last "driver" to be serviced, since it will always be the first one installed (i.e. the end of the "daisy chain").

2. Since the last loaded driver will always be the first device serviced following an interrupt (i.e. the beginning of the "daisy chain"), the order of loading multiple device drivers automatically assigns an order of priority. Time critical device drivers therefore, should always be loaded last.

3. The maximum time allowed for each interrupt service routine should generally not exceed 10ms.

DMA TRANSFERS

Channel 1 of the I/O processor (IOP) is permanently assigned to the Floppy Disk Controller on the System Board, so the input lines DMA 1 and EXT 1 are not available for use by any Expansion Board. All DMA transfers from/to the Expansion Boards must make use of channel 2 of the IOP (control lines DMA 2 and EXT 2).

The start of the Channel Control Block for channel 2 is located at 508H. On **no** account should:

(a) The priority bit in the CCW byte be updated to assign a greater priority to channel 2 than channel 1.
(b) The Channel Control Block for channel 1 be altered.

I/O ADDRESS ALLOCATION

The available I/O address range for the Expansion Boards is split into a number of categories according to the type of board. This approach is necessary to avoid the possibility of the Expansion Slots being filled with Expansion Boards utilising the same I/O addresses.

The following table details the I/O addressses assigned to thevarious categories of Expansion Cards. (The values of the port addresses are in hexadecimal).

Expansion Board Category	I/O Port Address☆	I/O Port Address★
Colour/Graphics Boards	80 to 8F	100 to 10F
IEEE 488 Controllers	90 to 9F	110 to 11F
Local Area Networks	A0 to AF	120 to 12F
Gateways	B0 to BF	130 to 13F
Miscellaneous Communications	C0 to CF	140 to 14F
ACT Reserved	D0 to DF	150 to 15F
Winchester Boards	E0 to EF	1E0 to 1FF
Modems	F0 to F7	1C0 to 1DF
Undefined	–	160 to 17F
Undefined	–	180 to 19F
Undefined	–	1A0 to 1BF

☆ 1 processor wait state is automatically inserted to any I/O read or write by accessing addresses in the column below.

* Processor wait states are only inserted to I/O read or writes by accessing the addresses in the column below if the IRDY input is utilised.

MECHANICAL CONSTRAINTS

The Expansion Slots chapter in the HARDWARE section illustrates the mechanical details (board size, connector details, maximum component height available for each slot, etc.), for a standard design of Expansion Board. Different designs are possible, utilising the available space in a different manner as long as the new design does not prevent the insertion of a standard design of Expansion Board into the other slot.

In general, all Expansion Boards should be designed to the tighter tolerances specified for the slot Expansion 1 to allow the integral ACT Modem, when fitted, to monopolize the Expansion 2 slot.

MISCELLANEOUS CRITERIA

1. One of the 8086/8089 instructions has a specific purpose on the System Board and is not available for use in any other software. This is the bus lock instruction **LOCK**, which is used by the BIOS to access interrupt vectors, generated by the Z80 SIO.

SUMMARY

Hardware Considerations

- ▷ Interrupt request line, INT2 is reserved for the ACT Winchester Controller Board.
- ▷ Every Expansion Board which uses the interrupt request line must also provide a status register.
- ▷ DMA request line 1 is not available as this is permanently assigned to the on-board Floppy Disk Controller.
- ▷ All Expansion Boards must be designed to fit into the Expansion 1 slot to allow the Apricot integral Modem to utilise Expansion 2.
- ▷ Address decoding must assign the device to the categories defined above.

Software Considerations

- ▷ All device drivers using interrupts, must follow the specification detailed above, to enable more than one device driver to use the same interrupt line.
- ▷ The time taken for each device driver interrupt service routine should not exceed 10ms.
- ▷ DMA Transfers can only use channel 2 of the I/O processor.
- ▷ The bus lock instruction, LOCK cannot be used.