CONTENTS

1 INTRODUCTION

1.1 The 4-70

The 4-70 is a member of English Electric Computers' range of compatible processors. Used in conjunction with System 4 peripheral devices and standard 4-70 software, it is capable of forming the heart of a wide range of general purpose computer system configurations.

1.2 Role of the 4-70 Processor in a Computer System

The functions of a general purpose computer system are to accept information presented on various types of peripheral media, to store it, to process it, and to output the results of such processing onto peripheral media.

A 4-70 computer system comprises main storage, to which is connected a Central Control Unit, associated input-output channels and an operator console, with peripheral devices connected to the channels. All but the peripheral devices, which provide the means of handling peripheral media, and which may include backing stores, come under the heading of '4-70 processors'.

The information held in the main store includes instructions, which are executed in sequence by the Central Control Unit causing it to perform various operations on stored items of information, and enabling them to control the operations of input-output channels and peripheral devices. Complete sequences of instructions are called 'programs'.

The two main activities of the processor are :-

- Transferring information between main store and peripheral devices under control of the Central Control Unit.

- The execution of program instructions by the Central Control Unit.

These activities go on in parallel.

The operator console provides the means for human operators to oversee the activities of the computer system as well as providing facilities for initiating its operation.

1.3 The Supervisor Program

The operations of any 4-70 computer system are controlled by a specially written program called the 'Supervisor'. As well as controlling the execution of all other programs (known as 'object programs') the Supervisor incorporates features enabling the human operator to oversee the system. Two features of the 4-70 processor are exploited by the Supervisor program :

1.3 (continued)

(i) the ability to operate in a 'privileged' environment in which the Supervisor is able to execute special instructions and use certain facilities, denied to object programs.

(ii) The 'interrupt' feature, which enables the Central Control Unit, in response to various conditions arising within and outside itself, to automatically break off the normal execution of an object program and initiate a predetermined sequence of instructions (the Supervisor) in such a way that the interrupted program can subsequently be resumed where it left off.

The Central Control Unit can only execute one program at a time, hence the Supervisor is only active when no object program is being executed.

1.4 Features of the 4-70 Processor

1.4.1 General

The logical entities which make up the 4-70 processor are :-

Main Store
Central Control Unit
Input-Output Channels.

1.4.2 Main Store

The main store, to which the Central Control Unit and the channels are connected, is fully described in Section 3 of this Specification.

Essentially, the main store provides the ability to store individually-addressed 8-bit units of information (referred to as 'bytes'). Bytes are held in store locations numbered consecutively from 0 upwards; the number of a location is its 'absolute address'. Store sizes range from 65,536 (2^{16}) bytes up to 1,048,576 (2^{20}) bytes, in units which are multiples of 65,536 bytes.

The number of store accesses that may be obtained concurrently is a function of the store size. See Section 3.2.1.2.

Each byte in the main store has a parity bit which is invisible to any program. Parity checking is automatic, and the failure of a check generates either a 'Machine-Check' condition or a channel fault.

A store protection system is provided which can be used to prevent programs from writing to areas of main store. This is described in Section 3.

The main store cycle time is 900 nsec for one word.

1.4.3 Central Control Unit Features

1.4.3.1 Registers and Scratchpad

The Central Control Unit is controlled by a number of registers. Some of these consist of locations in a small fast store called the Scratchpad. This store is also used to hold instruction specified operands which are required for immediate processing. Such operands are addressed relatively, see Section 1.4.3.2.2. Direct addressing of the Scratchpad is performed by certain privileged instructions.

1.4.3.2 Arithmetic Unit

1.4.3.2.1 General

The Arithmetic Unit is the part of the Central Control Unit generally responsible for the sequencing and execution of instructions. It includes a 'Sequence Control Counter' register which is used for sequencing instructions. This register contains the main store address of the next instruction to be executed. Instruction sequencing and execution involves the following cycle of activities :-

- fetching the next instruction from the designated store location.
- interpreting this instruction, fetching the operands required from specified registers and store locations, operating on them and storing results in the correct locations.
- updating the sequence control counter, either by adding the length of the instruction or by substituting an address generated by the instruction.

The instruction set used by object programs is the same as that used by IBM 360 processors and the 4-50 processor. It is defined in PS 4.6.3. Instruction formats, and the formats in which their data must be represented, are described in Section 1.5.

1.4.3.2.2 Processor States

Instructions specify main store addresses and also refer to sixteen 'general purpose registers' of 32 bits each, and four 'floating point registers' of 64 bits each. The Central

1.4.3.2.2 (continued)

Control Unit can operate in four 'Processor States', each of which has its own set of general purpose registers (and certain other registers). However, all four states use the same floating-point registers. These registers are held in the Scratchpad. Switching between processor states is performed when interruption occurs and also at the explicit request of the Supervisor. Object programs use one processor state all the time and have no power to switch states.

The Processor State mechanism is fully described in Section 2.7

1.4.3.2.3 Interruption

The rules for sequencing instructions described in Section 1.4.3.2.1 above can be overridden by the Arithmetic Unit when it is necessary to respond to certain conditions ('interrupt conditions') generated inside or outside the Central Control Unit. On such occasions a change of Processor State, accompanied by a change of the contents of the Sequence Control Counter and other registers, occurs. In the new state, the Supervisor processes the condition before switching the Central Control Unit back to its previous state, thus setting the control registers to their original values and permitting execution of the interrupted program to proceed without the program necessarily being aware of the occurrence of the interruption.

Full details of the Interrupt system are given in Section 5.

1.4.3.2.4 Privileged Mode

At the same time that a change of Processor State occurs, the Arithmetic Unit can be made to operate in a special 'privileged' mode (or to revert from this mode to the 'non-privileged' mode). In the privileged mode only, certain privileged instructions can be executed, which allow the Supervisor to exercise control over the system. For instance, there are privileged instructions allowing a change of Processor State to be forced, or giving the Supervisor access to the Scratchpad. All instructions for controlling input-output operations are privileged and can therefore only be executed by the Supervisor.

Non-privileged instructions can be executed either by the Supervisor of by object programs. They include a 'Supervisor Call' instruction, allowing an object program deliberately to generate an interrupt when it desires the Supervisor to perform some privileged action on its behalf.

1.4.3.2.5 Not used.

1.4.3.2.6 Virtual Addressing

So that programs may be dynamically relocated, i.e. moved to other locations in the main store at arbitrary times after their execution has started, two mechanisms are provided enabling programs to specify addresses which differ from the absolute addresses of the stored items referred to, automatically converting the specified addresses to the required absolute values.

1.4.3.2.6 (continued)

The first operates by adding to all program-generated addresses, excepting addresses used in input-output operations, a 'relocation base address' - this allows programs to specify addresses which are uniformly displaced by any desired amount from the corresponding absolute values - and the second is the 'paging' mechanism described in Section 1.4.8.2 below, and in Section 19.

Program-generated addresses of either type, which are not identical to the absolute addresses of the information referenced, are called 'virtual addresses'.

1.4.4 Store Access Unit

The Store Access Unit connects the Central Control Unit and the Channel Control Units to main store. It controls the priority of access and selects the required store module. The Channel Control Units are connected to a multiplexing unit called the 'Store Access Multiplexor'. The connections, of which there are six, are called 'Store Access Ways'. Further information is given in Section 3.2.1.2.

1.4.5 Input-Output Control Facilities

1.4.5.1 General

Information is transferred between the main store and peripheral devices by input-output channels. Such peripheral transfers can take place on all channels concurrently and in parallel with the operations of the Central Control Unit. Each channel is individually addressable and can also generate a separate interrupt condition in the Central Control Unit (the priority of these interrupts influences the choice of priorities for the access ways through which these channels access the main store).

Each channel includes one or more 'trunks', each of which terminates in a System 4 Standard Interface (PS 4.7.1); to each trunk may be attached one peripheral device control unit and its associated devices. Several devices may thus be attached to one channel, all being individually addressable.

The control functions of channels are performed by units called 'Channel Control Units'. These units perform the logic functions necessary for the operation of a channel and function independently of the Central Control Unit, except when it is required to initiate a transfer or to process an interrupt condition. Communication between a control unit and the rest of the processor is through the Store Access Unit's Access Ways. These carry all data transfers as well as interrupt and other control lines.

There are three different types of Channel Control Unit available for inclusion in a processor's input-output system. Each controls a different type of channel and covers a different performance range. Further details follow in the next sections.

1.4.5.2 Not used.

1.4.5.3 Classification of Channels

1.4.5.3.1 Selector Channels

A selector channel can control a single transfer between itself and one of the 256 peripheral devices which may be connected to it. Connections are made through device control units of which there may be one for each trunk available with the channel.

Selector channels available on the 4-70 are of two different types. One is used for very high speed transfers and is controlled by the 'Single Channel Control Unit'. The other is used for medium speed transfers and up to 8 of them may be controlled by a single 'Multichannel Control Unit'.

The properties of the two types are as follows :-

Single Channels - the channel controlled by a Single Channel Control Unit may have up to four trunks. It is capable of sustaining transfers on any one of these at rates of up to 1 Mb.

Multichannels - multichannels are controlled in groups of from 1 to 8 channels, by a single Multichannel Control Unit. The combined throughput rate of the channels is 660 Kb. Each has a single trunk.

Further information concerning the capabilities of these channels is given in Section 6 and in Appendix C.

1.4.5.3.2 Multiplexor Channels

A multiplexor channel allows all the devices connected to it to transfer information concurrently. It may have up to 16 trunks. The maximum number of devices on one channel is 256, these may all be connected to one trunk or distributed over several trunks. The control functions of a multiplexor channel are performed by a unit called the 'Multiplexor Channel Control Unit'.

Further information concerning the capabilities of this channel is given in Section 6 and in Appendix C.

1.4.5.3.3 Channel Characteristics

Further details are given in Section 6.

1.4.5.4 Channel Configurations

The units available for inclusion in a 4-70's input-output system may be used to form a large number of configurations, each of which differs in its throughput and control capabilities.

1.4.5.4 (continued)

The factors which limit the power of a system are described in Section 1.6 and in Section 6. The main limits are as follows:-

- maximum number of channels	16
- maximum number of channel control units	6
- maximum number of trunks	56
- maximum overall throughput rate	4Mb

1.4.6 Timer

The Central Control Unit is provided with a clock, this is used by the Supervisor for measuring the time which elapses between events and for generating an interrupt condition after a specified time interval. Details are given in Section 2.4.11.

1.4.7 Control Facilities

1.4.7.1. Operator Control

A control desk and console are provided for operator control. The functions of the console are described in Section 10, and the method of initiating the processor's operation in Section 7.

1.4.7.2 Engineer's Control Panel

Provision is made for the control of the processor from an engineer's control panel. Its facilities are described in Engineering Specification 642 4070 0011 R08, Section 5.

1.4.8 Direct Control

Direct Control facilities for inter-processor communication, as described in PS 4.6.10, are available as an option.

1.5 Instruction and Data Formats

1.5.1 General

The non-privileged instructions and data formats used by 4-70 are fully compatible with those of System 360 and the 4-50.

The privileged and non-privileged instruction sets are described in Appendix D and in PS 4.6.3 respectively.

1.5.2 Bit-numbering Convention

Throughout this document the generally accepted convention is followed of numbering the binary digits in any field, starting from 0 at the leftmost (most significant) end of the field.

1.5.3 Instruction Formats

The instruction set includes 144 different instructions. The formats of these instructions are as described below.

Instructions are 2, 4 and 6 bytes in length. The length is related to the number of main store addresses explicitly specified by the instruction, respectively zero, one or two.

1.5.3 (continued)

The leftmost byte of each instruction has an even address, i.e. each instruction starts on a half-word boundary.

There are five basic instruction formats, described as follows :-

(2-byte) RR (Register-to-Register operation)
(4-byte) RX (Register-to-Indexed store operation)
RS (Register-to-Store operation)
SI (Store-and-Immediate-operand operation)
(6-byte) SS (Store-to-Store operation).

These formats are shown in Diagram 1.

The abbreviations in Diagram 1 have the following significance (suffices designate operand numbers) :-

Op Code — Specifies the length, format and function of the instruction.

R_1,R_2,R_3 — Specify general-purpose or floating-point registers containing operands.

X_2 — Specifies a general-purpose register being used as an index register.

B_1,B_2 — Specify general-purpose registers containing base addresses.

D_1,D_2 — Specify 12-bit positive displacements which when added to the associated base addresses (and possibly index quantities) generate 24-bit main store (operand or instruction) addresses.

I_2 — Specifies an 8-bit 'immediate operand'.

L_1,L_2 — Define the lengths in bytes of the operand fields in 'decimal' instructions.

L — Defines the length of the operand field in a 'logical' instruction.

1.5.3 (continued)

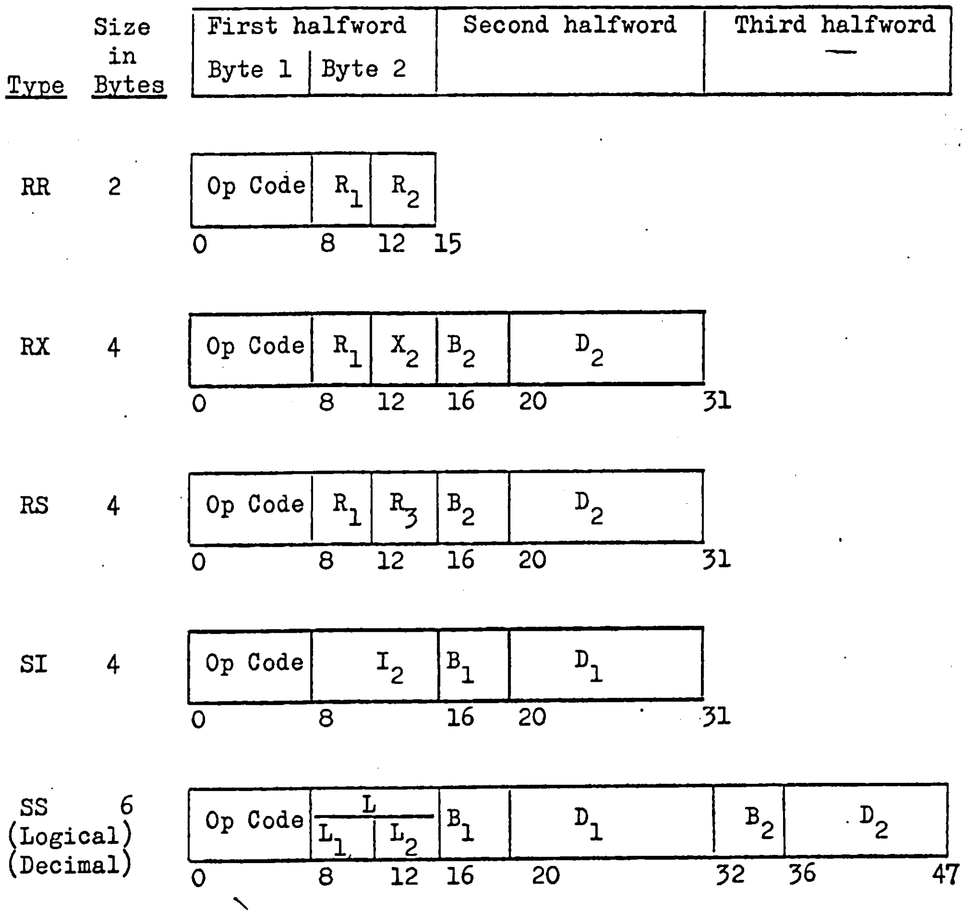


DIAGRAM 1 - INSTRUCTION CODE FORMATS

1.5.4 Data Formats

The data codes used by input and output equipment are described in PS 4.6.2.

Any binary pattern of 0s and 1s may be stored in the main store, general-purpose or floating-point registers. The interpretation of any binary information depends on the instructions which operate on it. Some instructions check the formats of the operands and generate interrupts in cases of error, to reduce the risk of undetected program errors. Data may be represented in the following forms, illustrated in Diagram 2 :-

Fixed point integers : integers are represented in binary. The '2s complement' convention is used for negative numbers, the leading bit representing the sign. The imaginary binary point

1.5.4 (continued)

is understood to follow the rightmost bit. Numbers are held in halfword, word and double-word form: in the latter case, the leftmost bit of the second word represents a digit and not a sign.

Floating-point binary: numbers are held in 'short' (single word) or 'long (double-word) form. Only the lengths of the Fractions differ. Each number consists of three separate parts. Sign (1 bit), Characteristic (7 bits), and fraction (24 or 56 bits). The sign convention is 'sign and modulus'; the sign bit is 1 for negative numbers. The Fraction is always positive, with binary point understood to lie to the left of its leftmost bit. The Characteristic represents a power of 16 in the range - 64 to + 63, recorded in excess 64 binary form.

It is possible to have numbers with zero Fraction, non-zero Characteristic and negative sign, which are acceptable as operands.

A non-zero floating-point number is said to be normalised if the first four bits of the fraction are not all zero.

Numbers in the range 10^{-78} to 10^{+75} can be represented.

Packed Decimal: decimal numbers are held two decimal digits to a byte, the rightmost four bits of the rightmost byte containing the sign. Decimal instruction can handle fields of up to 31 digits plus sign.

Zoned decimal: numbers are held in bytes, the rightmost four bits of each byte holding a decimal digit and the leftmost four bits a 'zone code'. The zone area of the rightmost byte contains the sign. Fields of up to 16 digits may be operated on.

Fixed-length logical: binary quantities not having numerical significance, or treated as unsigned numbers, are operated on in bytes, words and double-words.

Variable length logical: data, normally consisting of alphanumeric characters, can be addressed in fields of up to 256 bytes.

Fixed-Point Binary

Fixed-point halfword

Sign	Integer
0	1 15

Fixed-point word

Sign	Integer
0	1 31

Fixed-point double-word

Sign	Int—	eger
0	1	63

Floating-Point Binary

Short floating-point number

Sign	Characteristic	Fraction
0	1	8 31

Long floating-point number

Sign	Characteristic	Frac-	tion
0	1	8	63

Packed Decimal

Digit	Digit	
0	4	8

Digit	Digit	Digit	Digit	Digit	Digit	Digit	Sign

Maximum - 31 digits and sign (16 bytes)

Zoned Decimal

Zone	Digit	Zone	Digit	
0	4	8	12	16

	Digit	Zone	Digit	Sign	Digit

Maximum - 16 digits and sign (16 bytes)

Fixed Length Logical

Logical Data
0 31

Variable Length Logical

Character	Character	
0	8	16

	Character	Character

Maximum - 256 characters (bytes)

DIAGRAM 2 - DATA STRUCTURE

Issue	Date	Auth'y	Log	Title	Document No.
4	16.2.68	MRW	743	CENTRAL PROCESSOR MODELS 4-70/75	PS 4.10.70
					Page 1.11

1.5.5 Sign and zone codes

Most operations on packed decimal operands regard the binary combinations 1010, 1100, 1110 and 1111 as representing 'plus, and 1011 and 1101 as representing 'minus'. Other combinations in the sign field are usually regarded as erroneous.

A feature of the instruction code is that the Arithmetic Unit is capable of operating in 'EBCDIC' or 'ISO' mode. In practice this means that those instructions which generate sign or zone codes (EDIT, EDIT AND MASK, and UNPACK do the latter) generate patterns appropriate to the mode, as follows:-

	EBCDIC	ISO
Plus	1100	1010
Minus	1101	1011
Zone	1111	0101

1.6 Catalogue Numbers of 4-70 and 4-75 Items

1.6.1 The 4-70

1.6.1.1 Basic Processor Configuration

4870. The Central Control Unit; performs the instruction execution (including floating-point) and the control functions, of a 4-70 processor.

This catalogue number covers the cubicle steelwork, platters, interplatter wiring and power supplies cabling. The following items must be specified in conjunction with 4870:-

3346. Power Supply. On configurations with more than three Channel Control Units Power Supply Expansion 3347 will be required.

4871. Channel Control Unit Cubicle (Specified in Section 1.6.1.3)

4872. Engineers Maintenance Panel (attached to 4870).

4990. Mains Sequencer Unit.

1.6.1.1.1 The Central Control Unit Configuration

4070. This catalogue number describes the basic configuration and covers all the items covered in Section 1.6.1.1.

The Central Control Unit has provision for the addition of the following items:-

A timer.

A paging modification unit.

These units are available as follows:-

4070/6 Timer: this option is fitted in the 4871 cubicle. It is required by the standard software.

4870/1. Paging Facility Modification Kit: this kit enables the Central Control Unit 4870 to be converted to the Central Control Unit of a 4-75 processor. i.e. to item 4875.

The complete specification of the item 4870 requires that the amount of store reserved for the use of multiplexor channels is specified using the following catalogue items:-

4870/2	Store Reservation of 1,024 bytes.
4870/3	Store Reservation of 2,048 bytes.
4870/4	Store Reservation of 4,096 bytes.
4870/5	Store Reservation of 8,192 bytes.

The Central Control Unit has provision for the attachment of the following:

Up to 1Mb of Main Store.
Up to 6 Direct Control Lines.
Up to 6 Channel Control Units.
Up to (16-n) channels where n = number of direct control lines fitted.
Up to 56 Trunks.
Up to 506 devices on the multiplexor channels.
One Extension Channel Control Cubicle.

These units are specified in the succeeding sections.

1.6.1.2 Main Store

Main Store is available in sizes ranging from 65,536 bytes in increments of 65,536 bytes to 1,048,576 bytes. To provide this storage two types of store module are available. One provides 262 Kb, the other is expansible from 65 Kb bytes to * 262 Kb. The two types are as follows:- *

4170 Expandable Store Unit: provides 65Kb of store and provision for the attachment of three expansion units. The item includes cubicle steelwork (less swingframes), interplatter wiring and cabling to power supplies. It should * be noted that the expansion units simply increase the size of 4170 and do not affect simultaneity of access.

4170/1 1st Expansion Unit: provides 65Kb of store and when added to item 4170 increases the capacity to 131Kb. The item includes platters, cabling to 4170, link for 4870 and swing frames for mounting in 4170.

4170/2 2nd Expansion Unit: provides 65Kb of store and when added to 4170 plus 4170/1 the capacity becomes 196Kb. The item is similar to 4170/1 except for difference in the swing frames.

4170/3 3rd Expansion Unit: provides 65Kb of store and when added to 4170 plus 4170/1 plus 4170/2 the capacity becomes·262 Kb. The item comprises 4 platters, with boards, to fit into the dummy position in 4170/1 and /2.

4171 Store Unit: provides 262Kb of store and is the sum of 4170 and options /1, /2 and /3. Up to four of these units may be attached to Item 4870.

A single main store power supply unit 3345 will service stores of less than ½Mb. Larger stores require two of these units.

1.6.1.3 Input Output System

1.6.1.3.1 Introduction

The input/output system is constructed by fitting channel control units and trunks into the appropriate cubicles.

To specify a system it is necessary to state the required number of each unit and the interrelationship between the different units. The rules governing the incorporation of input/output units in a processor configuration are given in Section 6.

1.6.1.3.2 Cubicles

The cubicle facilities are as follows:-

4871. Channel Control Unit Cubicle: accommodates up to 4 channel control units occupying positions 0, 1, 2 and 3, it also provides facilities for up to 4 trunk platters. The item covers cubicle steelwork with fitted frames but excluding swinging frames. Interplatter wiring to accommodate all 4 channel control units is included as standard together with cabling to power supplies

4873 Channel Control: accommodates an additional two channel control units in positions 4 and 5, and provides facilities for 3 trunk platters. The item covers the cubicle steelwork less swingframes interplatter wiring and connection to 4871.

1.6.1.3.3 Channel Control Units

The Channel Control Units are specified as follows:-

4270 Multiplexor Channel Control Unit: this unit comprises 3 platters mounted on a L.H. swinging frame for fitting to the 4871 cubicle together with the 12 additional boards to be plugged into the 4871 cubicle. One or two units may be fitted in positions 0 or 2 of the central processor, and a Unit may be fitted in position 4 of optional cubicle 4873. Up to 16 trunks may be added one at a time by units 4273. Storage must be reserved to accommodate the full range of device addresses made possible by the connected device control units.

4271 Single Channel Control Unit: this unit is mechanically similar to 4270 and may be fitted in positions 0, 2 or 4 of the cubicles. 4 Trunks may be added by trunk units 4273.

4271/1 Single Channel Control Unit: this unit is similar to 4271 but is mounted on a R.H. swinging frame for fitting in positions 1, 3 or 5 of the cubicles.

4272 Multichannel Control Unit: this unit is mechanically similar to 4271 and may be fitted in positions 0, 2 or 4 of the cubicles. Up to 8 channels with 1 trunk/channel may be added by trunk units 4273.

4272/1 Multichannel Control Unit: this unit is similar to 4272 but is mounted on a R.H. swinging frame for fitting in positions 1, 3 or 5 of the cubicles.

1.6.1.3.4 Trunks

Trunks positions are allocated to channel control units in groups of 4 in such a way that the trunks (from one to four) present in a group must all be associated with the same channel control unit. Physically, trunk groups are arranged two to a platter, with the trunk positions provided by items 4871 and 4873 numbered from 0 to 55. The four trunks in each group are consecutively numbered, the number of the first trunk of the first group on a platter is a multiple of 8, and that of the first trunk of the second group is 4 greater. In all cases it is essential to state the numbers of trunks associated with each channel control unit in a processor's configuration.

Trunks are catalogued as follows:-

4273 Trunk Unit: provides one standard interface connection for a channel control unit. Trunks are arranged in groups of up to 4, all of which must be associated with the same channel control unit. 4273 will serve all trunk numbers from 0 to 55 with the exception of the first trunk of any group

This unit may only be specified as an addition to a group that contains a 'first' trunk selected by 4273/1, /2, or /3 as appropriate.

4273/1 Trunk Unit: provides the first trunk of the second group on a trunk platter. The item is used for trunks occupying the following positions:- 4, 12, 20, 28, 36, 44, 52. This item may only be specified when the appropriate 'first' trunk, item 4273/2 or /3 as appropriate, has been previously specified.

4273/2 Trunk Unit: provides the first trunk of the first group on certain trunk platters. The item is used for trunks occupying the following positions:- 0,8,32,40 and 48.

4273/3 Trunk Unit: provides the first trunk of the first group on certain trunk platters. The item is used for trunks occupying the following positions : 16 and 24.

1.6.1.4 Direct Control

A 4-70 processor equipped with System 4 direct control links may be connected to up to 6 other processors equipped with similar links. The links are provided by the following item:-

4070/2 System 4 Direct Control Connection: provides two direct control links which to function must be connected to similar links attached to other processors. The item may be added to any of the three available positions of item 4870.

1.6.1.5 Standard Control Equipment

The standard control equipment consists of a 4-70 console mounted on a control desk(item 4911) fitted with a right hand wing (item 4912). The right hand wing accommodates the standard typewriter 4741 and its control unit 4351. A full description of the control equipment other than the console is given in the following documents - PS 4.8.1, PS 4.8.2, PS 4.8.3. The console is specified as follows:-

4918. Operating Controls for the 4-70: provides the basic controls for the 4-70 and is mounted on item 4911. The item is logically specified in Section 10 of this specification. *

1.6.2 The 4-75

A 4-75 differs from a 4-70 only in the inclusion of the paging option in the Central Control Unit. Thus, with the exception of the basic processor configuration, all the catalogue items of the 4-70 are equally applicable to the 4-75. It should be noted however, that special configurations are required for multi-access computers.

The 4-75 items are as follows:-

4875. The Central Control Unit: this is a modified version of the 4-70 item 4870 and provides a paging facility. The catalogue number allows the 4075 to be specified without reference to 4-70 numbers.

4075 The catalogue number describes the basic 4-75 configuration which includes the following items:-

4990 - Mains sequencer unit.

4875 - Central Processor Unit.

3346 - C.P.U. Power Supply Unit.

4872 - Engineers Maintenance panel (attached to 4875).

4871 - Channel Control Unit cubicle.

Main Storage, Input/Output and Operating facilities are as described for the 4-70 configuration.

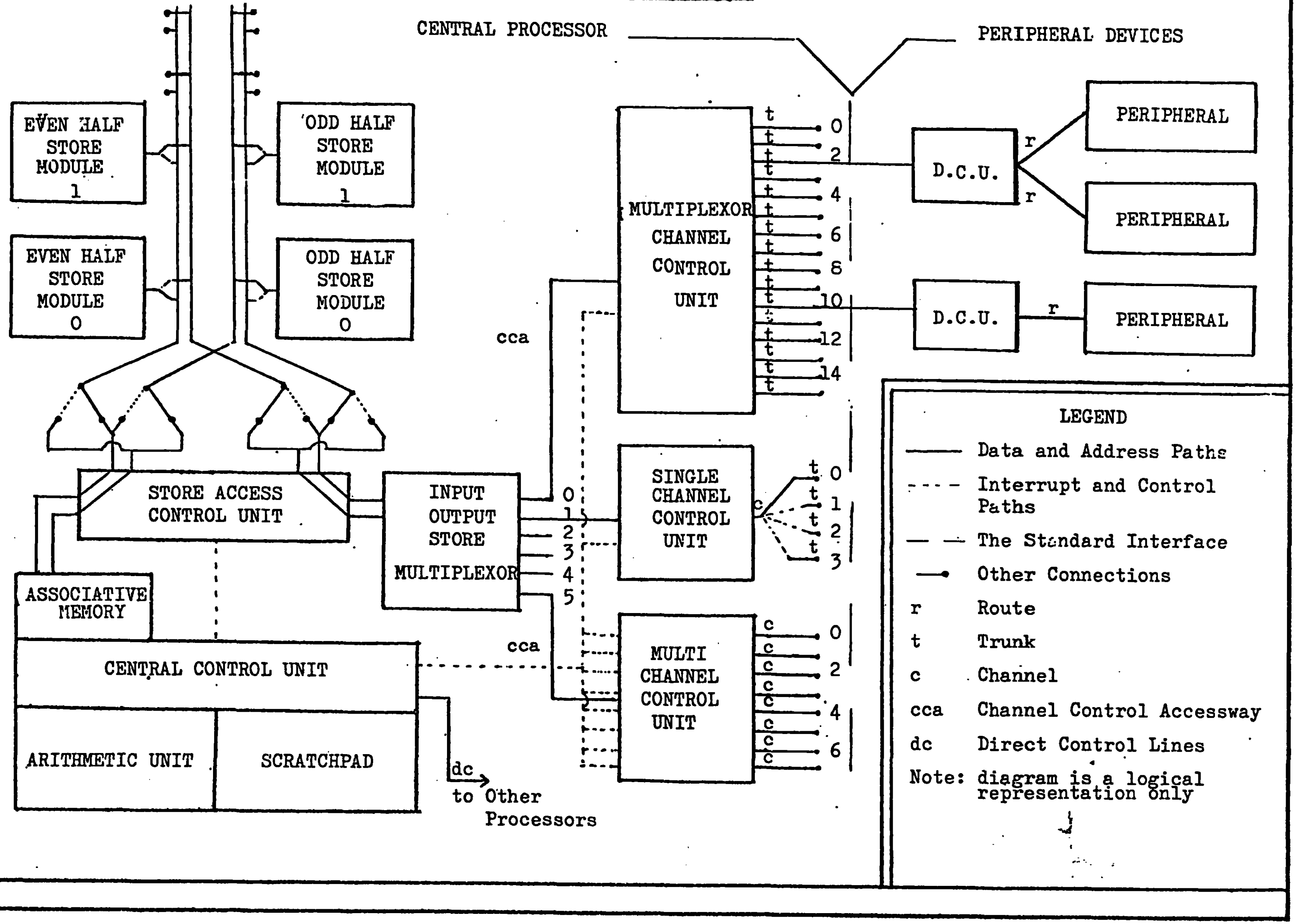
Issue	Date	Auth'y	Log
4	16.2.68	MRW	743

Title
CENTRAL PROCESSOR MODELS 4-70/75

Document No. PS 4.10.70

Page 1.18

THE SYSTEM 4-70 COMPUTER



LEGEND

———	Data and Address Paths
- - - -	Interrupt and Control Paths
— —	The Standard Interface
—●	Other Connections
r	Route
t	Trunk
c	Channel
cca	Channel Control Accessway
dc	Direct Control Lines

Note: diagram is a logical representation only

English Electric Computers

PRODUCT PLANNING SPECIFICATION

2. THE CENTRAL CONTROL UNIT REGISTERS

2.1 Introduction

In this section are described the registers used in controlling the operations of the Central Control Unit.

These registers are of three types :-

(i) Registers addressable by non-privileged programs, e.g. general-purpose and floating-point registers.

(ii) 'Control' registers, normally inaccessible to programs, which may be altered dynamically in the course of program execution or otherwise used in the control of the Arithmetic Unit.

(iii) 'Status' registers, which are used automatically to record and/or reload the contents of various control registers in certain circumstances.

These categories are not mutually exclusive, as certain registers belong to more than one.

Several of the registers described are implemented in the Scratchpad, and thus are accessible to the Supervisor. The layout of the Scratchpad is described in Section 4.

Although several of the registers described are affected considerably by interrupts, this section does not attempt to describe the interrupt system fully; this is done in Section 5.

2.2 Processor States

2.2.1 Introduction

At any time the Arithmetic Unit must be in one of four 'processor states'. Each processor state has its own set of general-purpose registers, and certain other registers; the fact that the Arithmetic Unit is functioning in a particular processor state means that instructions referring to general-purpose registers, and control functions depending on those other registers, use the set pertaining to that state. The processor states are referred to as P1, P2, P3 and P4.

The registers involved, apart from the general-purpose registers, are :-

Program Counter
Interrupt Status Register (ISR)
Interrupt Mask Register (IMR)

The functions of the first two of these are described in Section 2.3.2 below, and of the Interrupt Mask Register in Section 5. All these registers are implemented in the Scratchpad.

2.2.1 (continued)

There are functional differences between the processor states, for the following reasons :-

(i) interrupts always cause the Arithmetic Unit to switch to P3 or to P4, depending on the interrupt condition.

(ii) The P-counter, ISR and IMR of all four states, and certain other control registers, actually coincide with certain general-purpose registers of P3 and P4. This means that these two states have fewer general-purpose registers to use as such. One side-effect of this is that the instructions 'Edit and Mark' and 'Translate and Test' do not use general-purpose registers 1 and 2 in P3 and P4, but instead use registers 13 and 14 in P3, and 9 and 10 in P4.

2.2.2 Functions and Properties of Individual States

Each processor state is assigned a particular function, as follows :-

P1 Execution of object programs takes place in this state.

P2 Execution of many supervisor functions is carried out in this state, particularly sequences of supervisor instructions which are interruptable.

P3 This state is entered automatically following most types of interrupt. The Interrupt Flag Register, which is a control register indicating outstanding interrupt conditions, coincides with one of the general purpose registers of this state, as do the status registers and IMRs of P1, P2 and P3; most of the non-interruptable housekeeping activities of the Supervisor, including interrupt analysis, occur in this state.

P4 This state is entered automatically following interrupts associated with hardware malfunction. It is provided essentially so that even the Supervisor housekeeping functions performed in the P3 state can be interrupted to deal with these failure conditions as soon as they arise.

Using the Interrupt Mask Register, interrupts which would cause the P3 and P4 states to be interrupted and re-entered, and hence confuse the Supervisor, can be inhibited.

2.2.3 Reasons for Changes of Processor State

Changes of processor state occur either because of interrupts, which cause the Arithmetic Unit to switch to P3 or P4, depending on the interrupt condition; or because of the execution (by Supervisor) of the privileged instruction 'Program Control' which allows any desired state, specified explicitly or implicitly, to be entered.

2.3 Status Registers

2.3.1 Storage and Reloading of Control Registers

When the Arithmetic Unit changes from one processor state to another, the contents of those registers which exist separately for each state (listed in Section 2.2.1) are automatically preserved by virtue of the fact that the Arithmetic Unit starts to use another set. However certain unique control registers (e.g. the Sequence Control Counter) have to be reloaded automatically to give full effect to the change of state, and before this occurs the previous contents of some of these control registers have to be preserved automatically - these are the registers whose contents are likely to have altered since they were loaded at the previous change of processor state.

Each processor state has a pair of 32-bit Scratchpad registers, called status registers, for preserving and reloading the contents of the appropriate control registers.

2.3.2 Description of Status Registers

2.3.2.1 Program Counter

The first status register is the Program Counter, or P-Counter. The format of this is as shown :-

ILC	CC	Program Mask	Sequence Control Counter
0	2	4	8 31

The designations of the fields refer to the control registers whose contents are stored in those fields. They are not the registers themselves (ILC = Instruction Length Code; CC = Condition Code).

The updated contents of these registers, in the same format as they are recorded in the Program Counter Registers, are stored in a specified general-purpose register when the instruction 'Branch and Link' is executed.

The table below illustrates the use of the fields of the Program Counters of the old and new states upon a change of processor state :-

	'Old' field used to preserve contents of control register?	Control register reloaded with contents of 'new' field ?
ILC	Yes	No
CC	Yes	Yes
Program Mask	Yes	Yes
Sequence Control Counter	*Yes	Yes

*In this connection 'Program Control' is to be regarded as a Branch Instruction which has overwritten the Sequence Control Counter with the specified address just prior to the change of state.

2.3.2.2 Interrupt Status Register (ISR)

The format of the second status register, the ISR, is as follows :-

MSP	ISI	SP	Key	RBA	A	N	P	P KC	Call
0	1	3	4	8	20	21	22	23	24 31

(MSP = Main Store Parity failure.
ISI = Interrupt State Identifier.
SP = Scratchpad Parity failure.
A = ISO/EBCDIC Code Bit. }
N = Non-privileged mode bit. } mode bits.
P = Paging mode bit. }
RKC = Reservation Key Control (see Section D.2.10 and D.2.11).
RBA = Relocation Base Address.)

The 'Call' field is a 'live' control register. It is used during the execution of a 'Supervisor Call' instruction to record the byte specified by the instruction.

The table below illustrates the use of the fields of the ISR of the old and new states upon a change of processor state :-

	'Old' field used to preserve contents of control register?	Control Register reloaded with contents of 'new' field ?
MSP and SP	+No	No
ISI	*No	*No
Key	No	Yes
RBA	No	Yes
RKC	No	Yes
Mode Bits	No	Yes

+ See Section 2.4.8.

* The ISI field is used to record the identity of the previous state, i.e. on a change of state the contents of the ISI control register are stored in the ISI field of the ISR of the <u>new</u> state. The ISI register itself is reloaded with the identifying number of the new state.

2.3.2.3 Further remarks on State Changes

On any change of state, the following actions occur which affect control and status registers :-

(1) The contents of certain control registers are recorded in the status registers of the original state.

2.3.2.3 (continued)

(2) Certain control registers are reloaded from fields of the new state's status registers, and other sources.

(3) The previous contents of the ISI register overwrite the ISI field of the new state's ISR.

Notes :

(i) An interrupt may occur immediately after the execution of a 'Program Control' instruction. In this case, the value, recorded in the ISI field of the processor state initiated by the interrupt, indicates the state which would have been entered had no interrupt occurred, rather than the previous state in which Program Control was executed. However no change is made to the status registers of the former.

(ii) The status registers of the new processor state are unaltered except for the ISI field of the ISR.

(iii)If an interrupt occurs, or a 'Program Control' instruction is executed, which causes re-entry to the same state, note (ii) above is no longer true, since the status registers of the old and new states coincide. In this case, the identity of the last (different) state is lost since the ISI field is overwritten, and so caution must be exercised in the deliberate use of this technique.

2.4 The Function of the Control Unit Registers

2.4.1 Interrupt State Identifier (ISI)

2.4.1.1 Description

The sequence in which the processor states were interrupted before operations commenced in the current state, may be determined by examining the Interrupt State Identifier field of the Interrupt Status Register of each state. This facility permits the return to an interrupted state to be made. The return jump is made using a 'Program Control' instruction in the manner described below.

The code stored in an Interrupt State Identifier field is as follows :-

State Interrupted	Contents of ISI } field of new state
1	11
2	10
3	01
4	00

2.4.1.2 Action on Change of State

2.4.1.2.1 Automatic Interruption

During a change of state the contents of the Interrupt State Identifier register are stored into bits 1 and 2 of the Interrupt Status Register of the state being initiated. The Identifier register is then reset, with the code of the newly initiated state, by hardware.

2.4.1.2.2 Execution of a 'Program Control' Instruction

If the change of state is caused by the execution of a 'Program * Control' instruction which has its immediate operand bit 7 set to '0', the code, given in bits 5 and 6 of the immediate operand of the instruction, specifies the state to be initiated.

* Indirect Control Flag

If bit 7 is set "1" the state initiated is that specified by the Interrupt State Identifier field of the current Interrupt State Register.

During the execution of a 'Program Control' instruction a higher priority interruption can occur. When this happens, the code of the state that would have been initiated is stored in the ISI field of the Interrupt Status Register of the initiated state. Further information is given in Section 5.3.5.

2.4.2 Sequence Control Counter

2.4.2.1 Description

The Sequence Control Counter is a 24 bit register which is used to store the absolute address of the next instruction to be executed.

Staticising any instruction that is not a branch instruction causes the register to be incremented by the length of that instruction. During the execution of a branch instruction, the address specified by the instruction is converted from its virtual to its absolute form,and is then loaded into the Sequence Control Counter.

When a 'Branch and Link' instruction is obeyed, the virtual address corresponding to the contents of the Sequence Control Counter is stored in the general purpose register specified by the instruction.

Instruction executions which terminate in a request for the 'Op. Code Trap' interrupt, cause the Sequence Control Counter to be incremented by the instruction length indicated by the first bits of the information. The same length is stored in the Instruction Length Code Register.

On a 4-70 the Sequence Control Counter makes no allowance for Paged Addresses. This is also true on a 4-75 which is functioning in the Paging Mode, in this case instruction addresses are translated to their absolute form during instruction fetching.

2.4.2.2 Action on Change of State

On a change of state the address stored in the Sequence Control Counter is converted back to its virtual form and is then stored in bits 8 to 31 inclusive of the Program Counter Register of the terminated state. The Sequence Control Counter then receives a value equal to the value stored in the corresponding bits of the new state after it has been modified by the new value stored in the Relocation Base Register.

During the execution of a 'Program Control' instruction, the value stored in the Program Counter Register of the terminated state, is equal to the operand address specified by the instruction.

2.4.3 Relocation Base Register

2.4.3.1 Description

The Relocation Base Register is a 12 bit register whose contents, the 'relocation base address', are automatically added to the most significant halves of all program-generated main store data and instruction addresses. In the absence of paging, this provides an effective method of dynamic relocation allowing a program and its data to be moved unaltered between main store locations displaced by some multiple of 4,096 bytes. This action may occur at any time provided that the relocation base address is altered appropriately. The action of the paging system (see Section 19) does not inhibit the function of the Relocation Base Register and its contents are added to each address prior to their conversion by the system.

2.4.3.2 Operation

2.4.3.2.1 Introduction

The Relocation Base Register affects all instruction-specified main store access demands. There are two forms of operation both of which are described below.

2.4.3.2.2 Data Access

Instructions which cause main store to be accessed, specify a 12 bit displacement and one or two general purpose registers. Absolute address generation is performed by the concatenation of the 12 bits of the Relocation Base Register with the displacement to form a 24 bit value which is added to the least significant 24 bits of the specified register.

2.4.3.2.3 Instruction Fetching

The contents of the Sequence Control Counter are used directly to specify main store addresses required for instruction fetching. Allowance for the contents of the Relocation Base Register is made when the Counter is loaded during a change of state, and at this

2.4.3.2.3 (continued)

time the 12 bits of the Relocation Base Register are added to the top twelve bits of the Sequence Control Counter field of the Program Counter Register of the initiated state.

The resulting sum is stored in the Sequence Control Counter. The reverse action is performed when the contents of the Sequence Control Counter are stored.

2.4.3.3 Object Program Access

The address generated by a 'Load Address' instruction or stored during the execution of a 'Branch and Link' instruction is not affected by the Relocation Base Register.

The Supervisor controls the loading of the Relocation Base Register and hence object programs operate in ignorance of the value of its contents.

2.4.3.4 Loading the Register

The Relocation Base Register is only loaded during a change of state. Hence the value which it is to contain, whilst the processor is operating in a certain state, must be specified before that state is initiated. The required value is stored in the Interrupt Status Register of the particular state.

2.4.3.5 Action on Change of State

When a state is initiated, the Relocation Base Register is loaded with bits 8 to 19 inclusive of the initiated state's Interrupt Status Register. The previous contents of the register are not stored.

2.4.4 The Instruction Length Code (ILC)

2.4.4.1 Description

The Instruction Length Code Register is a 2 bit register which is loaded, during the staticising of each instruction, with the length of the current instruction. The contents of the register are stored during a change of state. The stored value may be incorrect, if the change of state was initiated as the result of one of the following interrupts :-

Power Failure
Machine Check or Address Error resulting from instruction addressing.

The contents of the Instruction Length Code register, when stored in a Program Counter Register following the execution of an interrupt, may be used to determine the address of the last instruction that was completed, terminated or suppressed prior to the interrupt, by subtracting the length of the instruction from the contents of the Sequence Control field of the Program

2.4.4.1 (continued)

Counter Register. An exception to this occurs in the use of the 'page turning' interrupt. See Section 19.

The meaning of the bits of the Instruction Length Code field of the Program Counter Register is as follows :-

Bit 1	Bit 2	Length of Instruction
0	1	Two bytes
1	0	Four bytes
1	1	Six bytes

2.4.4.2 Action on Change of State

On a change of state the contents of the Instruction Length Code Register are stored in bits 0 and 1 of the Program Counter Register of the terminated state. The contents of the register are not changed.

2.4.5 Store Protection Key

2.4.5.1 Description

The Store Protection Key is a four bit register which is compared with the appropriate field of the Store Reservation Key, whenever an attempt is made by the Arithmetic Unit to read from or write into main store. All 16 key combinations may be used. The method of comparison and the mechanism of store protection are described in detail in Section 3.4. The key can only be set during a change of state.

2.4.5.2 Action on Change of State

On a change of state, the Store Protection Key is loaded with the contents of bits 4 to 7 inclusive of the Interrupt Status Register of the initiated state. The previous contents of the register are not stored.

2.4.6 Condition Code

2.4.6.1 Description

The Condition Code Register is a 2 bit register which may be set to any of four possible values during the execution of certain instructions. The contents of the register can be tested by 'Branch on Condition' instructions. The effects that individual instructions have on the register are described for each instruction in either PS 4.6.3 or in Appendix D.

The contents of the Condition Code Register may be altered by the execution of the 'Set Program Mask' instruction. Execution of this instruction causes the contents of bits 2 and 3 of the general-purpose register specified by the instruction to be loaded into the Condition Code Register.

2.4.6.2 Action on Change of State

During a change of state, the contents of the Condition Code Register are stored in bits 2 and 3 of the Program Counter Register of the terminated state. The register is loaded with the corresponding bits of the Program Counter Register of the new state.

2.4.7 The Call Register

The Call Register of each processor state coincides with bits 24-31 of the state's Interrupt Status Register. The Call Register of a state is overwritten whenever a 'Supervisor Call' instruction is executed in that state; it remains unchanged by the processor between such occurrences.

2.4.8 Hardware Malfunction Indicators

The hardware malfunction indicators are two bits which are associated with the checking of the parity of information read from the Scratchpad and from main store. The conditions which activate the bits are described in Sections 3.2.4 and 4.3.2 respectively. The method of setting the bits is described in Section 5.3.2 and their address is given in Section 2.3.2.2.

2.4.9 Operation Modes

2.4.9.1 General Description

At all times the behaviour of the Arithmetic Unit is governed by the settings of three 'Mode Indicator' bits. Each bit determines whether the Arithmetic Unit is in one or other of two 'modes'. The three mode-pairs are independent of each other and affect different aspects of the Arithmetic Unit's behaviour.

The Mode Indicator bits are reloaded from the new state's Interrupt Status Register on every change of state.

The three mode-pairs, and the corresponding ISR bits, are as follows :-

ISR Bit	Alternative Modes
20	ISO Code/EBCDIC
21	Non-privileged/Privileged
22	Paging/Standard Address

In each case the first-named mode is in force when the corresponding Mode Indicator is 1, the other mode when it is 0. The modes are described in succeeding sections.

2.4.9.2 Action on Change of State

The Mode Indicators are loaded during a change of state with the contents of bits 20 to 22 of the Interrupt Status Register of the initiated state.

2.4.9.3 EBCDIC/ISO Mode Indicator

This indicator determines the character code used by the processor. Setting the bit to '0' causes EBCDIC to be used, whilst if set to '1' ISO is used. The effect of the bit is given in Section 1.5. .

2.4.9.4 Privileged Mode Indicator

This indicator when set to '1' causes the Arithmetic Unit to request a 'Privileged Operation' interrupt if it tries to executea 'privileged' instruction. When set to '0' any instructions may be obeyed.

Further information and a list of privileged instructions is given in Section 5.4.9.

2.4.9.5 Paging Mode Indicator

The Paging Mode Indicator controls the method used by the Arithmetic Unit for determining the addresses of instructions and data. When the Indicator is set to '0' the addresses are those specified by the current program and modified by the contents of the Relocation Base Register. When the indicator is set to '1', addresses are further modified by the Paging System. The first mode is the 'Standard Address Mode'; the second is the 'Paging Mode'. The function of the Paging System is defined in Section 19.

The Paging Mode Indicator is only effective on a 4-75 Processor which has itself been activated by the appropriate engineer's switch. Under these circumstances it is set when, on entering the current Processor State, that state is P1 or P2 and the appropriate bit is set in the Interrupt Status Register of the initiated state. 'Paging' cannot take place in P3 or P4.

2.4.9.6 Not used.

2.4.10 Interrupt Control Registers

2.4.10.1 Interrupt System .

A full description of the Interrupt System is given in Section 5.

2.4.10.2 Interrupt Flag Register

The Interrupt Flag Register is a unique register situated in the Scratchpad. Each of its 32 bits, is used to indicate the presence of a different outstanding interrupt condition, and the appropriate bit is set to '1' whenever an interrupt is requested.

2.4.10.3 Interrupt Mask Register

Each processor state has its own Interrupt Mask Register situated in the Scratchpad. Its effect is to cause any interrupt

2.4.10.3 (continued)

condition set in the Interrupt Flag Register to remain pending and not to take effect as long as the corresponding bit in the Interrupt Mask Register of the current state is zero.

As soon as the bit becomes a '1', or a new state is entered whose Mask Register has a '1' in the corresponding position, the interrupt is performed (unless delayed by one of higher priority).

2.4.10.4 Program Mask Register

2.4.10.4.1 Description

The Program Mask Register is a four bit program-accessible register whose four bits are associated with the following program error conditions :-

Significance
Exponent Underflow
Decimal Overflow
Fixed-point Overflow.

When any one of these conditions arises, and the corresponding Program Mask bit is zero, the condition is ignored (in the case of Significance errors, the way the offending instruction is completed is affected; see PS 4.6.3), and the appropriate bit in the Interrupt Flag Register is not set. When the Program Mask bit is set to '1' the error condition gives rise to an interrupt condition in the normal way.

The non-privileged instruction 'Set Program Mask' allows programs to alter the value of the register. During the execution of a 'Branch and Link' instruction, the contents of the Register are stored in the same general-purpose register as the Sequence Control Counter.

2.4.10.4.2 Action on Change of State

When a change of processor state occurs the contents of the Program Mask Register are stored in bits 4-7 of the Program Counter Register of the old state, and reloaded from the corresponding field of the new state.

2.4.11 The Timer

2,4.11.1 Introduction

The timer provides a timing facility and also an interrupt facility which may be set by program to cause an interrupt after a specified number of 1.7 millisecond interrupt cycles have elapsed. The range of times that can be covered before interrupt occurs is 1.7 to 440 milliseconds. The actual time to interrupt will be less than that specified if the specification does not occur immediately before the start of a timing cycle. The clock count may be read by

2.4.11.1 (Continued)

program; it has a cycle time of about 8 hours. The timer's overall accuracy is approximately 1 second in 8 hours.

The Timer consists physically of two registers which are defined below. Only in Privileged Mode can these two registers be accessed by programs, and they then appear to occupy main store locations 80-83. When not operating in Privileged Mode, programs referring to addresses 80-83 will access those actual locations, not the Timer registers.

The action of the Timer is not the same as that of the clock on the 4-50 processor.

2.4.11.2 The Clock Register

The Clock Register is a 32-bit register into which a '1' bit is added, in the least significant position at 6.72495 microsecond intervals. Overflow of the register causes the count to be reset to zero but does not cause an interrupt. The register cannot be reset by program and is only accessible to a program operating in Privileged Mode, using any instruction which reads out of main store addresses 80-83, except MVC which may cause parity failure. *

2.4.11.3 The Time Interrupt Register

The Time Interrupt Register is an 8-bit register which is used to specify the time that must elapse before the next clock interrupt request is generated. The contents of the register are compared with the third byte (bits 16 to 23) of the Clock Register, whenever bit 23 of the Clock Register changes (i.e. whenever carry occurs out of bit 24, approximately every 1.72 msec). The interrupt is requested when this comparison shows the two bytes to be identical.

The Time Interrupt Register can only be overwritten by a program operating in Privileged Mode, using any instruction which writes into main store address 82. It cannot be inspected by program; reading out from address 82 in Privileged Mode will produce the contents of bits 16 to 23 of the Clock Register.

3. MAIN STORE

3.1 Introduction

The main store is used to hold instructions and data; some parts of it are used for hardware functions. Access to main store is made by both the Central Control Unit and the channels. The general properties of main store are specified in Section 3.2. In the next section the methods of addressing the store are described, this is followed by a description of the method of reserving parts of the store for the use of an individual program. The functions of the reserved parts of main store are specified in Section 3.5.

3.2 General Physical Properties

3.2.1 Store Structure

3.2.1.1 Store Modules

Main store provides directly-addressable storage for a number of 8-bit bytes. With each byte is associated a 9th, parity bit, which is not accessible to programs. Bytes are consecutively addressed from 0 onwards.

Although it is possible to inspect and modify individual bytes, information is actually transferred between main store and the rest of the system in groups of four consecutively-addressed bytes called 'words'. The first byte in each word has an address which is a multiple of 4. Words are described as 'even' or 'odd' depending on whether this multiple is even or odd.

The main store can accommodate up to 1,048,576 bytes. These are arranged in four 'modules', as follows:

The 1st	module	contains	bytes	addressed	0	-	262,143
The 2nd	"	"	"	"	262,144	-	524,287
The 3rd	"	"	"	""	524,288	-	786,431
The 4th	"	"	"	"	786,432	-	1,048,575

Thus the actual number of modules present depends on the size of the store.

Each module is divided into an even half-module and an odd half-module, the former containing all the even words in the module and the latter all the odd words. The even half-modules collectively form the 'even half' of the store, and the odd half-modules the 'odd half'; which half of the store any byte belongs to is therefore determined by the 2^2 bit of its address. Thus the store is interleaved and may consist of 2, 4, 6 or 8 half-modules.

Each half-module is both independent and autonomous, and processes single store access requests made by the Store Access Unit from initiation until the Access Unit returns to complete the operation.

The store size may be any multiple from 1 to 16, of 65,536 bytes. Units (of 65,536 bytes) can only be added in the following order:-
4170, 4170/1, 4170/2, 4170/3; 4170, 4170/1, 4170/2, 4170/3;
4170, 4170/1, 4170/2, 4170/3; 4170, 4170/1, 4170/2, 4170/3.
Note: catalogue no. 4171 can be quoted to cover items 4170 + 4170/1 + 4170/2 + 4170/3 (in that order) as a single 262 Kb unit.

3.2.1.2 Store Access

3.2.1.2.1 Main store access demands are processed by the Store Access Control Unit. This unit accepts demands from both the Central Control Unit and the Input/Output Store Multiplexor. The action of these devices is described in the next sections.

3.2.1.2.2 The Input/Output Store Multiplexor

The Input/Output Store Multiplexor controls the store access demands of the Channel Control Units. It is connected to the Store Access Control Unit and has provision for the attachment of six Channel Control Accessways to it; each Accessway is attached to a single Channel Control Unit.

A channel control unit accesses main store through the Store Multiplexor, and must request its service each time it requires a store cycle to be made. The multiplexor services the channel control Units on a priority basis and having selected a channel remains connected to it until the start of the requested store cycle. Delays in starting a cycle cause all outstanding services to be delayed. Once a cycle has been successfully started the Store Multiplexor selects the highest priority outstanding requests and initiates a second cycle. It cannot however, select further requests for service until the first has been completed. A further restriction inhibits the completion of the second cycle before that of the first.

Concurrency between two input output store accesses may only be obtained if the accesses address opposite halves of the store.

3.2.1.2.3 Store Access Control Unit

The function of the Store Access Control Unit is to progress store access requests and select the appropriate half module of store. The selected module then completes the cycle and returns the required information via the Access Unit to the origin of the store request. At any one time the Access Unit can process the initial part of one access from the Central Control Unit and another from the Input/Output Store Multiplexor. Having initiated a store cycle on behalf of a unit, the Access Unit is capable of initiating a second, provided it is the other half of the store which is to be accessed.

3.2.1.2.4 Access Simultaneity

A half store module can process one access at a time and hence the store can maintain four concurrent accesses if, and only if, the two units attached to the Store Access Control Unit each require one access to each half of the store, and if the accesses to each half of the store address different half modules. It is not possible to have more than two concurrent accesses on a store that is not greater than 262Kb.

In cases of conflict between the Central Control Unit and the channels the access is given to the channels.

3.2.2 Store Configurations

The main store of the processor may be provided in one of a number of different configurations. The rules governing the

3.2.2 Continued.

construction of store configurations are given in Section 1.6.

3.2.3 Store Speed

The store has a cycle time of 900 .nanoseconds. and an access time of 360 nanoseconds. *

3.2.4 Store Parity Checking

The parity of all bytes leaving the store is checked; in addition the parity of data bytes coming to the store from the channels is checked.

The parity of all bytes entering the store is checked when they are being transferred during the execution of 'Move' instructions. Status information coming into store from a channel is not checked for parity; it is, however, given correct parity before it is stored. Further information is given in Section 6.4.4.6. *

A parity failure, other than one associated with input or output, causes the 'Machine Check' interrupt to be requested and will normally cause bit 0 of the Interrupt Status Register of the processor state 4 to be set (see Section 5.4.3).

3.3 Main Store Addressing

3.3.1 Address Generation

Absolute addresses are 24-bit binary numbers. In practice the main store contains at most 2^{20} bytes and so any address whose most significant four bits are non-zero is invalid (See Section 3.3.2 below).

The least significant 18 bits of an address specify a byte within a module and the most significant bits specify the module. Bit 2^2 selects the even or odd half module and bits 2^{18} to 2^3 select words within the half module.

All addresses supplied to the Store Access Control Unit are absolute. Certain addresses specified by the Arithmetic Unit are initially 24-bit virtual addresses, and have to be converted to absolute form. The method by which the addresses of instructions are converted is described in Section 2.4.2. Operand addresses specified by instructions are converted to absolute form as follows: each such address is generated as the sum of a 12 bit position 'displacement' and the contents of the least significant 24 bits of one or two general-purpose registers. The absolute value is formed automatically during this addition by concatenating the 12-bit contents of the Relocation Base Register with the displacement to form a 24 bit quantity which is used as an addend.

3.3.1 *Continued.*

If the Arithmetic Unit is in the Paging mode, the resultant address is then converted by the paging hardware (Section 19). Addresses generated by channels are not converted by the paging mechanism.

In this, and similar, 24 bit addition operations, bits which carry over into the 2^{24} position (overflow bits) are ignored and the store is said to 'wrap around' at 2^{24} bytes. Effectively $2^{24} - 1$ and 0 are consecutive addresses (leaving aside questions of validity).

3.3.2 Address errors

An attempt by the Arithmetic Unit to refer to a non-existent area of main store causes an 'Address Error' interrupt, to be requested (See Section 5.4.11).

When a channel operation specifies an *invalid address, the* operation is terminated with the Program Check bit set in the Channel Status Word (see Section 6.4.4.4).

3.4 Store Protection

3.4.1 Introduction

Areas of main store can be reserved for the independent use of up to 14 object programs and a Supervisor. The area reserved for a program need not be continuous; each separate area must consist of one or more blocks of store of 512 consecutive bytes. Each block has an associated Reservation Key, which may be set so as to inhibit the overwriting of any part of the block by *programs other than the one for* which the block was reserved. A block's Key is accessed whenever a word in the block is accessed.

3.4.2 Store Reservation Keys

3.4.2.1 General

A Store Reservation Key is a six bit register which has three fields. The format of a Reservation Key is as follows:-.

Comparison Field	Write Marker	Read Marker
0	4	5

Reservation Keys are loaded and stored using the 'Set Storage Key' and 'Insert Storage Key' instructions respectively. The action of these instructions is defined in Appendix D. The Central Control Unit Store Protection Key is described in Section 2.4.5, and the channel Protection Key is described in Section 6.2.2.

3.4.2.1 Continued.

The action of the fields is described in the following sections.

3.4.2.2 Protection Mechanism

When an attempt is made to write into any main store location, the contents of the 'comparison field' (most significant bits) of the Reservation Key associated with that location is automatically compared with the 4-bit Protection Key of the Central Control Unit (see Section 2.4.5) or Channel (see Section 6.2.2) initiating the access. The write action is inhibited unless these two quantities are equal, or either is zero. Inhibition of a write operation initiated by the Central Control Unit causes an Address Error interrupt to be requested (see Section 5.4.11.5); if the operation were initiated by a channel, the transfer is terminated with the Protection Check bit set in the Channel Status Byte (see Section 6.4.4.5). *

3.4.2.3 Read Protect Flag

There is no read protection facility on the 4-70.

3.4.2.4 Read Marker

The Read Marker bit of a Reservation Key is set to '1' whenever a read operation affects the block of 512 bytes associated with that bit. The bit can only be reset to 0 by the 'Set Storage Key' instruction.

3.4.2.5 Write Marker

The Write Marker bit of a Reservation Key is set to '1' whenever a write operation affects the block of 512 bytes associated with that bit. The bit can only be reset to 0 by the 'Set Storage Key' instruction.

3.5 Reserved Store

3.5.1 General

The first part of main store contains a number of areas which are used for special hardware functions. The areas from byte location 96 upwards are for use by multiplexor channels and are protected against Central Control Unit overwriting; the multiplexor channels can only write to these areas by raising a special 'Privileged Access' signal. The areas from Byte 0 upwards are protected against I-O overwriting unless the appropriate I-O Channel raises the Privileged Access signal (e.g. to store Channel Status Words).

ie. WORD 24

3.5.2 The First Part of Main Store

The Reserved Store is allocated as follows:-

3.5.2 (continued)

Store Locations	Use	Reference Section
0-23	Initial Program Load	7
24-71	Utility areas used by Decimal Multiply and Divide instructions	
32-63	Diagnose dump area	D2.9.1
72-75	Channel Address Word (CAW)	6.2.2
76-79	MCCU Diagnose dump	D2.9.2
80-83	Apparent timer location (in Privileged Mode only)	2.4.11
84-87	Not used	
88-95	Channel Status Word (CSW)	6.4.1
96-1023	Multiplexor Control Areas for first 58 multiplexor devices	3.5.3, 6.6.3
1024-2047	Ditto for further 64 multiplexor devices	
2048-4095	Ditto for further 128 multiplexor devices	
4096-8191	Ditto for further 256 multiplexor devices, on a channel or channels not associated with the previous 250 devices.	

Notes :

(i) Locations 0-95 may be overwritten by the Central Control Unit. They may only be overwritten by Channel Control Units in the presence of the Privileged Access signal. Overwriting of locations 96-8191, by any agency, is only permitted when the Privileged Access signal is present.

S.R.K. ?

(ii) Software must ensure that the storage tag on the first 512 bytes of store is zero.

3.5.3 Multiplexor Control Areas

3.5.3.1 Reservation Properties

Areas of main store reserved for multiplexor channel control words are protected against overwriting by any store demand that is not accompanied by a special 'Privileged Access' signal. *
This signal can only be generated by multiplexor channel control *

3.5.3.1 Continued.

when storing Control words, and hence the information in the reserved store is completely protected.

The response of the Store Access Control Unit to access demands which address the reserved part of main store and which are not accompanied by the reservation key is the same as if a non-existent part of store had been addressed.

3.5.3.2 The Allocation of Reserved Store

Sufficient main store may be reserved for the attachment of 506 devices to multiplexor channels. If fewer devices are required, store sufficient for 58, 122 or 250 may be reserved. In each case 4 words must be reserved for each device.

A channel must have its control information stored in the area lying either before or after, but not on both sides of main store location 4,096. Even when shared between channels, the information in an area may be arranged in any convenient manner.

3.6 Store Retention

Main store is not affected by either power failure or 'Power Off' and hence its contends after such a failure are the same as before the failure.

4. SCRATCHPAD STORE

4.1 Introduction

The Scratchpad is a small store which forms part of the Central Control Unit. It holds a set of registers which are used for a number of different purposes associated with the functioning of the Central Control Unit. The general purpose registers of each processor state, and the floating-point registers which are common to all states, are implemented in the Scratchpad. Some of the registers have special functions and are considered to be part of the Scratchpad only for convenience of addressing.

The use made of the various registers is described in the next section. This section is followed by a section which describes the general physical properties of the store. The layout of the registers are specified in Section 4.4.

4.2 Scratchpad Usage

4.2.1 General

The Scratchpad registers are used for a number of functions, some of which are associated with normal program operations and are specified in this section. The following is a list of register functions and of cross-references:-

General purpose and floating point registers are described in this section.

Interrupt control registers are described in Section 5.

Processor control registers are described in Section 2.

The associative memory registers are described in Section 19.

4.2.2 General-Purpose Registers

There is a set of sixteen general-purpose registers associated with each processor state. These registers may be addressed by instructions which specify registers as operands (see Section 4.4.2). Many of these registers have special functions as described in the sections mentioned above.

4.2.3 Floating Point Registers

The Scratchpad contains eight words which are combined in pairs to form four 64-bit floating point registers; these registers are addressed as described in Section 4.4.3. The registers may either be used as single-length or double-length registers, depending on the instructions which use them.

4.3 General Physical Properties

4.3.1 Store Structure and Performance

A scratchpad is a 72 word high-speed store. It has a 60 nanosecond access time and a cycle time which depends upon the mode of operation (Read/Write - 280 nsecs, Read/Read - 320 nsecs, Write/Write - 240 nsecs).
Each word consists of four eight bit bytes. The words of the Scratchpad are not all numbered consecutively and so its apparent size is 120 words on a 4-70 and 136 words on a 4-75.

4.3.2 Parity Checking

There is a parity bit associated with each byte of the store and a parity check is made on all bytes leaving the store. The detection of a parity error causes the machine check interrupt to be requested and will normally cause bit 3 of the Interrupt Status Register to be set (see Section 5.4.3).
After switching on, the parity of any word will not be correct until it has been loaded with an L, LM or LSP instruction.

4.4 Store Addressing and Layout

4.4.1 Introduction

Scratchpad registers may be addressed in two ways. Every register has a unique number which is used by the instructions 'Load Scratchpad' and 'Store Scratchpad' to refer to it. In addition those registers which act as general-purpose and floating-point registers may be addressed by instructions (operating in the appropriate processor state) in the usual way.

4.4.2 The General-Purpose Registers

4.4.2.1 Introduction

There are four sets of sixteen scratchpad words which may be addressed by instructions which specify a general-purpose register as an operand. Each set is associated with a different Processor State and it is the registers of the set associated with the current Processor State which are accessed when such an address is specified. Some of the general-purpose registers of P3 and P4 coincide with Control Registers, as shown in Section 4.4.2.2.

4.4.2.2 Special Functions of General-Purpose Registers

Table 1

General-purpose Register Number	Register Utilisation			
	P1	P2	P3	P4
0	GPR	GPR	IMR, P1	Segment Table Base+
1	GPR	GPR	ISR, P1	GPR
2	GPR	GPR	P-Counter P1	GPR

4.4.2.2 Continued.

General-Purpose Register Number	Register Utilisation			
	P1	P2	P3	P4
3	GPR	GPR	Interrupt Flag Register	GPR
4	GPR	GPR	IMR, P2	Address Page Turning Routine P3+
5	GPR	GPR	ISR, P2	Page Identity+
6	GPR	GPR	P-Counter, P2	GPR
7	GPR	GPR	GPR	GPR
8	GPR	GPR	IMR, P3	GPR
9	GPR	GPR	ISR, P3	*GPR
10	GPR	GPR	P-Counter, P3	*GPR
11	GPR	GPR	GPR	GPR
12	GPR	GPR	GPR	IMR, P4
13	GPR	GPR	*GPR	ISR, P4
14	GPR	GPR	*GPR	P-Counter, P4
15	GPR	GPR	Weight	Weight

Note: 'GPR' indicates that the register addressed has no special function.

* These registers are used by the instructions 'Edit and Mark' and 'Translate and Test' when executed in P3 and P4.

+ Indicates usage on a 4-75 only, on a 4-70 register is a GPR.

4.4.3 Floating-Point Registers

Floating-Point registers may be addressed (as 0, 2, 4 or 6 only) by floating-point instructions in any state. An 'Address Error' interrupt occurs if one of these instructions specifies an incorrect register address. The floating-point registers may also be addressed by the 'Load Scratchpad' and the 'Store Scratchpad' instructions. The absolute Scratchpad address of the floating-point registers is given in Section 4.4.5.

4.4.4 The Load and Store Scratchpad Instructions

The privileged 'Load Scratchpad' and 'Store Scratchpad' instructions specify absolute scratchpad addresses, and may

4.4.4 Continued.

address any register in the Scratchpad. The function of these instructions is modified when they address the Associative Memory; further details are given in Section 19.

4.4.5 Scratchpad Layout

Table 2

Word Address (Hexadecimal)	Register Assignment
00	Segment Table Base Register+.
01	General Purpose Register. P4
02	General Purpose Register. P4
03	General Purpose Register. P4
04	Address of page turning routine P3+
05	Page Identity Register+
06	General Purpose Register. P4
07	General Purpose Register. P4
08	General Purpose Register. P4
09	General Purpose Register. P4
0A	General Purpose Register. P4
0B	General Purpose Register. P4
0C	Interrupt Mask. P4
0D	Interrupt Status. P4
0E	Program Counter. P4
0F	P4 Weight Register

+ indicates usage on a 4-75 only, on a 4-70 register is a GPR.

Word Address (Hexadecimal)	Register Assignment
20	Interrupt Mask. P1
21	Interrupt Status. P1
22	Program Counter. P1
23	Interrupt Flag Register.
24	Interrupt Mask. P2
25	Interrupt Status. P2
26	Program Counter. P2
27	General Purpose Register. P3
28	Interrupt Mask. P3
29	Interrupt Status. P3
2A	Program Counter. P3
2B	General Purpose Register. P3
2C	General Purpose Register. P3
2D	General Purpose Register. P3
2E	General Purpose Register. P3
2F	P3 Weight Register
40	General Purpose Register. P2
41	General Purpose Register. P2
42	General Purpose Register. P2
43	General Purpose Register. P2
44	General Purpose Register. P2
45	General Purpose Register. P2
46	General Purpose Register. P2
47	General Purpose Register. P2

4.4.5 Continued.

Word Address (Hexadecimal)	Register Assignment	
48	General Purpose Register. P2	
49	General Purpose Register. P2	
4A	General Purpose Register. P2	
4B	General Purpose Register. P2	
4C	General Purpose Register. P2	
4D	General Purpose Register. P2	
4E	General Purpose Register. P2	
4F	General Purpose Register. P2	
60	General Purpose Register. P1	
61	General Purpose Register. P1	
62	General Purpose Register. P1	
63	General Purpose Register. P1	
64	General Purpose Register. P1	
65	General Purpose Register. P1	
66	General Purpose Register. P1	
67	General Purpose Register. P1	
68	General Purpose Register. P1	
69	General Purpose Register. P1	
6A	General Purpose Register. P1	
6B	General Purpose Register. P1	
6C	General Purpose Register. P1	
6D	General Purpose Register. P1	
6E	General Purpose Register. P1	
6F	General Purpose Register. P1	
70	Floating Point Register	
71	Floating Point Register	
72	Floating Point Register	
73	Floating Point Register	
74	Floating Point Register	
75	Floating Point Register	
76	Floating Point Register	
77	Floating Point Register	
80	Associative Memory Word 1)	
81	Associative Memory Word 2)	
82	Associative Memory Word 3)	
83	Associative Memory Word 4)	Present on
84	Associative Memory Word 5)	a 4-75
85	Associative Memory Word 6)	only.
86	Associative Memory Word 7)	
87	Associative Memory Word 8)	

Note:

1. Locations below 80 which are not mentioned above are not existent and, when addressed, respond as if their contents are zero.

4.4.5 Continued.

2. The Associative Memory registers do not physically form part of the Scratchpad; they are, however, accessed when addressed as above.

3. Addressing beyond location 127 causes an 'Address Error' interrupt to be requested on a 4-70 and also on a 4-75 operating with the appropriate Engineers' key set to inhibit paging. When addressing the Associative Memory is permitted, any address less than 256 is valid and 'wrap-around' occurs on addresses greater than 256.

4.5 Store Retention

The contents of the Scratchpad are not preserved during the occurrence of a power failure and must be stored in Main Store during the millisecond that precedes power loss.

5. THE INTERRUPT CONTROL SYSTEM

5.1 Introduction

5.1.1 Section Introduction

In this section the Interrupt Control System is described. The nature and function of the system is mentioned in this introduction and a detailed definition of its action is given in Section 5.3. This section includes details of the action of the 'Program Control' instruction. The registers associated with the system are described in Section 5.2, and the nature of the individual interrupt conditions is defined in Section 5.4.

5.1.2 System Function

A necessary requirement of a multiprogramming computer system, which can maintain many concurrent input and output operations, is a control system which enables the Supervisor to be entered as soon as important control information becomes available, or an error condition occurs. These functions are performed by the Interrupt Control System. This system presents control information to the Supervisor by means of a change of state.

5.1.3 Interruptions

Occurrences which demand that the processor's current operation be suspended and that a change of state should occur, are called interrupt conditions. Interrupt conditions may occur in a number of different parts of the computer system, and each occurrence causes a request to be made to the Interrupt Control System for the execution of its associated interrupt. The execution of an interrupt causes a change of state to occur and permits the presentation of associated control information to the Supervisor. The Interrupt Control System is also activated by the execution of a 'Program Control' instruction.

Interrupt conditions may be grouped into 6 classes each of which is treated differently by the Interrupt Control System. The method of classification and the properties of the classes are described in Section 5.3.1. The priority order of the interrupt conditions is shown in Section 5.2.2.2.

5.1.4 Features of the Interrupt Control System

One of the primary functions of the Interrupt Control System is to respond to requests for interrupt execution and to staticise the requests in a priority queue which is maintained in a register named the 'Interrupt Flag Register'. Control over the execution of the interrupt requests is maintained by program with the use of mask registers, and it is only those interrupts which are not inhibited by masks that can be selected for execution.

The operation of the Interrupt Control System causes instruction execution to be suspended for a time, the length of which is dependent on whether or not a change of state occurs. Information on the action times of the Interrupt System is given in Appendix B.

5.2 The Registers of the Interrupt Control System

5.2.1 Introduction

A description of the registers associated with the Interrupt System is given in this section. The use made of the registers by the system is described in Section 5.3.

5.2.2 The Interrupt Flag Register

5.2.2.1 Introduction

The Interrupt Flag Register is used to list requests for interrupt executions. The list is used by hardware to determine which interrupt, if any, should be executed; it may be referenced by program to see which interrupt conditions are pending.

Each Interrupt condition has an associated 'Flag' bit in the Interrupt Flag Register. The relation between individual interrupt conditions and the Flag bits is given in the next section. A Flag bit is set to '1' whenever its associated interrupt is staticised. It is reset to '0' when the interrupt is executed. The resetting occurs during the resulting change of state.

The Interrupt Flag Register is a 32 bit scratchpad register. Its address is given in Section 4.4.5.

The register is not protected and may be written into by a program which is operating in Processor State Three or which specifies the 'Load Scratchpad' instructions.

If flags are reset as a result of a write operation which addresses the Interrupt Flag Register, the associated interrupts are lost. Should a channel's interrupt flag be reset, then further interrupt requests from the channel are kept pending. Input/Output interrupts must not therefore be changed by program.

5.2.2.2 Classification of Interrupt Flags

A definition of each of these conditions is given in Section 5.4.

Table 3

Priority	Interrupt Condition	Flag (Bit position)	State Initiated	Weight
1.	Power Failure	31 (2^0)	4	0
2.	Machine Check	30 (2^1)	4	4
3.	External Signal No. 0/Channel No.0	29 (2^2)	3	8
4.	External Signal No. 1/Channel No.1	28 (2^3)	3	12

cont/.....

Priority	Interrupt Condition	Flag (Bit position)	State Initiated	Weight
5.	External Signal No. 2/Channel No.2	27 (2^4)	3	16
6.	External Signal No. 3/Channel No.3	26 (2^5)	3	20
7.	External Signal No. 4/Channel No.4	25 (2^6)	3	24
8.	External Signal No. 5/Channel No.5	24 (2^7)	3	28
9.	Channel No. 6.	23 (2^8)	3	32
10.	Channel No. 7.	22 (2^9)	3	36
11.	Channel No. 8.	21 (2^{10})	3	40
12.	Channel No. 9.	20 (2^{11})	3	44
13.	Channel No. 10.	19 (2^{12})	3	48
14.	Channel No. 11.	18 (2^{13})	3	52
15.	Channel No. 12.	17 (2^{14})	3	56
16.	Channel No. 13.	16 (2^{15})	3	60
17.	Channel No. 14.	15 (2^{16})	3	64
18.	Channel No. 15. Mult	14 (2^{17})	3	68
19.	Elapsed Time Clock	13 (2^{18})	3	72
20.	Console Interrupt Request	12 (2^{19})	3	76
21.	Supervisor Call Instruction	11 (2^{20})	3	80
22.	Privileged Operation	10 (2^{21})	3	84
23.	Op-Code Trap	9 (2^{22})	3	88
24.	Address Error (execute, addressing specification, protection, paging)	8 (2^{23})	3	92
25.	Data Error	7 (2^{24})	3	96
26.	Exponent Overflow	6 (2^{25})	3	100
27.	Divide Error	5 (2^{26})	3	104
28.	Significance Error*	4 (2^{27})	3	108
29.	Exponent Underflow*	3 (2^{28})	3	112
30.	Decimal Overflow*	2 (2^{29})	3	116
31.	Fixed Point Overflow*	1 (2^{30})	3	120
32.	Program Test	0 (2^{31})	3	124

* These interrupt conditions may be masked both by the Interrupt Mask Register and the Program Mask Register.

5.2.3 The Interrupt Mask Registers

The Interrupt Mask Registers are used to indicate which interrupt conditions are permitted to interrupt the current program. There is one mask register for each processor state and in general they are loaded so that, should state switching occur to an interrupt response state, then that state is protected against further interruptions. Individual states are used in the manner described in Section 2.2.2.

The interrupt mask registers are 32 bit scratchpad registers. Their address is given in Section 4.4.5. There is one mask register associated with each state and at any particular time it is the mask register associated with the operating state which is used by hardware.

The individual bits of the Interrupt Mask Registers are associated with the same interrupt conditions as the corresponding bits in the Interrupt Flag Register. Setting a bit to '1' will permit the corresponding interrupt to be executed whilst setting the bit to '0' will inhibit the interrupt and leave it pending.

The Interrupt Mask Register should not be confused with the Program Mask Register described in Section 2.4.10.4. The latter is able to prevent certain error conditions from setting their associated interrupt flag bits. The Interrupt Mask Register controls the time at which flag bits which <u>have</u> been set take effect.

5.2.4 The Weight Registers

The Weight Register is a 32 bit Scratchpad register which is used to identify an interrupt condition causing the initiation of a new state. After such a condition occurs, the weight register of the new state is cleared and a 'weight code' is inserted into its least significant seven bits. This code identifies the interrupt condition associated with a condition is in the manner descr;bed. in Section 5.2.2.2.

Processor states 3 and 4 both have their own weight registers, the address of these registers is given in Section 4.4.5.

5.3 Action of the Interrupt Control System

5.3.1 Introduction

When an interrupt condition occurs, it generates an interrupt 'request'. The Interrupt System processes an interrupt request in three distinct phases. The first is the initial response to a request and the staticising of the request, this is described in Section 5.3.2. The second is the selection from the list of pending interrupts of an interrupt to be executed; this process is described in Section 5.3.3.

The third stage of the process is the execution of an interrupt and this is described in Section 5.3.4. The effect these operations have on the processor is to a large extent decided by the class of interrupt being processed; a description of the properties of the various classes of interrupt is given in Table 4. The action of the Interrupt Control System when it has been activated by the execution of a 'Program Control' instruction is described in Section 5.3.5.

5.3.2 Interrupt Response and Staticising (Phase 1)

Interrupt conditions may occur at any time. The method of processing them is dependent on their classification, see Table 4. Those of Classes 3 and 4 can only activate the Interrupt Control System after the execution of an instruction by the Central Control Unit. All processes of the Interrupt System are completed before the next instruction in the interrupted sequence is obeyed. Under these circumstances the processing of an interrupt has no damaging effect on the interrupted instruction sequence.

When Class 6 interrupt conditions (not suppressed by the Program Mask) arise, the instruction is completed. In the case of Class 5 interrupts, the execution of the instruction is terminated or suppressed.

Class 1 interrupts activate the Interrupt System immediately they are detected and cause the termination of the current instruction even if the interrupt is masked. If it is masked, and the interrupt occured during instruction staticisation, the next instruction may be incorrectly addressed. In the case of the 'Machine Check' interrupt the cause of the interrupt is identified as either main store or Scratchpad or both by setting, if they are not already set, one or both of two one bit staticising registers reserved for the purpose. Subsequently, when the 'Machine Check' interrupt is next executed, the registers are reset and their contents are loaded into the appropriate bits of the Processor State 4 Interrupt Status Register.

The end result of the first stage of the interrupt sequence is to set, (if it is not already set), the flag (in the Interrupt Flag Register) appropriate to the interrupt request.

The length of time required for interrupt processing depends on whether or not an interrupt is executed: if no interrupt is executed the delay is small (see Appendix B).

5.3.3 The Selection of an Interrupt (Phase 2)

The interrupt selection sequence of the Interrupt System is activated whenever one of the following events occurs:-

1. A flag is staticised in the Interrupt Flag Register. (See Section 5.2.2). For this purpose it does not matter if the same flag is already set.

TABLE 4

Type	Interrupt Class	Priority	System Response	Program Restrictions	Effect of Masking Interrupt	Effect of a second Interrupt on an unexecuted Interrupt
1	Machine Fault	1 and 2	Immediate	Must only be masked to inhibit further interrupts from same source.	Indeterminate	Previous Interrupt is lost
2	Page Turning	immediate	See Section 19.2.4.	-	-	-
3	External Status (includes clock)	3 to 20	Between instructions	-	Interrupt remains waiting	Channel Interrupts Second interrupt staticised immediately after first is executed. Other Interrupts Previous Interrupt lost.
4	Supervisor Call, Program Test	21 and 32	At termination of instruction execution	-	Indeterminate	Previous Interrupt lost.
5	Instruction Error	22 to 27	During instruction (instruction suppressed or terminated)	Should not be masked in P_1. Should not occur in P_2, P_3, P_4.	Indeterminate or as specified.	Previous Interrupt lost.
6	Calculation Error	28 to 31	At end of instruction execution The setting of the Program Mask Register controls the completion of the instruction.	As above.	The result is described in the individual interrupt and instruction definitions.	Previous Interrupt lost.

2. The current Interrupt Mask Register is overwritten; this condition assumes that any previously masked interrupt flags may cause an immediate interrupt.

3. A 'Program Control' instruction is executed and its Program Test bit is <u>not</u> set.

When one of these events occurs, the flags of the Interrupt Flag Register and the mask bits of the current Interrupt Mask Register are scanned in order of descending priority. The interrupt selected for execution is the first one reached which is set to '1' and is not masked. If no such interrupt is found, the Central Control Unit resumes its normal operation.

5.3.4 The Execution of an Interrupt (Phase 3)

When an interrupt has been selected by the Interrupt System the associated flag in the Interrupt Flag Register is reset, and, if the interrupt is of class 1, Processor State 4 is initiated. Otherwise Processor State 3 is initiated.

During the change of state the contents of the Control Registers are placed in the Status Registers as described in Section 2.3. Fresh information is then loaded into the control registers as described in Section 2.3 and the interrupt condition is identified to program by placing the relevant weight in the appropriate Weight Register.

In the case of the execution of a channel interrupt, channel status information is placed in the appropriate Channel Status Word. The Interrupt Flag associated with the channel is set again if there are further interrupt requests associated with the Channel.

At the termination of the state changing sequence normal processing commences with the execution of the instruction at the address specified in the Sequence Control Counter. At least one instruction must be executed in the new state before any outstanding interrupt request causes the Interrupt Control System to be reactivated.

5.3.5 Interrupt Response to the Program Control Instruction

5.3.5.1 Introduction

The Interrupt Control System is activated during the execution of a 'Program Control' instruction. The control sequence initiated is dependent on the setting of the Program Test bit of the instruction (bit 3 of the Immediate Operand). The properties of this bit are specified in the following sections.

5.3.5.2 Program Test Bit Set

If the Program Test bit of the Program Control instruction is set to '1' a normal change of processor state takes place. The initiated state is that specified by the instruction in the manner described in Section 2.4.1. Operations commence in the initiated state with the execution of an instruction and all interrupt requests are inhibited until the instruction has been executed. Immediately following this execution, the Program Test interrupt is requested. The Interrupt Sequence then proceeds in the normal manner.

5.3.5.3 Program Test Bit Not Set

If the Program Test bit of the 'Program Control' instruction is set to '0' the actions associated with a normal change of processor state are executed. The Central Control Unit control registers other than the Interrupt State Identifier are not reloaded, however, until a scan has been made of the Interrupt Flag Register and the Interrupt Mask Register of the state to be initiated. If, as a result of this scan an interrupt is executed, the execution proceeds as for a normal interrupt. The 'Interrupt State Identifier' field of the state initiated as a result of this interrupt execution contains the value associated with the state specified by the 'Program Control' instruction. (See Section 2.4.1.2.2).

The overridden change of state can then be re-initiated, using the information stored in 'Interrupt State Identifier' field to specify the state. If no interrupt is executed as a result of the flag scan, the change of state is completed normally and processing commences in the state specified by the 'Program Control' instruction.

5.4 Individual Interrupt Conditions

5.4.1 Introduction

The individual interrupt conditions are described in this section. The associated actions are described in the sections of this document devoted to the relevant aspects of the processor. Further definitions of the effect that instruction instigated interrupts have on the instigating instruction are given for the privileged instructions in Appendix D, and for the other instruction in PS 4.6.3. Further information concerning the action of channel interrupts is given in Section 6.3. The 'Page Turning * Interrupt' is described in Section 19.

5.4.2 Power Failure (2^0 - Flag Bit 31)

The power failure interrupt is requested any time a power failure occurs in the processor or store as the result of a line failure or the depression of the 'Emergency Off' button on the console. The power failure interrupt flag is set and the remainder of the instruction is suppressed. It is a program

cont/.....

restriction that the mask bit in processor state P4 for this interrupt condition must always be zero. This permits the program to operate in Processor State P4 for the purpose of closing down the machine during a one-millisecond interval between power failure and actual power loss to the system.

5.4.3 <u>Machine Check (2^1 - Flag Bit 30)</u>

The machine check interrupt is requested whenever a parity error is detected as the result of a read operation instigated by the Central Control Unit to either the Scratchpad or main store. The machine check interrupt flag is set and the remainder of the current instruction is suppressed. If the interrupt is masked, the Arithmetic Unit continues by executing the next instruction in sequence. The method of distinguishing between the possible causes of this interrupt are given in Section 2.4.8. Further information is given in Sections 2.4.8 and 5.3.2. It is a program restriction that the mask bit in Processor State 4 for this interrupt condition must always be zero.

5.4.4 <u>External Signal Interrupts</u>

5.4.4.1 <u>External Signal 0 (2^2 - Flag Bit 29)</u>

The external signal interrupt is requested whenever an external signal is received on Direct Control Trunk 0 (DCT 0) associated with the Direct Control Option. The appropriate interrupt flag is set. If the appropriate Direct Control Trunk is not fitted the interrupt may be used for a channel (see Section 5.4.5).

This interrupt condition may only be set if the relevant Direct Control Trunk is connected. See Section 8 for further details. This interrupt may occur only after completion of the current instruction.

5.4.4.2 <u>External Signal 1 to 5 (2^3 - Flag Bit 28 to 2^7 - Flag Bit 24)</u>

Identical to above except for trunks 1 to 5.

5.4.5 <u>Channel Interrupts</u>

5.4.5.1 <u>Channel 0 (2^2 - Flag Bit 29)</u>

This interrupt provides the means by which the processor can receive and act upon signals from peripheral devices connected to channel 0. Requests for interruption may occur at any time and more than one request may occur at the same time. The requests are saved by the appropriate device control units until accepted by the processor and are then processed one at a time.

This interrupt can occur only after execution of the current instruction is completed. Final status information is stored in the main store in the Channel Status Word. Further information is given in Section 6.3. *

Direct control trunks, if fitted, have the highest priority interrupts associated with them. If they are not fitted, a channel may make use of the interrupt. The relationship between channel number and direct control trunk number is given in Section 5.2.2.2.

5.4.5.2 Channel 1 to 15 (2^3 - Flag Bit 28 to 2^{17} - Flag Bit 14)

These interrupts are identical to the 'channel 0' interrupt save for their associatin with different channels and priorities.

5.4.6 Elapsed Time Clock Interrupt (2^{18} - Flag Bit 13)

The Elapsed Time Clock interrupt is requested when the third byte of the Clock Register becomes equal to the Time Interrupt Register. Further details are given in Section 2.4.11.

5.4.7 Console Request (2^{19} - Flag Bit 12)

The 'console request' interrupt is effected whenever the interrupt key on the operator's console is operated. The console request flag is set. This interrupt can occur only after execution of the current instruction.

5.4.8 Supervisor Call (2^{20} - Flag Bit 11)

The 'Supervisor Call' interrupt is requested as a result of the execution of the 'Supervisor Call' instruction. The Program Counter Register and Interrupt Status Register of the interrupted state are updated normally. The rightmost 8 bits of the Interrupt Status Register (the Call Register) of the terminated state receives the R_1/R_2 field of the instruction.

Further information is given in Section 2.4.7.

5.4.9 Privileged Operation (2^{21} - Flag Bit 10)

The 'privileged operation' error arises when a Privileged Instruction is attempted in the Non-Privileged Mode. The instruction's execution is suppressed. Further information is given in Section 2.4.9.4. The Privileged Instructions are as follows:-

Set Storage Key
Insert Storage Key
Write Direct
Read Direct
Diagnose
Start Device
Test Device
Halt Device

cont/.....

This interrupt can occur only after execution of the current instruction is completed. Final status information is stored in the main store in the Channel Status Word. Further information is given in Section 6.8.

Direct control trunks, if fitted, have the highest priority interrupts associated with them. If they are not fitted, a channel may make use of the interrupt. The relationship between channel number and direct control trunk number is given in Section 5.2.2.2.

5.4.5.2 Channel 1 to 15 (2^3 - Flag Bit 28 to 2^{17} - Flag Bit 14)

These interrupts are identical to the 'channel 0' interrupt save for their association with different channels and priorities.

5.4.6 Elapsed Time Clock Interrupt (2^{18} - Flag Bit 13)

The Elapsed Time Clock interrupt is requested when the third byte of the Clock Register becomes equal to the Time Interrupt Register. Further details are given in Section 2.4.11.

5.4.7 Console Request (2^{19} - Flag Bit 12)

The 'console request' interrupt is effected whenever the interrupt key on the operator's console is operated. The console request flag is set. This interrupt can occur only after execution of the current instruction.

5.4.8 Supervisor Call (2^{20} - Flag Bit 11)

The 'Supervisor Call' interrupt is requested as a result of the execution of the 'Supervisor Call' instruction. The Program Counter Register and Interrupt Status Register of the interrupted state are updated normally. The rightmost 8 bits of the Interrupt Status Register (the Call Register) of the terminated state receives the R_1/R_2 field of the instruction.

Further information is given in Section 2.4.7.

5.4.9 Privileged Operation (2^{21} - Flag Bit 10)

The 'privileged operation' error arises when a Privileged Instruction is attempted in the Non-Privileged Mode. The instruction's execution is suppressed. Further information is given in Section 2.4.9.4. The Privileged Instructions are as follows:-

Set Storage Key
Insert Storage Key
Write Direct
Read Direct
Diagnose
Start Device
Test Device
Halt Device

cont/.....

Check Channel
Program Control
Load Scratchpad
Store Scratchpad
Idle.

5.4.10 Op Code Trap (2^{22} - Flag Bit 9)

The 'Op Code Trap' error occurs when an attempt is made to execute an instruction which has an operation code that is either not assigned or not available on the processor. No operation is performed. If this interrupt is masked, the current instruction is ignored.

5.4.11 Address Error (2^{23} - Flag Bit 8)

5.4.11.1 Introduction

'Addressing' errors cover five classes of error. These are: execute error, addressing error, specification error, protection error and paging error. All these errors are directly attributable to the instruction executed at the time of the interrupt, rather than to the data the instruction was intended to manipulate.

There is no direct method for determining which of the several possibilities actually caused an address error interrupt, the cause can only therefore be inferred from the contents of the control registers, etc.

Note that, because of the way in which 4-70's instruction sequencing mechanism 'pre-fetches' the next word of instructions wherever possible, Address Errors may be erroneously reported if instructions are held either in the last word of the store or, when in the paging mode, in the last word of a segment when there is no physical location corresponding to the next virtual address in sequence.

5.4.11.2 Execute Error

This occurs when an 'Execute' instruction specifies another 'Execute' instruction to be performed. The operation is suppressed.

5.4.11.3 Addressing Errors

5.4.11.3.1 Main Store Address

The 'Main Store Address' error occurs when an address specifies any part of data or an instruction which is outside the available store for the particular installation. The instruction operation is terminated for an invalid data address and the results of the instruction are unpredictable. The instruction operation is suppressed for an invalid instruction address.

This error also occurs if any instruction attempts to overwrite a location in the Multiplexor Control Areas of main store (bytes 96-N, where N = 1023, 2047, 4095 or 8191 - see Section 3.5) or if the instruction 'Set Storage Key' specifies an address in the range 96-N.

5.4.11.3.2 Register Address Error

The 'Register Address' error occurs when either of the following conditions has occurred:-

(1) The first operand address field of an instruction designates an odd register address for a pair of general registers that contain a double word operand. The operation is suppressed.

(2) A floating point instruction addresses a floating point register other than 0, 2, 4 or 6. The operation is suppressed.

5.4.11.4 Specification Errors

5.4.11.4.1 Misalignment

The 'Misalignment' error occurs when a data or instruction address does not specify a double word, word, or halfword boundary as required by the particular instruction concerned. The operation is suppressed.

5.4.11.4.2 Decimal Arithmetic

The 'Decimal Arithmetic' error occurs when either of the following conditions occurs:-

(1) The multiplier or divisor in decimal arithmetic exceeds 15 digits and sign. The operation is suppressed.

(2) The first operand field is not longer than the second operand field in decimal division or multiplication. The operation is suppressed.

5.4.11.5 Write Protection Error

The 'Write Protection' error occurs when, during the execution of an instruction an attempt is made to write to an area of Main Store for which the contents of the comparison field of the associated Reservation Key do not match the contents of the Store Protection Key, and neither of the keys is zero. The instruction is suppressed if the keys are found not to match at the first access; it is terminated with unpredictable results if the protection failure occurs during some subsequent access.

5.4.11.6 Paging Errors

5.4.11.6.1 Introduction

Paging Errors can only occur on a 4-75 operating in the paging mode. In all cases the execution of the current instruction is suppressed, and if the 'Address Error' interrupt is masked, the Arithmetic Unit proceeds with the execution of the next instruction.

5.4.11.6.2 Segment Number Error

The 'Segment Number' error occurs when a segment number is specified which is beyond the limit specified in the Segment Table Base Register.

5.4.11.6.3 Table Addressing Error

This error occurs when the address of a Segment Table entry, or the address of a Page Table entry calculated by adding the Page Number to a Segment Table entry, is invalid.

5.4.11.6.4 Page Number Error

The 'Page Number' error is requested when a new entry is placed in the Associative Memory which has a Page Number different from that specified by the Paged Address. This only applies if the action occurs during the conversion of a Paged Address; it indicates an incorrect Page Number.

5.4.12 Data Error (2^{24} - Flag Bit 7)

5.4.12.1 Introduction

Data Errors are caused by conditions which occur during the execution of an instruction as a result of the data it manipulates. The current operation is terminated (suppressed if the operation is ('Convert to Binary') upon detection of any of the errors.

5.4.12.2 Invalid Codes

The 'Invalid Codes' error occurs when the sign or digit codes of operands in decimal arithmetic, editing or 'Convert to Binary' are invalid.

5.4.12.3 Decimal Data

The 'Decimal Arithmetic' error occurs when one or other of the following conditions is detected:-

(1) Fields overlap incorrectly in decimal arithmetic.

(2) A decimal multiplicand has too many high order significant digits.

5.4.12.4 'Scratchpad Addressing Error'

The 'Scratchpad Addressing Error' occurs during the execution of either a 'Load Scratchpad' or 'Store Scratchpad' instruction when addresses beyond 127 (decimal) are specified. This error is not relevant to a 4-75 operating with the appropriate engineers' key set to permit paging.

5.4.13 Exponent Overflow

The 'Exponent Overflow' error occurs when the result characteristic of a floating point addition, subtraction, multiplication or division is greater than 127. The operation is terminated.

5.4.14 Divide Error (2^{20} - Flag Bit 5)

The 'Divide' error condition arises when any of the following occur:-

(1) A quotient exceeds the register size in fixed point division, including division by zero. The division is suppressed.

(2) The result of a 'Convert to Binary' instruction exceeds one word. The conversion is completed by ignoring information which is outside the register size. Data above 32 bits is truncated and only the lower 32 bits are saved.

(3) A quotient exceeds the specified data field in decimal divide. The division is suppressed.

(4) Floating point division is attempted with a divisor whose fraction is zero. The operation is suppressed.

5.4.15 Significance Error (2^{27} - Flag Bit 4)

This error occurs when the result fraction of a floating-point add or subtract instruction is zero. If the Program Mask bit is zero, the error condition is suppressed and the operation is completed by setting the sign and characteristic of the result to zero. If the Program Mask bit is not zero, the 'Program Error' interrupt flag is set and the operation is completed with the characteristic unaltered. The Interrupt Mask bit does not affect the result.

5.4.16 Exponent Underflow (2^{28} - Flag Bit 3)

The 'Exponent Underflow' error occurs when the result characteristic of a floating point addition, subtraction, multiplication or division is less than zero. The operation is completed by making the result true zero (all zeroes). The condition may be suppressed by the 'Exponent Underflow' program mask bit.

5.4.17 Decimal Overflow (2^{29} - Flag Bit 2)

The 'Decimal Overflow' error occurs when the result field is too small to contain the result of decimal add, subtract or zero and add. The operation is completed by ignoring the overflow data. The condition may be suppressed by the 'Decimal Overflow' program mask bit.

5.4.18 Fixed Point Overflow (2^{30} - Flag Bit 1)

The 'Fixed Point Overflow' error occurs when a high order carry occurs or high order significant bits are lost in fixed point addition, subtraction, shifting, or sign control operations. The operation is completed by ignoring the overflow data. The condition may be suppressed by the 'Fixed Point Overflow' program mask bit.

5.4.19 Program Test (2^{31} - Flag Bit 0)

The 'Program Test'interrupt provides program control over the processor during program testing. The 'Program Test' interrupt flag is set during the execution of a 'Program Control instruction which has its Program Test bit set. When the related interrupt mask bit is set, the interrupt is effected after the first instruction executed in the program state initiated by the instruction. Servicing the interrupt causes the interrupt to be reset. When the interrupt mask bit is reset := 0 the interrupt is not taken but remains pending.

6. THE PERIPHERAL CONTROL SYSTEM

6.1 Introduction

6.1.1 Peripheral Operations and Sub-channels

There are only four input-output instructions in the System 4 order code. They are all in SI format and all privileged. The instructions, described in detail in Appendix D, are:

Start Device
Test Device
Halt Device
Check Channel

Peripheral operations, usually transfers of information between the 4-70 main store and some peripheral device attached to the processor, are initiated by the Central Control Unit using the Start Device instruction, and after initiation proceed in parallel with program execution, so that after the Central Control Unit has executed Start Device it may continue executing other instructions while the peripheral operation is in progress. Subject to certain restrictions, further operations may be initiated on other devices, which can proceed in parallel with each other and with instruction execution in the Central Control Unit.

A peripheral operation consists of a sequence of one or more actions affecting the same peripheral device. Examples of such actions are:-

a write transfer causing the contents of specified consecutive main store locations to be written onto the peripheral medium;

a read transfer causing specified consecutive main store locations to be overwritten by source information from the peripheral medium;

a 'housekeeping' operation (e.g. initiation of 'Rewind' on a magnetic tape);

the transfer to specified main store locations of secondary information relating to the condition of the peripheral device.

One 'Start Device' instruction can thus initiate several successive actions on the same peripheral device. Each action is specified by a 'command', held in main store, and the effect of the Start Device instruction is to specify a particular peripheral device and to initiate the execution of a sequence of commands just as if the latter were a program of instructions. The sequencing of successive commands is carried out by a logical unit called a 'Sub-channel'. One sub-channel may control several devices, but can only execute one operation at a time. The number of peripheral operations which may proceed in parallel is therefore equal to the number of sub-channels, and this is determined by the particular types of input-output hardware present in the system (described in Section 6.6).

The use of the term 'sub-channel' is consistent with IBM and RCA terminology; in general the peripheral control system, without being fully compatible (e.g. different command codes are used), follows IBM precedents quite closely.

At the end of a peripheral operation, the Central Control Unit is interrupted, and at the same time 'status' information relating to the operation and to the peripheral device concerned is placed in a standard location in main store. This information enables the Supervisor program to determine completely the identity of the peripheral device concerned and whether or not the operation was successful, and, in case of failure, to initiate appropriate error procedures. Since only one set of main store locations is available for the status information, the input-output and interruption hardware automatically ensures that, should more than one operation terminate and request interruption at the same time, only the interruption associated with one sub-channel is executed, and only the status information associated with that sub-channel is then placed in main store; interruptions associated with other competing sub-channels remain pending and the associated status information is not stored until they are actually executed.

These interruptions take place in an order determined by the relative priorities of the sub-channels and by the presence or absence of bits in the Interrupt Mask Register (see Section 5.2).

Status information may also be stored, in the same locations, at the time when Start Device, Test Device, or Halt Device are executed. To avoid the possibility of interruption occurring at the same time, and one or other set of stored status information thereby getting corrupted, all peripheral interruptions have to be inhibited (masked) while one of these instructions is executed, until such time as any status information stored during execution has been moved elsewhere.

6.1.2 Channels

6.1.2.1 Introduction

A channel is a logical System 4 unit for the passage of information between main store and peripheral devices, having the following properties:-

(i) It may have one or more peripheral devices attached to it. These devices are individually numbered, from 0 up to a maximum of 255.

(ii) It includes one or more sub-channels which control the peripheral operations on the devices connected to the channel (and no others). The sub-channels are not explicitly numbered.

(iii) Each channel has its own associated interrupt flag in the Interrupt Flag register (see Section 5.2.2). Interruptions associated with any of the sub-channels of the channel all set this flag, and generate the appropriate weight (see Section 5.2), when executed.

(iv) Each channel has a unique number. On 4-70 this number is in the range 0-15. The number is used by the four input-output instructions to address the required channel, and also to associate the channel with the appropriate flag in the Interrupt Flag Register. Since the flag bits for channels 0-5 also may be assigned to Direct Control trunks, a 4-70 having n of the latter is only permitted to have (16-n) channels.

There are two types of channel, 'Selector' channels and 'Multiplexor' channels. These are described below.

6.1.2.2 Selector Channels

A Selector Channel is one having only one sub-channel. Although many devices may be attached to a Selector Channel; only one of them can be involved in a peripheral operation at any time.

6.1.2.3 Multiplexor Channel

A Multiplexor Channel is one which includes as many sub-channels (up to a maximum of 256) as it has devices connected; thus, all the devices connected to a Multiplexor channel may be engaged in peripheral operations simultaneously.

6.1.3 Device Numbering

All the devices attached to any channel are numbered from 0 to 255. The instructions Start Device, Test Device, and Halt Device nominate a particular device by specifying a 12-bit address, 4 bits of which are the channel number and the other 8 form the device number on that channel. In addition, the status information stored when a sub-channel interrupt is executed includes the device number which, in conjunction with the interrupt weight (itself in one-to-one correspondence with the channel number), serves to identify the device concerned uniquely. When several devices attached to a Multiplexor channel all share the same Device Control Unit, their device numbers will have the same pattern in their more significant bits; thus, all the devices attached to a DCU capable of controlling up to 8 devices will have the more significant 5 bits of their device numbers identical.

6.2 The Mechanism of Peripheral Operations

6.2.1 The Start Device Instruction

'Start Device' is a privileged instruction in SI format. Its immediate operand field is ignored. The least significant 12 bits of the operand address are interpreted as the number of the peripheral device on which the operation is to be performed, and this serves to alert the sub-channel controlling the device.

When the Start Device instruction (as distinct from the peripheral operation it is intended to initiate) is complete, the setting of the Condition Code indicates whether or not the peripheral operation has been successfully initiated. If the Condition Code is 0, it has; if 1, 2, or 3, it has not. Condition Code setting 1 further indicates that status information has been placed in standard locations in main store. Because of this possibility, all peripheral interrupts must be masked when Start Device is executed, because they would also cause status information to be placed in the same locations. The conditions which lead to the setting of Condition Codes 1, 2 or 3 are detailed in Appendix D.2.1; in what follows it is asumed that the peripheral operation is successfully initiated.

As stated in Section 6.1.1., a peripheral operation is controlled by the appropriate sub-channel executing a sequence of 'commands' held in main store, rather like a conventional program of instructions. The Start Device instruction does not itself specify the locations of these commands, which must have been placed in the store prior to the execution of Start Device; instead, the sub-channel obtains the address of the first command from bytes 73-75 of the main store, and the protection key which it is to use for the duration of the operation (in exactly the same way as a program does, to prevent the sub-channel from overwriting forbidden areas) from byte 72. Bytes 72-75 are collectively referred to as the 'Channel Address Word' (CAW) and must be loaded with the required key and address prior to the execution of Start Device.

Commands are held in 'Channel Command Words' (CCW). Each CCW occupies 8 consecutive bytes, starting at a location which is a multiple of 8. The detailed format of a CCW is described in Section 6.2.3 below; essentially each CCW specifies a command code, a main store address (usually the starting address of a set of consecutive locations which are to be read from or written to in a peripheral transfer) and a byte count, to specify the length of the transfer. Successive CCWs are stored in consecutive locations, and in general the sub-channel sequences commands by updating the address of the current CCW by 8 bytes at a time; the exceptions to this are:-

(i) when a particular type of command ('Transfer in Channel'), which acts as an unconditional jump (to the CCW at the address specified), is executed;

(ii) when a particular condition ('Status Modifier') arises in the peripheral device concerned, causing the sub-channel to update the address of the current CCW by 16, instead of 8 bytes, and thus to skip the next CCW in sequence. Usually the command that is skipped is a Transfer in Channel, and the two features just described are used together to provide a 'looping' facility for CCWs.

The continuation of a peripheral operation from one CCW to the next is called 'chaining'. Each CCW (except a Transfer in Channel command, for which continuation is implicit) specifies whether or not chaining is to take place at the termination of its action - if no chaining is specified, the whole operation is terminated. Two types of chaining are possible:-

(i) Data chaining: this is requested when, at the end of the action specified by a CCW, the current 'block' on the peripheral medium is not exhausted, and the action is to be continued by the next CCW using exactly the same command code as the preceding one, but with a new store address and byte count. It is possible, e.g., to input successive records in the same block to different parts of the store, each record being read by a new CCW 'data chained' to its predessor. Furthermore, if the lengths of these records are unknown to the programmer, but each is preceded by two bytes containing its length, the same effect can be obtained by using two CCWs for each record, the first CCW reading the two-byte length into the 'byte count' field of the second CCW which reads the record itself.

(ii) Command chaining: this is requosted when the next command is not necessarily the same as the one before, and, in so far as it is concerned with 'blocking' of data on the peripheral medium, will affect a new block. This facility is particularly useful for random access devices which require sequences of different actions to be carried out in performing a single data transfer.

A (single) Transfer in Channel CCW may intervene between two chained CCWs, so that the latter do not have to be adjacent in the store.

All main store addresses specified in the CAW and in CCWs are absolute, and are not subject to relocation either by the Relocation Base Address or by the 4-75's Paging mechanism.

6.2.2 Channel Address Word (CAW)

The Channel Address Word occupies bytes 72-75 of main store. It must be loaded prior to the execution of Start Device with the protection key to be used for the duration of the peripheral operation, and the address of the first CCW. This address must be a multiple of 8 and must refer to a valid store location, or the peripheral operation will be suppressed.

As soon as the Start Device instruction has been completed (i.e. the next instruction in sequence is initiated) the Channel Address Word is available for overwriting.

The format of the Channel Address Word is as follows:

Key	0000	Address of first CCW
0	4	8 31

Bits 4-7 should be zeros.

6.2.3 Channel Command Words (CCW)

Channel Command Words occupy 8 consecutive bytes, the address of the first byte being a multiple of 8. CCWs may be placed in any valid store locations; the first one, at least, must have been loaded prior to the execution of Start Device. It is possible for subsequent CCWs to be formed by the action of the peripheral operation itself; thus, a 'read' action initiated by one CCW may cause data from the peripheral medium to wholly or partly overwrite the locations occupied by the next CCW. However CCWs are not altered by the sub-channel itself as a consequence of being executed.

The format of a CCW (not Transfer in Channel) is as follows:-

Command Code	Data Address

0 8 31

Flags	000	(Not used)	Byte Count

32 37 40 48 63

The format of a Transfer in Channel CCW is as follows:-

XXXX1001	Address of next CCW

0 8 31

Ignored

32 63

The significance of the various fields in a CCW which does not specify Transfer In Channel are as follows:-

(i) Command Code (8 bits): this indicates to the sub-channel the action which is to be performed. When data chaining takes place over a string of CCWs, the code specified in the first CCW of the string is applied to all other CCWs in the string, and the Command Code fields of the latter are ignored.

Command Codes are listed in Section 6.2.6.

(ii) Data Address (24 bits): this is the absolute main store address of the first byte of the data area, i.e. the block of consecutive locations involved in the action, into or out of which information will usually be transferred. When the Command Code specifies 'Read Reverse' the address is that of the rightmost byte of the data area; for all other codes the address is that of the leftmost byte.

If the action involves overwriting the data area, the storage key of the bytes overwritten must agree with the key specified in the CAW, unless one or both keys are zero.

(iii) Flags (5 bits): the flag bits, described in detail in Section 6.2.7 below, are used to supply additional information to the sub-channel, affecting command execution and what happens when the action of the CCW is complete. Two of the flag bits, 'Chain Data' and 'Chain Command', specify whether data or command chaining is required, and another, 'Suppress Length Indicator', also has an important bearing on the termination of the action. If neither data nor command chaining is specified, the operation terminates at the completion of this command.

(iv) Byte Count: this specifies the number of bytes of information to be transferred in performing the action. The sub-channel decrements the value specified in the Byte Count field in step with augmenting (or decrementing, in the case of Read Reverse) the Data Address (though neither of these fields in the CCW itself is altered). When the count has reached zero, the action is complete.

Contrary to IBM practice, an initial count of 'all zeros' is interpreted as 65,536.

If the peripheral device signals an 'end' condition (e.g. the end of a block on a magnetic medium) before the count has expired, the action is terminated. If the Chain Data flag is set, the whole peripheral operation is also terminated with an error condition indicated. If the Chain Command flag is set, the operation is terminated only if the Suppress Length Indicator flag is not set, again with an error condition indicated; otherwise, normal command chaining takes place. Should the device reach the 'end' condition simultaneously with the expiry of the count, i.e. as a consequence of the transfer of the last byte, the sub-channel will respond to the count expiry, and perform data chaining if requested, before recognising the device 'end' condition.

The 'unexpired portion' of the byte count (i.e. the difference between the value specified in the CCW and the number of bytes actually transferred) of the last CCW executed is presented as part of the status information with peripheral interrupts associated with the normal or abnormal termination of peripheral operations.

Contrary to IBM 360 practice, the fact that the expiry of a byte count may not simultaneously cause the device to reach an 'end' condition is not regarded as an error condition, even when data chaining is not specified. In these circumstances the sub-channel must wait for the device 'end' condition to be reported before initiating command chaining, if requested, or terminating the operation.

6.2.4 Termination of Peripheral Operations

A peripheral operation may be terminated for one of three possible reasons:-

(i) Normal termination. When no errors have been detected in the course of the operation, normal termination occurs on the completion of an action specified by a CCW with neither of the chaining flags set.

This is in accord with IBM 360 practice. However in 4-40 and 4-50, normal termination can also occur when a device 'end' condition is reported before the byte count expires in a CCW having both the Chain Data and Suppress Length Indicator flags set. In 4-70 and IBM 360, this situation still causes the operation to be terminated, but, since the Suppress Length Indicator flag is ignored when the Chain Data flag is set, the termination is accompanied by the 'incorrect length' error indication.

(ii) Error termination. When an error is detected by the sub-channel in the course of a peripheral operation, the latter is terminated - usually at once, but in some cases not until the end of the current block on the peripheral medium. The error conditions are reported in the status information stored when the interrupt associated with the termination is executed.

(iii) As a consequence of 'Halt Device'. The Central Control Unit may execute the instruction 'Halt Device' to terminate a specified peripheral operation; the latter is terminated at the earliest possible moment. As with other forms of termination, an interrupt is executed; the status information stored with the interrupt gives no indication of the cause of termination.

6.2.5 Transfer in Channel

When the least significant 4 bits of the Command Code field of a CCW contain 1001, the command is 'Transfer in Channel'. The Data Address field of the CCW is used to specify the address of the next CCW (it must contain a multiple of 8), and the second word of the CCW is ignored altogether.

Two Transfer in Channel commands may not be executed in succession, and the first command executed after Start Device may not be a Transfer in Channel - i.e. neither the address in the CAW, nor the address specified in a Transfer in Channel CCW, may itself designate a Transfer in Channel CCW.

A Transfer in Channel CCW may follow immediately after a CCW with its Chain Data or Chain command flag set - the chaining, of the type specified, is to the CCW designated by the Transfer in Channel CCW, as if the latter had not intervened.

6.2.6 Command Codes

6.2.6.1 Introduction

All eight bits of the command code are transferred to the peripheral device concerned, to be interpreted by the device control unit. For the detailed interpretations of commands the Product Specifications of the appropriate devices should be consulted. As far as the sub-channel is concerned, commands (other than Transfer in Channel) are grouped into six classes, determined by the least significant 3 or 4 bits of the command code, as follows ("M" represents a modifier bit which may have significance to the device control unit, but not to the sub-channel):-

Command Code	Command
M M M M 0 0 0 1	Sense
M M M M M 0 1 0	Read Reverse
M M M M M 0 1 1	Write
M M M M M 1 0 0	Write Erase
M M M M M 1 0 1	Read
M M M M M 1 1 1	Write Control

These actions are described in the following sections. (NB: IBM 360 does not provide 'Write Erase' and its command codes differ).

6.2.6.2 Sense

This causes the transfer of 'secondary status information' from the device control unit (not the peripheral medium) to the data area in main store. The format of this information is specified in the Product Specifications of the devices concerned. Parity is not checked during the action.

6.2.6.3 Read Reverse

Information is transferred from the peripheral medium to the data area in main store while the former is passing in the reverse direction. The data address specified is that of the rightmost byte of the data area, and it is decremented during the transfer.

6.2.6.4 Write

Information is transferred from the data area in main store to the peripheral medium.

6.2.6.5 Write Erase

The action is as for 'Write' - information is extracted from the data area in main store, but instead of being written to the peripheral medium is 'lost', and the specified number of bytes on the latter are erased. Only applicable to magnetic tape devices. Errors due to incorrect accesses to the main store are therefore still liable to occur.

6.2.6.6 Read

Information is transferred from the peripheral medium to the data area in main store.

6.2.6.7 Write Control

Control information is transferred from the data area in main store to the device control unit. The response of the latter is described in the relevant Product Specifications. The device control unit does not check the parity of the bytes it receives, though main store parity error may occur.

6.2.7 Flags

6.2.7.1 Introduction

The five flag bits occupy bits 32-36 of the CCW. A flag is said to be 'set' if the corresponding bit is a 1. The flag bits have the following significance:-

Bit Number	Flag
32	Chain Data
33	Chain Command
34	Suppress Length Indicator
35	Skip Data
36	Program Controlled Interrupt

6.2.7.2 Chain Data

The Chain Data flag is set when data chaining is required on completion of the action specified by the CCW. When the byte count expires, and the Chain Data flag is set, the action of the current CCW is continued by the next one, or, if the next command is Transfer in Channel, by the CCW designated by the latter. The Command Code field of the 'continuation' CCW is ignored. The device control unit is unaware of the occurrence of data chaining; the same block on the peripheral medium is operated on. It is essential that the sub-channel should complete the chaining process (including execution of Transfer in Channel if specified) in time to service the next byte in the current block.

If the Chain Data flag is set and the peripheral device reports an 'end' condition before the byte count has expired, data chaining is suppressed and the peripheral operation is terminated, with an error indication. If the 'end' condition occurs as a consequence of transferring the byte which causes the byte count to expire, it will not be reported to the sub-channel until data chaining has taken place.

6.2.7.3 Chain Command

The Chain Command flag is set when command chaining is required after the action specified by the CCW has been terminated and the 'end' condition signalled by the device control unit (as described in the appropriate Product Specifications). The flag is ignored if the Chain Data flag is set.

If the byte count expires before the 'end' condition is reported, command chaining is delayed until the 'end' condition occurs. If the 'end' condition is reported before the count expires, command chaining occurs only if the Suppress Length Indicator flag is also set; otherwise error termination of the operation occurs.

The next CCW (which may specify Transfer in Channel) is taken from the double-word adjacent to the current CCW unless the 'Status Modifier' condition is reported by the device control unit, in which case these 8 bytes are ignored and the CCW taken from the next double-word, whose address is therefore 16 greater than that of the CCW which specified command chaining.

If neither the Chain Data nor Chain Command flags is set in the CCW, normal termination occurs at the completion of the action, assuming no errors have occurred.

6.2.7.4 Suppress Length Indicator

When the Suppress Length Indicator flag is set in a CCW, and the Chain Data flag is not set, the fact that a device 'end' condition has been reported before the expiry of the CCW's byte count is not regarded as an error condition. This flag is ignored if the Chain Data flag is set. The effect of this in relation to chaining is described in the preceding section. When chaining is not requested, premature occurrence of the device 'end' condition causes error termination instead of normal termination unless this flag is set.

6.2.7.5 Skip Data

When the Skip Data flag is set in a CCW executing a Read, Sense, or Read Reverse command, the information read from the peripheral medium or device control unit is not written into the main store; the data area is therefore not altered. However, the data address is updated and checked for all types of error, including protection; so the data area must be as 'valid' as it would have to be were this flag not set.

In other types of command the flag is ignored.

Used in conjunction with data chaining, this flag allows parts of an input block to be ignored ('skipped'), and other parts to be read into store.

6.2.7.6 Program Controlled Interrupt (PCI)

The setting of this bit in a CCW causes the sub-channel to request a peripheral interrupt after it has staticised, i.e. started to execute, that CCW. Depending on the type of channel hardware involved, the interrupt may be requested either immediately after staticising, or not until the first byte of the transfer specified by the CCW has been passed. When the interrupt is actually executed will depend on the priority of competing peripheral interrupt requests, and it may not take effect until some time after being requested. The course of the peripheral operation is not affected.

The cause of the interruption is specified as part of the status information stored when it is executed. The purpose of such interruptions is to advise the Central Control Unit of the fact that a peripheral operation has reached a certain stage.

6.3 Peripheral Interrupts

Peripheral interrupts are of three types:-

(i) Termination interrupts (described in Section 6.2.4)

(ii) Program Controlled Interrupts (described in Section 6.2.7.6)

(iii) 'Manual Request' interrupts. These are caused by peripheral devices as the result of operator or other external action, not by sub-channel operations.

All peripheral interrupts cause status information (the 'Channel Status Word', described in Section 6.4) to be placed automatically in standard locations in main store when executed, and identify the device concerned by the weight placed in the P3 weight register (see Section 5.2) and the device number included in the status information. Because all channels use the same store status area, all further peripheral interrupts must be masked, as soon as any one is executed, until this area of store is available for overwriting again.

6.4 Status Reporting

6.4.1 Channel Status Word (CSW)

Whenever a peripheral interruption occurs (refer to Section 6.3), and sometimes in the course of executing the instructions Start Device, Test Device or Halt Device, status information relating to the device and channel concerned are placed automatically in some or all of main store absolute locations 88-95. These locations are referred to collectively as the 'Channel Status Word' (CSW).

The setting of the Condition Code to 1 at the end of the instruction indicates that the Channel Status Word has been stored during the execution of Start Device, Test Device, or Halt Device. During the execution of these instructions, all peripheral interrupts must remain masked until there is no danger of status information being overwritten.

Sometimes a condition arises in a device which cannot be indicated fully in the CSW. In such cases a bit is set in the CSW indicating that secondary information should be obtained from the device control unit by the execution of a 'Sense' command (see Section 6.2.6.2).

6.4.2 Format of CSW

The CSW is laid out as follows:-

(Byte 88)	(Bytes 89-91)
Device No.	Address of next CCW
0	8 31

(Byte 92)	(Byte 93)	(Bytes 93-4)
SDB	CSB	Residual byte count
32	40	48 63

Bytes 88-91 (bits 0-31) are only altered in conjunction with peripheral interrupts.

The significance of the fields is as follows:-

(i) Device number (8 bits). This indicates the number (on the channel - see Section 6.1.2.4) of the device involved in the interruption.

(ii) CCW address (24 bits). This is the absolute address of the next CCW to be staticised by the sub-channel when a termination interrupt or PCI occurs, and is usually 8 bytes greater than the address of the last CCW executed.

(iii) SDB (8 bits).- Standard Device Byte. This is described in Section 6.4.3.

(iv) CSB (8 bits) - Channel Status Byte. This is described in Section 6.4.4.

(v) Residual Byte Count (16 bits). Following a termination interrupt, this indicates the difference between the specified value of the Byte Count of the last CCW executed, and the actual number of bytes transferred.

In the descriptions of the Standard Device Byte and the Channel Status Byte which follow,,a bit is said to be 'set' or 'present' if it is a 1.

6.4.3 Standard Device Byte (SDB)

6.4.3.1 Introduction

The Standard Device Byte is stored in location 92. The contents of the byte are used to record conditions associated with a particular peripheral device and its control unit.

The bits of the SDB are assigned as follows:-

Bit (of CSW)	Abbreviation	Condition
32	MR	Manual Request
33	TIP	Termination Interrupt Pending
34	DB	Device Busy
35	CB	Device Control Unit Busy
36	DE	Device End
37	SI	Secondary Indicator
38	INOP	Device Inoperable
39	SM	Status Modifier

When stored in conjunction with a peripheral interruption, only the MR, SI and INOP bits are significant. With Program Controlled (not termination or Manual Request) Interrupts, the SDB may be ignored. Other bits are significant when stored in the course of Start Device, Halt Device and Test Device instructions. The conditions associated with some of the bits are described in greater detail in the Product Specifications of the devices concerned.

6.4.3.2 Manual Request (MR)

The presence of this bit with a peripheral interruption indicates that the device concerned requires attention by the Central Control Unit. This condition is generated by means either of a manual switch or of signals transmitted over a data transmission or data exchange system. The appropriate Product Specifications may be consulted for further details.

The condition can only be generated at a time when none of the following oonditions are present in the device or its control unit:-

Interrupt Pending (TIP)
Device Busy (DB)
Control Unit Busy (CB)

The condition is cleared when the interrupt occurs. If CC = 1 following a Start Device or Test Device, and MR = 1 in the stored SDB, it should <u>not</u> be assumed that the Manual Request condition exists in the addressed device. It may exist in a device attached to the same sub-channel or device control unit. In these circumstances, the condition is not cleared, and the identity of the device will be discovered when the interrupt occurs.

6.4.3.3 Termination Interrupt Pending (TIP)

This bit is associated with the condition which exists in the device and control unit prior to the execution of a termination interrupt. The condition is cleared by the storage of the CSW at the time of the interrupt; its presence in the SDB should be ignored.

The presence of this condition prevents the initiation of another peripheral operation on the same sub-channel.

6.4.3.4 Device Busy (DB)

This bit is associated with the condition which exists when the peripheral device (but not necessarily its control unit) is busy.

The presence of this condition inhibits the initiation of a peripheral operation on this device.

6.4.3.5 Control Busy (CB)

This bit is associated with the 'busy' condition in a device control unit. This condition inhibits the initiation of a peripheral operation involving this particular control unit.

6.4.3.6 Device End (DE)

This bit is associated with the condition of a peripheral device in between peripheral actions. If the device is not in this condition a new peripheral operation cannot be initiated, unless the first command is Sense.

6.4.3.7 Secondary Indicator (SI)

The presence of the SI bit in the SDB indicates the presence of secondary status information in the device control unit. This information is usually generated as a result of error conditions detected by the control unit (e.g. an unacceptable command code, parity error), and is described in detail in the relevant Product Specifications.

In this situation the only command accepted by the device is a Sense command, which causes the secondary information to be read into the store (see Section 6.2.6.2); at the end of the Sense operation this bit will be reset to zero.

The presence of secondary status information will cause a peripheral operation on the device concerned to terminate (with the SI bit set in the SDB when the termination interrupt occurs) when the device next reaches an 'end' condition; i.e. command, but not data, chaining, is inhibited.

6.4.3.8 Device Inoperable (INOP)

This bit is associated with an 'inoperable' condition in the device. Should a device become inoperable in the course of an operation, the latter is terminated. This condition will inhibit the initiation of a new peripheral operation unless the first command is Sense.

6.4.3.9 Status Modifier (SM)

The Status Modifier bit is associated with conditions which arise in peripheral devices in certain circumstances (detailed in the appropriate Product Specifications) and, when command chaining occurs, cause the sub-channel to augment the address of the next CCW to be executed by 8 bits, thus skipping the next CCW in sequence.

The presence of the bit in the SDB may be ignored.

6.4.4 Channel Status Byte (CSB)

6.4.4.1 Introduction

The Channel Status Byte is stored in location 93. The contents of this byte are used to record conditions associated with the channel/sub-channel controlling a peripheral operation, rather than with the device and its control unit.

These conditions are all liable to arise in the course of a peripheral operation; some are error conditions, causing the operation to terminate prematurely. Some may arise during the execution of Start Device, causing the operation to be suppressed. The error conditions, apart from Channel Control Check (see Section 6.4.4.7), should be ignored except after Start Device or termination interrupts.

The bits of the CSB areassigned as follows:-

Bit (of CSW)	Abbreviation	Condition
40	PCI	Program Controlled Interruption
41	* IL	Incorrect Length
42	* PGC	Program Check
43	* PTC	Protection Check
44	* CDC	Channel Data Check
45	* CCC	Channel Control Check
46	CTIP	Termination Interrupt Pending
47	CTI	Termination Interrupt

(* indicates an error condition)

The bits in the CSB must be inspected by the Supervisor following a Start Device instruction when CC = 1 indicates that the CSW has been altered, and also following a peripheral interrupt which is not a Manual Request (see Section 6.4.3.2). In the former case, only conditions actually arising during the execution of Start Device are reported.

6.4.4.2 Program Controlled Interrupt

The presence of this bit in the CSB stored with a peripheral interrupt indicates that a PCI flag was set in one of the CCW's executed in the associated peripheral operation.

Should a subsequent CCW of the same peripheral operation be executed, having its PCI flag set, before the interrupts requested by PCI flags in one or more preceding CCW's have been executed, only one interrupt will occur; all outstanding PCI requests will be cleared by that interruption. However if another PCI flag is detected after the execution of the interrupt, a further interrupt will be requested. Should there be one or more outstanding PCI requests at the termination of the peripheral operation, the termination interrupt will clear them; in this case the PCI bit will be present in the CSB stored when the termination interrupt is executed.

The information stored in the CSW when a PCI interrupt is executed reflects the state of the peripheral operation at the time of interruption, rather than when the PCI flag was first, or subsequently, detected.

6.4.4.3 Incorrect Length (IL)

When the I^L bit is set in the CSB accompanying a termination interrupt, it indicates that the peripheral operation was prematurely terminated because a device 'end' condition was reported to the sub-channel before the expiry of the byte count in a CCW which has either its Chain Data flag set, or its Suppress Length Indicator flag not set. Premature occurrence of a device 'end' condition can only be prevented from causing the 'incorrect length' error if the Chain Data flag is not set and the Suppress Length Indicator flag is set (but if the CCW had an initial Byte Count of 'all zeros' and no bytes had been transferred, IL would not be set provided only that the Chain Data flag was not set).

Note that a device 'end' condition may be generated because of some other error condition (e.g. Protection Check) arising in the middle of a peripheral block, and that in such cases the IL bit may be set in the CSB in addition to other error bits in the CSB or SDB. It may also be set following termination by Halt Device, even if the Suppress Length Indicator flag was set in the last CCW.

6.4.4.4 Program Check (PGC)

When the PGC is found to be set in the CSB accompanying a termination interrupt, or stored during the execution of Start Device, it indicates that the peripheral operation concerned was terminated or suppressed because of a program-originated error. The conditions which give rise to this error are:-

(If CSB stored during Start Device)

- CAW specifies a CCW address in a non-existent area of store.

- CAW specifies a CCW address which is not a multiple of 8.

- The command code of the CCW designated by the CAW contains in its least significant 4 bits one of the patterns 1001 (Transfer in Channel), 0000, 1000, 0110, or 1110 (unrecognised codes).

(If CSB accompanies termination interrupt).

- CCW specifies a data area in a non-existent area of store (regardless of whether or not the Skip flag is set).

- A Read, Read Reverse or Sense command attempts to overwrite reserved store locations (even if Skip flag is set). See Section 3.5.

- Address of next CCW specified by a Transfer in Channel Command is in a non-existent area of store.

- Address of next CCW specified by a Transfer in Channel Command is not a multiple of 8.

- A Transfer in Channel command follows immediately after another Transfer in Channel (the address given in the CSW in this case is that of the second Transfer in Channel CCW).

- Command Code of a CCW contains one of the patterns 0000, 1000, 0110 or 1110 in its least significant 4 bits.

6.4.4.5 Protection Check (PTC)

This bit is set following error termination of a peripheral operation due to an attempt having been made to overwrite a main store location whose key differs from that specified in the CAW, both being non-zero. Note that this error may occur even if the Skip flag is set in the relevant CCW.

6.4.4.6 Channel Data Check (CDC)

The presence of this bit in the CSB accompanying a termination interrupt indicates that a parity error was detected on a byte being transferred between main store and peripheral device during the operation which was terminated.

Parity checking is not performed during a Sense command, which cannot therefore give rise to this error. Parity is not checked by the device control unit during a Write Control command, but is checked as the information leaves the store. Parity checking still occurs during a Write Erase command, but not when the Skip flag is set in a Read or Read Reverse command.

The occurrence of a parity failure does not cause the sub-channel to stop a peripheral operation immediately. Normally the operation will proceed until the next device 'end' condition occurs, i.e. command chaining is inhibited, but data chaining proceeds normally.

When the parity error is detected during an output operation, the erroneous byte is transmitted unaltered. When the device control unit detects the error condition, it may cause the operation to be terminated immediately by generating an 'end' condition, and/or set its Secondary Indicator bit (see Section 6.4.3.7); the action of the device control unit is detailed in the appropriate Product Specifications.

When the parity error is detected during an input operation, each incorrect byte is replaced in the main store by an 'all ones' byte (except when the Skip flag is set).

6.4.4.7 Channel Control Check (CCC)

The presence of this bit indicates that the parity of the CAW or a CCW was erroneous; this condition terminates the operation immediately. When this bit is set, other information in the Channel Status Word should be treated as suspect. It may be set following the execution of Start Device (CAW or first CCW parity error) or in conjunction with a termination or PCI interrupt.

6.4.4.8 Channel Termination Interrupt Pending (CTIP)

The presence of this bit in the CSB may be ignored. It is associated with the condition which exists in a channel prior to the execution of a termination interrupt.

6.4.4.9 Channel Termination Interrupt

The presence of this bit in the CSB stored on execution of a peripheral interruption indicates that the interruption was due to the termination of a peripheral operation (see Section 6.3).

6.4.5 Validity of CSW

The validity of the information stored in the Channel Status Word depends on the circumstances leading to its storage.

When status information is stored during a Start Device, Test Device or Halt Device instruction (indicated specifically by the Condition Code being set to 1) the first half of the Channel Status Word is not altered, and the Residual Byte Count field should also be ignored. Only the Standard Device Byte and Channel Status Byte are relevant, the latter being significant only after Start Device, when it indicates conditions generated in the course of the instruction - all conditions associated with CSB bits are effectively cleared at the beginning of the instruction. Apart from this, the CSB only contains valid information after the execution of Halt Device on a selector channel, and may be ignored anyway in this case.

The conditions giving rise to the different settings of the Condition Code by the four peripheral instructions are described in Appendix D.2. 1-4.

When a peripheral interrupt occurs, the whole Channel Status Word is overwritten, though in some cases the contents of various fields should be ignored. In the table below are listed the various conditions giving rise to peripheral interrupts, with indications (V = Valid, U = Validity uncertain, X = ignore - invalid or not meaningful) of the significance of the fields.

	Device	CCW	X		Residual
Interrupt	Number	Address	SDB	CSB	Count
1. Manual Request (MR = 1)	V	X	V	X	X
2. Program Controlled Interrupt (PCI = 1, CTI = 0)	V	V	V	V	X
3. Normal termination	V	V	V	V	V
4. Error termination					
(a) CCC = 1	V	X	V	V	X
(b) PGC or PTC = 1	V	U	V	V	X
(c) CDC, IL, SI or INOP = 1; or after Halt Device	V	V	V	V	V

The validity of the CCW address is uncertain in case 4(b) because the time at which the information is stored in uncertain; it is valid at the time it is stored.

Should certain types of hardware failure occur, a Channel Status Word may be stored in conjunction with an interrupt in which the SDB and CSB are 'all zeros'. In this case all the other fields should be ignored.

6.4.6 Analysis of SDB and CSB after Interrupt

The steps in which the SDB and CSB should be analysed following a peripheral interrupt are as follows (at each step it is assumed that the conditions of none of the previous steps apply).

Step	Indications	Interpretation
1	MR = 1	Manual Request Interrupt (occurs only when the sub-channel is not involved in a peripheral operation).
2	CCC = 1	Because of parity failure in control information, other status information is suspect.

Step	Indications	Interpretation
3	PCI = 1, CTI = 0	Program Controlled Interrupt. The progress of the operation may be determined from the Next CCW Address field. Note that, even if the Channel concerned is of the type in which a PCI is not requested until the servicing of the first byte of the transfer specified by the relevant CCW, a PCI requested by an earlier CCW may be executed before that time and still appear to originate from the later CCW.
4	CTI = 1 IL) PGC) PTC) CDC) all = 0 CCC) MR) SI) INOP) TIP) DB) CB) not all = 0 DE)	Normal termination (after termination by Halt Device, IL may = 1; and after a Sense command INOP may = 1, but secondary status information will still have been transferred correctly) Note: If TIP, DB, CB, DE, INOP all = 0, a device error condition is indicated which will cause subsequent Start Device (and Test Device and Halt Device) instructions to generate CC = 3, and hence inhibit future operations, including Sense commands.

5	PTC = 1 or
	PCC = 1
		Program error in CCWs.

6	CDC = 1	Data parity error.
7	IL = 1	Count error ('short block')
8	SI = 1	Error has occurred; secondary status information available from device, using Sense command (but see note under step 4).

9	SI = 0 and
	INOP = 1
		Device error has occurred which cannot be examined using a Sense command.

10	SDB = all zeros
	CSB = all zeros
		Hardware error.

The analysis of the SDB and CSB to be performed after peripheral instructions which have set CC = 1 is described in Appendix D.2 1-3.

6.5 Connection of Peripheral Devices

6.5.1 The Standard Interface

The term 'Standard Interface' refers to the logical and physical 'plug and socket' arrangement by which every peripheral device control unit is connected to a System 4 processor. The interface is 'standard' because each device control unit is designed to have a standard 'plug', and the same sort of matching 'socket' is provided by all channels on all System 4 processors. Thus (subject to performance limitations, and certain other restrictions) any device control unit may be attached to any channel on any processor, and any channel can accept all the different types of device control unit. A device control unit may control one or more devices, all of which are therefore connected to the processor via the interface to which the control unit is attached.

The System 4 standard interface is one byte wide, which is to say that it contains sufficient lines to enable one byte of information, with parity, to be transferred across it at a time. This applies to control information as well as data. The standard interface is used in somewhat different fashions by selector and multiplexor channels. The essential difference is that whereas on a multiplexor channel each byte of information transferred across the interface is preceded by another byte identifying the peripheral device concerned, on a selector channel (on which only one peripheral device can be engaged in a peripheral operation at a time) the device identification byte is not required.

Most peripheral devices are capable of operating on either type of channel. However, random access devices must usually be attached to a selector channel, and remote communications devices to a multiplexor channel.

6.5.2 Trunks

A 'trunk' is a physical unit, associated with a channel, which provides a single standard interface 'socket' for the attachment of one device control unit. One channel may have several trunks. Trunks, like sub-channels, are not explicitly addressable, but they should not be confused with the latter. For instance, whereas a selector channel may have several trunks but has only one sub-channel, on a multiplexor channel the number of sub-channels is the same as the number of devices, which is never less than the number of device control units (i.e. of trunks).

A trunk is a physical entity whereas a sub-channel is a logical entity.

6.6 4-70 Channel Control Units

6.6.1 Introduction

The physical implementation of channels and sub-channels on 4-70/75 is by units called Channel Control Units (CCUs). There are three different types of CCU. They differ in the types of channel they implement (selector or multiplexor), in their performance, and in the numbers of channels, sub-channels and trunks they can provide.

Each Channel Control Unit is an autonomous unit, capable of operating in parallel with other CCUs and with the Central Control Unit, There may be up to 6 CCUs on a single 4-70. Each CCU has its own 'Accessway' to the Input-Output Store Multiplexor (see Section 3.2.1.2). CCUs are not explicitly addressable, and the channels they control may be arbitrarily numbered.

The three different types of CCU are called:-

Single Channel Control Unit (SCCU)
Multichannel Control Unit (MCCU)
Multiplexor Channel Control Unit (MXCU)

The performance capabilities and rules for connection of CCUs are detailed in Appendix C. The characteristics of the different types of CCU are summarised in the table below:-

	SCCU	MCCU	MXCU
Channel Type	Selector	Selector	Multiplexor
Maximum number of channels	1	8	1
Maximum number of sub-channels	1	8 (1 per channel)	256 (1 per device)
Maximum number of trunks	4	8 (1 per channel)	16

6.6.2 Priorities

Two types of priority need to be considered; relative priority for servicing different sub-channels during concurrent peripheral operations, and relative priority for execution of interrupt requests from different devices.

Priority of service for channels is determined firstly by the priority of the CCUs involved (this depends on the particular Accessways to which they are connected); then by the relative priorities of the different channels on the same CCU (this is independent of the channel numbering). Priority between trunks on the same channel is determined by the channel hardware, and between devices attached to the same trunk by the device control unit.

Interrupt priority between channels is determined by the channel numbering, channel 0 having highest priority. Priority between devices on the same channel is determined on the same basis as governs the relative priorities for service, described above.

6.6.3 Storage of Control Information

Each CCU is responsible for updating and storing the control information which governs the operation of its sub-channels. Each is provided with internal registers for updating the control information for one sub-channel. The SCCU, having only one sub-channel, can retain the control information in these registers all the time (and hence attain its high level of performance). The MCCU and MXCU hold the control information for all their sub-channels in, respectively, a private scratchpad store, and the reserved area of main store (see Section 3.5).

For each device on an MXCU, 16 consecutive bytes are reserved. The absolute address of the first byte for any device is evaluated by multiplying the device number (0-255) by 16 and adding a base address which is either 0 or 4096. The same base address is used for all the devices on one MXCU. If there are two MXCUs, either they may use the same base address (i.e. share the same 4096-byte block of reserved store locations), in which case different devices on the MXCUs must have different device numbers; or one may use base address 0, and the other 4096. If there are three MXCUs (the maximum), two of them at least must use the same base address.

An MXCU using base address 0 cannot have devices numbered in the range 0-5 since the reserved store locations start at 96. The last reserved location available may be location 1023, 2047, 4095 or 8191, allowing 58, 122, 250 or 506 multiplexor devices altogether.

The 16 bytes belonging to a particular multiplexor device are used to hold the following information during a peripheral operation, while it is not actually being updated:-

Byte 0	:	Device number
Bytes 1-3	:	Address of next CCW
Byte 4	:	(Bits 0-3) Key (from CAW) (Bits 4-7) Least significant 4 bits of command code
Bytes 5-7	:	Current data address
Byte 8	:	(Bits 0-4) Flags
Byte 9	:	Channel Status
Bytes 10-11	:	Current byte count
Bytes 12-15	:	Not used

6.6.4 Miscellaneous Features of CCUs

6.6.4.1 Trunk Cable Lengths

Trunks may be up to 100 cable feet long, except on the SCCU when they must be between 25 and 75 cable feet.

6.6.4.2 PCI Requests

Sub-channels of the SCCU and MXCU do not request Program Controlled Interrupts until servicing the first byte of a transfer specified by a CCW in which the PCI flag is set. Sub-channels of the MCCU, on the other hand, request PCI in the process of chaining to a new CCW which has its PCI flag set.

6.6.4.3 MCCU Diagnose Facility

A special diagnostic facility is available on the MCCU when it is addressed after the execution of a 'Diagnose' instruction. This facility is described in Appendix D.2.9.2.

6.6.4.4 Miniburst Mode on MXCU

The operation of a multiplexor channel alters if, after servicing a byte, the device concerned is found to be still requesting service. In this case the sub-channel control information is not returned to the reserved part of main store but is retained in the registers of the MXCU until the next byte has been serviced; and so on. Thus, if successive services are requested by a device at a sufficient rate, the MXCU behaves as if it had only one sub-channel. This behaviour cannot be specified by program.

7. INITIAL PROGRAM LOADING

7.1 Introduction

In this section is described the method by which an operator initiates the operation of the processor.

7.2 Operator Instigated Initial Program Loading

7.2.1 General

An operator initiates the operation of a processor by executing a sequence of actions which involve a peripheral device and the Operators Control Panel. The functions of each of the controls and indicators on the Operators Control Panel is specified in Section 10.

7.2.2 System Reset

The first stage in the initial program load sequence is the pressing of the System Reset Control. This action causes the Central Control Unit to clear the following registers:-

Condition Code Register
Program Mask Register
Sequence Control Counter
Interrupt State Identifier
Store Protection Key
Operation Mode Indicators
Relocation Base Register

The Central Control Unit then initiates Processor State 3, and halts all its operations. In addition the channels and peripheral devices are also reset. The system then waits for the action of the Load Control.

7.2.3 The Program Load Sequence

Activation of the Load Control on an Operators Console causes the Associated Central Control Unit to initiate the 'initial program load sequence'. Before this control is pressed, the address of the peripheral device, which is to read the 'initial program load sequence', must be set up on the Load Device Switches. The first hexadecimal digit specifies the channel number, the next two the device number on that channel.

The action of the Program Load Sequence is as follows:-

1. The Channel Address Word is cleared and as a result specifies a Channel Command Word at address 0 and a Zero Protection Key.

2. The second word in store (locations 4 to 7) is filled with the second half of a Channel Command Word. This word specifies a 24 byte command chained transfer.

7.2.3 Continued.

3. The first word in store (location 0 to 3) is filled with the first half of a Channel Command Word. The command specified by this word is 'Read' and the address '0'.

4. The action of a 'Start Device' instruction is performed; the device address is then taken from the Load Device Switches.

5. If the addressed sub-channel is available, the action continues as for a normal Start Device, i.e. as if CC = 0. If any of the conditions associated with CC = 1, 2 or 3 (see Appendix D.2.1) occurs, the peripheral operation is suppressed and the action is halted.

6. After the first block has been read into bytes 0-23, command chaining occurs to the CCW now in bytes 8-15. Operations in the channel continue in the normal way until chaining is complete. Note that information may only be read into the Reserved Store (0 -   ) until the first channel interrupt occurs.

7. At the end of the transfer, the Channel interrupt is executed and status information is stored in the Channel Status Word. If any errors are indicated (see 7.2.4 below) the action is halted, otherwise the action continues and the appropriate Weight is placed in the Weight Register of Processor State 3. *

8. The central control registers associated with the Program Counter Register are loaded from the information stored in the first word of store. Those registers which are associated with the Interrupt Status Register are loaded from the second word of store.

9. Normal operations commence by staticising the instruction specified by the Sequence Control Couter.

7.2.4 Fault Detection

Faults detected in stage 5 and 7 cause the sequence to be halted. The faults detected in stage 5 are those associated with the execution of a 'Start Device' instruction.

The faults in stage 7 are detected by checking to see if any of the following conditions are true.

CCC = 1 CDC = 1 PTC = 1 PGC = 1

PCI = 1 CTI = 0 MR = 1 SI = 1 INOP = 1 TIP = 0 DE = 0

7.2.5 Format of First Record

The format of the first 24 byte record read into store must be as follows.

7.2.5 (continued)

Word	Allocation
0	Program Counter
1	Interrupt Status Register
2	First word of second Channel Command Word
3	Second word of second Channel Command Word
4 and 5	More channel command words or as required.

7.2.6 Erroneous Initiation

A processor that is operating will attempt to perform the initiated program load sequence whenever the 'Load Control' is operated.

7.3 Processor Instigated Initial Program Loading

7.3.1 General

A 4-70 processor may either initiate the operation of a reset or 'stopped' 4-70 processor or redirect the action of a working 4-70 processor if the processors are linked by specially connected direct control lines. The method used is a modified form of the initial program load sequence and can only be successfully accomplished if the correct control words are present in the main store of the processor which is to be initiated.

7.3.2 Links

The initial program load command is carried by two lines in each direction in each direct control connection between 4-70 processors. It is thus possible for a processor to send or receive the command to or from up to six other processors. For a *
further description, refer to PS 4.6.10. *

In addition to the normal direct control link a processor which is to be initiated must have a special hardware connection made between its control system and one or other of the incoming initial program load lines. This link is not a catalogue item and must be specifically requested.

7.3.3 Operation

A processor instigates an initial program load operation in a remote processor when it executes a 'Write Direct' instruction which has the bits of its Immediate Field all set zero with the exception of one or both of the bits which select the initial program load lines (i.e. the least significant two bits of the field). Any processor linked to the selected initial program load line reacts by staticising into its control registers the contents of a Program Counter and Interrupt Storage Register previously stored in main store words 0 and 1 respectively. Processor operation commences in Processor State 3 with an inhibition on interrupt execution that lasts until the completion of the first instruction. This instruction must load the current Interrupt Status Register with the corresponding value stored in

7.3.3 (continued)

main store and thus ensure that subsequent interrupts do not cause the loss of this information.

As a side effect of the operation, the contents of main store bytes 2 and 3 are replaced with the value indicated by the Load Device Switches on the processor's console.

8. DIRECT CONTROL

Direct Control is an optional feature of the 4-70 that can be used for communication between processors and also for initiating the operation of one processor by another. (See Section 7). The instructions associated with the Direct Control feature are described in Appendix D. The function and properties of the System 4 Direct Control System are specified in PS 4.6.10.

9. ENGINEERING

9.1 Introduction

In this section are specified the processor's dimensions and power requirements. The Engineer's Control Panel is not specified as it is fully defined in the 'Engineers' Description' of the 4-70.

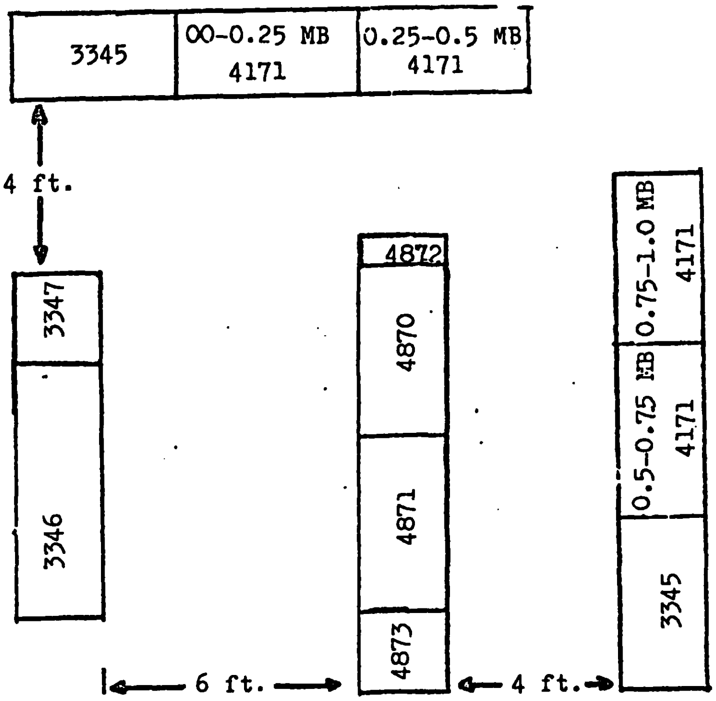
9.2 The Physical Layout of the Processor

The processor consists of from 10 to 21 cubicles which are arranged with the Central Control Unit and the Channel Control Unit cubicles surrounded on two sides by the main store and its power supplies and on the third side by the power supplies of the Central Control Unit and of the auxiliary Channel System. The engineer's control panel is attached to the Central Control Unit double cubicle.

Each cubicle which is not a power supply has a door on its front, another on its back. These doors each hold three platters on which are mounted the processors logic etc. There are in addition three platters mounted in the centre of each cubicle. The cubicles must be so positioned that the doors may be fully opened. In particular free access must be given to the door that covers the engineer's control panel.

The layout of the processor is as follows:-

Diagram 2



PROCESSOR LAYOUT

The length of the cable run from the Engineer's control panel to Operator's control panel is 75 feet.

9.3 Cubicle Requirements

Cubicles are 27½ inches deep and height from floor is 61¾ inches. Their width, weight and power requirements are as follows:-

Table 8

Item	Width inches	Weight pounds	Power KVA
Central Control Unit 4870 plus 4872	58	1,220	4.0
Channel Control Unit Cubicle 4871	54	1,120	5.3
Channel Control 4873	27	624	2.7
Main Store Cubicle 4171	54	1,120	5.2*
Power Supply Cubicles			
Central Control Unit 3346	81	1,792	21.5
Main Store 3345	54	1,232	14.0
Power Supply Expansion 3347	27	672	5.5

* 1.3 per 65 KB

10. OFERATING FACILITIES

10.1 Introduction

In this section are described the controls and indicators which are available to an operator. Engineer-only controls are described in Engineering Specification 642 4070.0011 R08 Section 5. Operator Controls are of two types: the first are those used for controlling the operation of the hardware features of the processor and are concerned with such operations as initial program loading and halting the processor; the second group of controls are those used for communicating commands to the supervisor program. The second type of control takes the form of an alphanumeric keyboard, the properties and functions of which are described in the specifications concerned with operating devices. Further information may also be found in those specifications which describe the relevant supervisor requirements.

Operator's controls are of two types; those that must be readily available, and those that are covered by a flap and are not normally used except in particular circumstances. The first type are described in the next section whilst the second are described in Section 10.4.

10.2 Operator's Control Panel

The Controls and Indicators described in this section are mounted on the Operator's Control Panel. The panel is itself mounted on the Central Processor Control Desk; this is described in PS 4.8.3. The controls are mounted such that: the Emergency Off Button is situated by itself on the left of the panel; the standard controls and indicators in the centre of the panel, with the indicators above the control buttons. The special controls and indicators are under a flap to the right of the standard controls.

10.3 Standard Controls and Indicators

10.3.1 The Emergency Off Button

Operation of this control causes instant removal of power from the entire system, prematurely terminating the current instruction but not physically damaging devices or media. Instruction sequencing stops immediately and there is no interrupt to P4 state. * *
Engineer's action is required to reset the 'emergency off' state after this control has been operated.

The control is a button of different appearance from other controls, and is located separately from them. It is bezelled and strongly sprung to obviate accidental operation.

10.3.2 Controls

10.3.2.1 Stop

If the Central Control Unit is running, actuating this control causes it to go to the 'stop' state; the Stop Indicator then lights.

10.3.2.1 (continued)

In the 'stop' state, an instruction is staticised but not yet executed. Input-Output operations continue until their natural termination is reached.

When the 'stop' state has been reached as a result of the operation of this control, running may only be resumed by operating the Start Control, or the General Reset followed by Load. *

10.3.2.2 Start

Operation of this control will cause a 'halted' Central Control Unit to recommence its operation.

10.3.2.3 Load

Operation of this control will cause the 'initial program load' sequence to be initiated. The action must be preceded by setting the Loading Device switches and by operating the General Reset Control. Further information is given in Section 7. If this control is pressed whilst the Central Control Unit is operating, the 'initial program load' sequence will be executed.

10.3.2.4. Power On/Off

This is an alternate action switch which initiates the 'power on' and 'power off' sequences of the entire installation. The switch is inoperable if the Remote Indicator is lit.

Setting this control to ON initiates a controlled run-up of power throughout the system.

After power has been switched on, the Load Control is inoperative until returns indicate that the computer is fully ready for use. When the power run-up is completed, the processor is in the 'stop' state, and running may be commenced by operation of the Load Button. The last action of the Sequencer Unit is to energise the Controls and Indicators of the centre panel of the Console Unit. Consequently, the Stop Indicator does not light until the computer is fully ready for use.

Setting the control to OFF, initiates a controlled rundown of power throughout the entire system. After the operation of the control, there will be a 10 second delay before the power goes fully off.

10.3.3 Indicators

10.3.3.1 Stop

The Stop indicator lights to indicate that the processor is in the 'Stop' state. This state is entered automatically on detection of an error (except 'Awaiting Interlock Release') or as a result of Operator or Engineer intervention (e.g. depression of STOP button). The indicator is lit after the 'power-on' sequence has terminated.

10.3.3.2 System Error

The System Error Indicator is lit on the detection of the occurrence of one or more errors of a kind which are not resolvable by program action. The lighting of this indicator, unless it occurs during 'initial program loading', indicates that engineering attention is required.

The occurrence of any of the following conditions causes the indicator to light :-

Parity error detected during a Main Store access

Parity error detected during a Scratchpad access

Parity error detected during a Microprogram access

Parity error detected during the accessing of a Store Protection Key

Parity error detected by the processor on a byte read from a peripheral. (Note: most device control units send parity corrected bytes to the processor)

Program error (see the description of the Program Check Bit in Section 6.5.10.4) detected during the execution of the 'initial program load' sequence.
Device specified as the 'initial program load' device is inoperable.

The System Error Indicator can be reset by removing the error indication using the Engineers' Control panel (unless the error was a Microprogram Error or Awaiting Interlock Release) or by pressing the General Reset Control.

10.3.3.3 Supply Warning

This indicator is lit when it is detected that the main supply is outside a statutory ± 6% tolerance but has not yet reached the level of deviation at which automatic shut down occurs.

10.3.3.4 Remote

This indicator when lit shows that power control is at the engineer's panel and that the Power On/Off Button on the operating panel is inactive.

The Remote Indicator is also a control button which when pressed causes all the indicators on the control panel to light.

10.4 Special Controls and Indicators

10.4.1 Introduction

The special controls and indicators are located on the operator's control panel under a flap. The layout of the controls and indicators is as follows :-

10.4.1 (continued)

Top row	- general reset, console interrupt
Middle row	- idle indicator; program state indicators
Bottom row	- channel address switch, device address switches.

10.4.2 General Reset Control

The General Reset Control is used to halt the actions of the processor in such a manner that it may be restarted using the 'initial program load' sequence.

Operation of the General Reset Control causes changes to occur in the Central Control Unit, in the Channel Control Units and in the device control units and peripherals attached to those channels. The following actions take place :-

The hardware central control registers listed below are cleared :-

Condition Code Register
Program Mask Register
Sequence Control Counter
Interrupt State Identifier
Protection Key
Mode Bits
Relocation Base Register

Central Control Unit operation is halted and Processor State 3 is initiated. The Channel Control Units cease operating and reset the appropriate registers.

Device Control Units and Peripheral Devices are reset in the manner described in the appropriate specifications.

The Central Control Unit waits ready to respond to a command from the 'load' button.

10.4.3 Console Interrupt

Operation of this control causes the 'Console' interrupt to be requested in the Central Control Unit.

10.4.4 Load Device Switches

These are a set of three hexadecimal digit switches. The device, which has the channel and device number address which is indicated by the setting of these switches, is the device which is set by the processor as the source of its control information during the 'initial program load' sequence.

10.4.5 Program State Indicators

These are two indicators which together form a two bit binary number that indicates the Central Control Unit's current

10.4.5 (continued)

processor state. The code is as follows:-

Code	Current State
00	Processor State 4
01	Processor State 3
10	Processor State 2
11	Processor State 1

10.4.6 Idle

This indicator lights when the Central Control Unit is executing the 'Idle' instruction.

11. RELIABILITY AND MAINTAINABILITY

11.1 Introduction

The reliability to be attained by the processor is specified in this section. The maintenance charging meter is also described.

11.2 Reliability

It is not possible to define exactly the reliability of a 4-70/75 processor in terms of its Mean Time Between Failures. Instead a table is given indicating the relative reliability of its component units. The actual reliability of a system could vary by as much as +400% and -50%.

Note : the reliability figures only include those failures which occur during normal operational periods. This means that faults which occur during special test runs with marginal conditions applied are excluded when assessing the MTBF.

Failures caused by Program, Software or operator error are also excluded.

TABLE 9

Mean Time Between Failures

Item	MTBF in hours
Central Control Unit 4870 plus 4872	480
Multiplexor Channel Control Unit 4270	1,950
Single Channel Control Unit 4271	2,250
Multichannel Control Unit 4272	1,900
Main Store Module 4171	510
Power Supply Cubicles	
Central Control Unit 3346	3,000
Main Store 3345	4,550
Power Supply Expansion 3347	9,100

11.3 Maintainability

The 4-70 processor must comply with the maintenance requirements for System 4 as outlined in PS 4.3.4. The Engineer's Control is specified in Engineering Specification 642 4070 0011 R08, Section 5. *

11.4 Maintenance Charging Meter

Each 4-70 Central Control Unit has an associated Maintenance Charging Meter. A meter has the following characteristics :

(i) The Maintenance Charging Meter is a direct reading hour meter covering the range 0-9999 hours.

(ii) The meter is located in the processor or power cabinet such that it may be easily viewed.

11.4 (continued)

(iii) The meter will record time when all the following conditions are satisfied :-

- the processor power is switched on
- the Stop indicator is not lit (i.e. the machine is in the 'go' state)
- the Engineers' Key is not in position.

(iv) The meter will not record time when :-

(a) the processor power is switched off
or (b) the processor power is switched on
the Stop indicator is lit
the Engineer's Key is not in position
or (c) the processor power is switched on
the Engineer's Key is in position
(regardless of whether the Stop indicator is lit or not).

(v) The meter will record time when the 'idle' instruction is being executed.

12. BIBLIOGRAPHY

PS 4.2.3	Hardware Catalogue
PS 4.3.1	Installation Requirements
PS 4.3.2	Engineering Requirements
PS 4.3.4	Maintenance Requirements
PS 4.6.2	Data Representation and Character Codes
PS 4.6.3	The Non Privileged Instructions of the 4-50 and 4-70 processors
PS 4.6.10	Direct Control on the 4-50 and 4-70 processors
PS 4.7.1	Standard Peripheral Interface
PS 4.7.2	Operating Requirements for the Control of Peripheral Devices
PS 4.7.3	Glossary for System 4
PS 4.8.1	Operating and Enquiry Stations
PS 4.8.2	Basic Typewriter Systems
PS 4.8.3	Control Desks for Central Processors
PS 4.8.10	Hardware Specifications

19. PAGING

19.1 Introduction

19.1.1 The 4-75

Paging is a method of dynamic address conversion and is the distinguishing feature between a 4-70 and a 4-75 processor. The relevant catalogue numbers are given in Section 1.6.

19.1.2 Purpose of Paging

The range of addresses referred to by any program which uses paging is effectively divided into 4,096-byte units called pages, each page comprising the bytes addressed from 4,096n to (4096n + 4095), (n = 0, 1, 2,.... 4095). The range of addresses used by such a program is called its virtual store, and may extend to 2^{24} bytes. The paging system allows any page of the virtual store to be contained in any physical main store block of 4096 bytes whose absolute addresses start at a multiple of 4,096, e.g. absolute locations 4096N to (4096N + 4095), by causing every program-generated address of the form 4096n + x ($0 \leq x \leq 4095$) automatically to be converted to the form 4096N + x.

Address conversion is performed at instruction execution time and is therefore truly dynamic. The conversion is effectively performed by table look-up; there is a different value of N for each n, and there is no constraint on the values of N to run consecutively. Any one or more pages of a program's virtual store may be absent from the main store, in which case the Supervisor ensures that the table entries for those values of n confain markers which cause the program to be interrupted as soon as it refers to an absent page. This interruption occurs in such a way that the Supervisor is able to bring the required page into the store, insert the appropriate value of N in the table, and cause the program to resume with the instruction which referred to the missing page.

The purpose of paging is therefore to allow a program to refer to a virtual store whose individual pages may be dynamically relocated in the main store, and incidentally to enable a program to operate in a physical main store area-smaller than its virtual store. As indicated in Section 19.1.4 below, the arrangement of conversion tables also enables different programs to share pages.

Paging operates in conjunction with the Relocation Base Address Register; thus, if the value in the latter is R, and the virtual address generated by a program is 4096n' + x ($0 \leq x \leq 4095$) the address presented to the paging mechanism for conversion is actually 4096(n' + R) + x, so that the conversion tables actually give the values of N corresponding to n where n = n' + R (modulo 4096).

19.1.3 When Paging Operates

The conversion of an address by the paging mechanism (if fitted) takes place if, and only if, the following conditions all obtain:-

(a) the processor is in the paging mode (see Section 2.4.9.5);

(b) the processor is operating in the P1 or P2 state;

(c) the address in question is an instruction-specified operand address greater than 71, or any instruction address specified by a Branch or Execute instruction or by the Sequence Control Counter. Thus instructions as well as data are relocated by the paging mechanism.

Note 1. Addresses generated by input output channels, are not converted. It is the Supervisor's responsibility to ensure that virtual store areas required for input output operations are present in the physical main store and, if they cross page boundaries, that they occupy consecutive locations therein.

Note 2. The addresses left in General Purpose Registers by the instructions 'Load Address' and 'Branch and Link' (both versions) are unconverted.

Note 3. Instruction-specified operand addresses in the range 0-71 are not converted. *

19.1.4 Outline of Paging Mechanism

As described in Section 19.1.2 the basic function of the paging mechanism is to convert addresses of the form 4096n + x $(0 \leqslant x$ $n \leqslant 4095)$ to the form 4096N + x (x unchanged). For this purpose the 12-bit field containing n is split into an 8-bit and a 4-bit field, containing numbers referred to as s and p, respectively, so that n = 16s + p $(0 \leqslant p \leqslant 15, 0 \leqslant s \leqslant 255)$.

Thus the virtual store of each program is effectively split into a maximum of 256 segments, each segment containing up to 16 pages. s is the segment number, and p the page number within that segment, of any particular page. Address conversion is achieved by using two types of tables, called 'Page Tables' and 'Segment Tables'. Each program operating in the paging mode uses one page table for each segment it addresses, and a single segment table. These tables are held in the main store. Each page table comprises a set of consecutive 4-byte locations, or entries; the $(p + 1)^{th}$ entry of the page table for segment s contains the value of N (if it exists) corresponding to the page for which n = 16s + p. Each entry also includes a one-bit marker to indicate whether or not the page is present in the main store, i.e. whether or not N exists. The segment table similarly consists of a list of 4-byte entries, the $(s + 1)^{th}$ entry containing the (absolute) starting address of the page table for segment s. The starting address of the segment table for the current program is held in a special Scratchpad register (the Segment Table Base Register) where it is placed by the Supervisor prior to resumption of the program. Each program can thus have its own segment table.

19.1.4 Continued.

Any page table may be referred to by more than one program, by having its start address included in more than one segment table. In such cases the numbers of the segments concerned need not be the same, since the contents of the page table do not refer to the segment number. This allows different programs to have access to common segments, which are included in the virtual stores of each of these programs; although each program may refer to a common segment by a different number, the numbering of the pages within the segment is the same.

To convert a virtual address to absolute form requires two main store accesses, to a segment table entry (whose address is determined by adding the segment number to the contents of the Segment Table Base Register) and then to a page table entry (whose address is determined by adding the number of the page within the segment to the contents of the segment table entry). As it would take too long to perform this operation for every address, the paging mechanism also includes a fast Associative Memory, arranged to contain the numbers and locations (i.e. the values of n and N) of the eight pages most recently referred to. Every time an operand or instruction address of the form 4096n + x is to be converted, the Associative Memory is searched to see if this value of n is already present therein. If it is, the corresponding N is extracted from the Associative Memory. If it is not, n is split into its s and p portions, which are used to extract the value of N from the appropriate page table and thus generate the required absolute address : 4096N + x. In addition these values of n and N are inserted in the Associative Memory, replacing the item least recently used. If there is no value of N, i.e. the page in question is not present in the main store, a 'Page Turning' interrupt occurs, the number of the required page being left in a Scratchpad location so that the Supervisor can bring that page into the main store and update the page tables accordingly.

The contents of the segment and page tables, and hence the allocation of virtual storage and the assignment of pages to locations in the main store, are controlled completely by the Supervisor.

Further details of the paging mechanism are described in Section 19.2.

19.1.5 Store Protection and Monitoring

Since, in the paging mode, all program-generated addresses are converted via the segment and page tables, and since, as described in Section 19.2, these tables incorporate checks on the virtual addresses presented for conversion, store protection can be ensured by the Supervisor without relying exclusively on the use of the reservation keys (see Section 3.4). These may be used as extra checks on the system.

19.1.5 Continued.

The Write and Read markers which form part of the reservation keys can be used by the Supervisor to discover the use which has been made of any page, and hence to determine which page or pages should be discarded from the main store when new ones have to be brought in.

19.1.6 Compatibility Between Paging and Non-Paging Modes

Non-privileged programs written for non-paging systems may be run under paging conditions, and vice versa, provided that the necessary program/Supervisor interfaces, etc., are compatible and that the storage allocation procedures of the Supervisor are suitable. In particular:-

(a) A program which relies on paging may run in non-paging mode provided that the physical main store is large enough to accommodate the virtual store used by the program, virtual addresses being treated as absolute.

(b) A non-paging program may run in paging mode profided that enough consecutive segment numbers are allocated to it to provide a virtual store compatible with the range of addresses used by the program.

19.2 The Paging Mechanism

19.2.1 Types of Address Converted

Section 19.1.3 lists the conditions governing the use of the paging (address conversion) mechanism. When these conditions obtain, all instruction addresses (including branch addresses) and instruction operand addresses are automatically presented to the paging mechanism for conversion, after addition of the relocation base address. As previously described, a virtual address of the form 4096n' + x becomes, after addition of the relocation base address, 4096n + x (where n = 16s + p) and is converted to 4096N + x.

The terms n, N, s, p and x are used throughout this section to refer to typical addresses.

19.2.2 The Associative Memory

19.2.2.1 Formats and Use

The Associative Memory comprises eight non-addressable 24-bit cells. Each cell contains the following:-

Virtual page number (n)	(12 bits)
Absolute page number (N)	(12 bits)

19.2.2.1 Continued.

A cell is said to be valid when the values of n and N which it contains correspond to one of the eight most recently addressed pages in the main store. Further information on validity is given in Section 19.2.2.4.

Any address to be converted, after incrementing by the relocation base address, is first referred to the Associative Memory. If the relevant value of n is equal to the value of any of the valid cells (all of which are examined) conversion is completed by extracting the value of N from that cell and substituting it for the most significant 12 bits of the address.

If no matching valid cell in the Associative Memory exists, table look-up in the segment and page tables is automatically performed; if this is successful, i.e. if the page is present in the main store, a new entry for this page is automatically generated and inserted in the Associative Memory, and the conversion is completed. Otherwise interruption occurs.

19.2.2.2 Order of Recency

Included in the Associative Memory is an automatic mechanism which effectively remembers the order in which the cells within it were most recently used, i.e. had their contents successfully matched with addresses presented for conversion. This mechanism is automatic and not under program control. It makes its effects felt in two ways:

(a) After a failure to match an address to be converted against any of the values of n in the Associative Memory, the new entry subsequently generated by look-up in the segment and page tables over-writes the cell containing the least recently used entry, which itself becomes the most recently used. The remaining cells are ordered below in the previous sequence.

(b) When a 'Load Scratchpad' or 'Store Scratchpad' instruction is used to insert or read out an item in the Associative Memory, the cell over-written, or read from, is the one least recently used; however, as soon as that cell has been referenced (i.e. immediately the contents of the cell have been transferred from or to the main store, and before the next transfer) it becomes the most recently used cell and the others step down, letting another take its place at the tail end of the order of recency.

19.2.2.3 Loading and Storing the Associative Memory

If a 'Load Scratchpad' or 'Store Scratchpad' instruction is executed which refers to a Scratchpad location in the range 128-255 (decimal), the words involved are transferred between the designated main store locations and the Associative

*

19.2.2.3 Continued.

Memory. Regardless of the actual Scratchpad addresses specified, the first word transferred either overwrites (Load Scratchpad) or is read from (Store Scratchpad) the least recently used cell (which immediately becomes the <u>most</u> recently used) and so on. Thus a different cell is accessed for each word transferred. The number of words transferred is determined by the L field.

All 24 bits of each cell are read out or overwritten. The format of the main store words involved is as follows:-

ignored	n	N
0	8	20 31

After the completion of a 'Store Scratchpad' instruction involving the Associative Memory, all cells are automatically set invalid, regardless of the number actually read out.

(a) Any address between 128 and 255 inclusive, refers to the Associative Memory.

(b) If more than 8 addresses are specified within the above limits, the mechanism continues to load or store the Associative Memory and overwrites or stores again its contents either until the count reaches zero or until the scratchpad address wraps around and address 0 is specified.

(c) It is permissible to address the Associative Memory as the last part of an operation which initially specified the normal part of the Scratchpad.

Further information is given Section 4.4.5 and D.2.5.

19.2.2.4 <u>Validity of cells</u>

A cell becomes invalid when a 'Store Scratchpad' instruction involving the Associative Memory is executed; <u>all</u> the cells are made invalid, whether or not their contents were actually transferred.

A cell becomes valid for one of two reasons:-

(a) When a new entry is generated by a successful automatic look-up in the page and segment tables, following failure to find a match among the existing entries.

(b) When a cell is overwritten by the 'Load Scratchpad' instruction.

19.2.2.5 <u>Page Usage Analysis</u>

The page numbers of pages accessed during the execution of a

19.2.2.5 Continued.

program can be estimated either by inspecting the main store reservation keys (See Section 19.1.5) or by examining the order in which the most recently accessed pages were used. This latter method requires less processor time and is performed as follows:-

(a) before the execution of a program is either initiated or restarted, all the cells of the Associative Memory are loaded with a standard entry which will not be used by program (s = 255, p = 15 and N = 4,095, i.e. all '1's, is the most likely choice). The validity bits of the cells are then reset by executing a 'Store Scratchpad' instruction specifying one of the cells. Finally the n, N value of pages likely to be used by the program are stored in the memory by executing the 'Load Scratchpad' instruction..

(b) after execution of a program has at least temporarily ceased, the cells of the Associative Memory are loaded into main store. The order in which they are placed will reflect the recency of usage and any cells not used will be identified by the previously chosen void value for n and N.

19.2.3 Segment and Page Tables

19.2.3.1 The Segment Table Base Register

The absolute starting address of the current segment table is contained in a Scratchpad location whose address is specified in Section 4.4.5. The format of this register is as follows:-

Limit	Segment table address	
0	8	31

The address is contained in bits 8 - 31. Its two least significant bits must be zero. The limit field contains the largest segment number which the program is permitted to use.

Following a failure to match an address in the Associative Memory, a word in the segment table is referenced by adding 4s to the address in the Segment Table Base Register, to form its address. An 'Address Error' interrupt is requested when referring to the segment table, if s is greater than the limit or if the address generated lies outside main store.

19.2.3.2 The Segment Table

Each segment table entry, of which there can be up to 256, occupies 4 bytes and has the following format:

Ignored	Page table address	
0	8	31

The absolute starting address of the appropriate page table is held in bits 8-31. The two least significant bits must be zero.

A segment table entry is used in the following way:

the address of the page table entry required is formed by adding 4p to the page table address. An 'Address Error' interrupt occurs if, in referring to the page table, the address generated lies outside the main store.

19.2.3.3 The Page Table

Each page table entry, of which there can be up to 16 per table, occupies 4 bytes and has the following format:

A	Spare	p	N	
0	1	16	20	31

(A = Availability)

The leftmost bit of an entry indicates whether the corresponding page is present (1) or absent (0) in the main store. If the page is present, bits 16 - 19 and 20 - 31 contain p, the page number within the segment, and N, the absolute page location, respectively. The segment number is not included, since the same page table may be referred to under different segment numbers by different programs. The 'spare' field, bits 1 - 15 (and bits 16 - 31, if bit 0 = 0) may contain any information.

In addition to the interrupt mentioned in Section 19.2.3.2 above, an 'Address Error' interrupt (Page Number Error) will be requested if the page is marked as present (bit 0 = 1) but the value of p in bits 16 - 19 does not agree with that used in referring to the entry. This error is detected after generating the new entry in the Associative Memory, and prevents the completion of the address conversion process.

If the page is marked as absent, a 'Page Turning' interrupt (Section 19.2.4.2) occurs.

Segment and page table entries are not altered by the address conversion process, and are completely under program (i.e. Supervisor) control.

19.2.4 Program Interrupts

19.2.4.1 Introduction

While a program is operating in the paging mode it is subject to all the normal types of interruption. In addition, there is a 'Page Turning' interrupt which occurs when, following a failure to match an address presented for conversion with any of the values of n in the Associative Memory, the contents of the appropriate page table entry shows that the required page is not in the main store.

The 'Page Turning' interrupt is executed without activating the normal flag scanning sequence of the Interrupt Control System. The interrupt thus has immediate effect and does not have a position in the normal interrupt priority queue.

The 'Page Turning' interrupt has neither an interrupt flag nor a mask bit associated with it. It cannot occur whilst the processor is operating in the standard address mode.

Several Page Turning interrupts may occur in the course of executing a single instruction, in all cases, however, the execution can be successfully restarted once the missing page (s) has (have) been fetched.

19.2.4.2 The Page Turning Interrupt

The Page Turning' interrupt is executed in the following manner:

1. The current instruction is inhibited.

2. The central control storage registers of the interrupted state receive the normal information stored during a change of state. The Sequence Control Counter contains the address of the start of the current instruction. The instruction length code may be stored incorrectly.

3. The Segment/Page number which could not be interpreted is placed in bits 8 to 19 of register 5 of processor state four.

4. The Interrupt State Identifier field is loaded with the code associated with the interrupted processor state.

5. The Sequence Control Counter is loaded with the contents of register 4 of Processor State Four.

6. Normal processor operation commences in processor state three.

19.2.4.2 Continued.

Address conversion is caused to occur at a sufficiently early stage in the execution of every instruction to permit the instruction to be restarted, following a 'Page Turning' interrupt, from scratch. All SS instructions, and the instructions 'Load Multiple' and 'Store Multiple' are checked initially to ensure that if their operand fields cross page boundaries, the absolute addresses of the pages in question are available. The worst case which can occur is an 'Execute' instruction which executes a 'Move'-type instruction; this may involve up to 8 'Page Turning' interrupts (so that the 'Execute' instruction is initiated nine times in all), since the 'Execute' instruction, the 'Move' instruction, and its two operand fields may each straddle two pages which were not in the store prior to initiating the instruction for the first time.

19.2.4.3 Program Error Conditions

The three error conditions specifically relating to the paging mode have been described in Section 19.2.3. In addition, all data and instruction addresses associated with an instruction are checked in the normal way either before conversion (for alignment errors, etc.) or after conversion (for protection violation, etc.) as appropriate, and all the other checks are performed as usual. Detection of error conditions leads to instruction termination in the usual way; no further address conversion is performed on the instruction.

20. EMULATION FACILITIES

It is not possible for the 4-70 Central Control Unit to emulate the action of other processors.

APPENDIX A

INSTRUCTION EXECUTION TIMES

SYMBOLS

B	-	Number of Bytes processed (B ≤ L)
CRT	-	Channel Response Time (see Appendix C Section 5)
D	-	Number of Digits
I.Ex	-	Instruction Execution Time
INT	-	Time taken to service interrupt if applicable
K	-	Number of Control Characters
L_1	-	Number of bytes in first operand
L_2	-	Number of bytes in second operand
L	-	Number of bytes specified by L field
P	-	Number of 4-bit shifts (next highest integer if not divisible by 4, minimum is 1).
R	-	Number of Registers.

GENERAL NOTES FOR 4-70 and 4-75

1. All instruction times include staticising time. Times are the average produced by a string of instructions of the same type, randomly chosen.

2. Single modification is assumed throughout. To RX instructions which use both index and base registers (double modification) add 0.15 usecs, except those marked +, ‡ or x. To those marked + add 0.30 usec, to those marked ‡ add zero and to those marked x add 0.05 usec.

3. Instruction times do not include recomplementation where necessary.

4. Normalisation, where performed, is taken as one hexadecimal place shift.

5. Decimal instruction times are calculated on the basis of the first operand field being greater than the second.

6. Measured instruction times will be within ± 5% of those quoted.

7. Control Characters, K, are all characters in the first operand field except Digit Select and Start Significance.

/continued.....

NOTE FOR 4-75 OPERATION

4-75 instruction execution times are greater than those of a 4-70 by an amount which is dependent on instruction type and is composed of three independent factors. The first factor results from the basic functioning of a 4-75 and is independent of its mode of operation. The second factor, applicable only when the processor is in the paging mode, results from the address conversion overhead of the Associative Memory. The final factor covers the "maintenance" of the Associative Memory and applies where it is necessary for the contents of the Associative Memory to be changed before an instruction can be completed.

Instruction times corrected for the first two factors are given in the following tables. The 'maintenance correction' is dependent on program structure etc., and can only be applied to individual cases for which it is known what the likelihood of any store access requiring maintenance action is. The following figures give maintenance execution times for each instruction class. To enable the times to be applied, the number of accesses required for each operation is also given.

The necessity for applying the maintenance factor is indicated by ∅ in the tables.

ASSOCIATIVE MEMORY MAINTENANCE TIMES

Instruction	Type of Access	No. of Accesses	Maintenance (usecs)
RR Instructions	instruction	0.5	5
RX Branches) RS Shifts) SI Privileged)	instruction	1.0	5
RX, SI except those above	instruction	1.0	5
	operand	1.0	5 4.7(single index RX)
RS (LM, STM)	instruction	1.0	5
	operand	1-16	6
SS	instruction	1.5	5
	operand	L+L	5.5 (end)
		$L_1 + L_2$	7.0 (start)

PRIVILEGED INSTRUCTIONS

Instruction	Format	Mnemonic	Op Code	Timing (µs)		
				4-70	4-75 non-paging	4-75 paging ø
Diagnose	SI	DIG	83	10.00	10.06	10.12
Load Scratch-Pad	SS	LSP	D8	3.04 + 0.48R	3.13 + 0.54R	3.70 + 0.60R
Idle	SI	IDL	80	2.62	2.68	2.74
Program Control	SI	PC	82	2.12 + INT	2.12 + INT	2.12 + INT
Store Scratch-Pad	SS	SSP	D0	2.73 + 0.52R	2.82 + 0.58R	3.39 + 0.64R
Insert Storage Key	RR	ISK	09	3.06	3.12	3.18 Wrong ?
Set Storage Key	RR	SSK	08	1.38	1.38	1.38 Wrong ?
Read Direct (Optional)	SI	RDD	85	3.66	3.74	3.81
Write Direct (Optional)	SI	WRD	84	3.68	3.80	3.92
Start Device	SI	SDV	9C	0.9 + CRT	0.96 + CRT	1.02 + CRT
Halt Device	SI	HDV	9E	0.9 + CRT	0.96 + CRT	1.02 + CRT
Check Channel	SI	CKC	9F	0.9 + CRT	0.96 + CRT	1.02 + CRT
Test Device	SI	TDV	9D	0.9 + CRT	0.96 + CRT	1.02 + CRT

PROCESSING STATE CONTROL INSTRUCTIONS

Set Program Mask	RR	SPM	04	0.68	0.71	0.74
Supervisor Call	RR	SVC	0A	1.48	1.51	1.54

BRANCHING INSTRUCTIONS

Instruction	For-mat	Mnem-onic	Op Code	Timing (μs)		
				4-70	4-75 Non-Paging	4-75 Paging ∅
Branch and Link	RR	BALR	05	Branch = 2.54 No Branch = 1.76	Branch = 2.60 No Branch = 1.78	Branch = 2.66 No Branch = 1.79
Branch and Link +	RX	BAL	45	2.38	2.44	2.50
Branch on Condition	RR	BCR	07	Branch = 2.06 No Branch = 1.28	Branch = 2.12 No Branch = 1.30	Branch = 2.18 No Branch = 1.31
Branch on Condition +	RX	BC	47	Branch = 1.90 No Branch = 1.37	Branch = 1.96 No Branch = 1.40	Branch = 2.02 No Branch = 1.43
Branch on Count	RR	BCTR	06	Branch = 2.58 No Branch = 1.80	Branch = 2.64 No Branch = 1.82	Branch = 2.70 No Branch = 1.83
Branch on Count +	RX	BCT	46	Branch = 2.42 No Branch = 1.89	Branch = 2.48 No Branch = 1.95	Branch = 2.54 No Branch = 2.01
Branch on Index High	RS	BXH	86	Branch = 3.24 No Branch = 2.71	Branch = 3.30 No Branch = 2.74	Branch = 3.36 No Branch = 2.77
Branch on Index Low or Equal	RS	BXLE	87	Branch = 3.24 No Branch = 2.71	Branch = 3.30 No Branch = 2.74	Branch = 3.36 No Branch = 2.77
Execute ‡	RX	EX	44	3.66 + I.Ex.	3.78 + I.Ex.	3.90 + I.Ex.

DECIMAL INSTRUCTIONS

Instruction	Format	Mnemonic	Op Code	Timing (μs)		
				4-70	4-75 Non-Paging	4-75 Paging ∅
Add Decimal	SS	AP	FA	$4.93 + 0.83L_1 + 0.16L_2$	$5.01 + 0.86L_1 + 0.18L_2$	$5.81 + 0.89L_1 + 0.19L_2$
Subtract Decimal	SS	SP	FB	$4.93 + 0.83L_1 + 0.16L_2$	$5.01 + 0.86L_1 + 0.18L_2$	$5.81 + 0.89L_1 + 0.19L_2$
Multiply Decimal	SS	MP	FC	$5.10 + 1.80L_1 + 4.50L_2$ $+ 3.35L_2 (L_1 - L_2)$	$5.16 + 1.86L_1 + 4.65L_2$ $+ 3.59L_2 (L_1 - L_2)$	$5.98 + 1.92L_1 + 4.80L_2$ $+ 3.83L_2 (L_1 - L_2)$
Divide Decimal*	SS	DP	FD	(a)$7.90 + 5.20L_1 - 4.9L_2$ $+ 6.8L_2 (L_1 = L_2)$ (b)$8.80 - 0.8L_1 + 1.9L_2$ $+ 6.8L_2 (L_1 - L_2)$	(a)$8.22 + 5.35L_1 - 5.05L_2$ $+ 7.2L_2 (L_1 = L_2)$ (b)$9.12 - 0.82L_1 + 1.93L_2$ $+ 7.2L_2 (L_1 - L_2)$	(a)$9.02 + 5.50L_1 - 5.20L_2$ $+ 7.6L_2 (L_1 = L_2)$ (b)$9.92 - 0.83L_1 + 1.96L_2$ $+ 7.6L_2 (L_1 = L_2)$
Compare Decimal	SS	CP	F9	$4.57 + 0.72L_1 + 0.16L_2$	$4.63 + 0.74L_1 + 0.18L_2$	$5.41 + 0.75L_1 + 0.19L_2$
Move with Offset	SS	MVO	F1	$3.38 + 0.72L_1 + 0.17L_2$	$3.46 + 0.74L_1 + 0.18L_2$	$4.26 + 0.75L_1 + 0.20L_2$
Pack	SS	PACK	F2	$3.06 + 0.64L_1 + 0.19L_2$	$3.14 + 0.66L_1 + 0.21L_2$	$3.94 + 0.67L_1 + 0.22L_2$
Unpack	SS	UNPK	F3	$3.34 + 0.34L_1 + 0.19L_2$	$3.42 + 0.36L_1 + 0.21L_2$	$4.22 + 0.37L_1 + 0.22L_2$
Zero and Add	SS	ZAP	F8	$5.17 + 0.36L_1 + 0.19L_2$	$5.25 + 0.38L_1 + 0.21L_2$	$6.05 + 0.39L_1 + 0.22L_2$

* (a) 2 leading digits of divisor non-zero.
(b) 2 leading digits of divisor zero.

Issue	Date	Auth'y	Log
4	16.2.68	MAW	743

Title: CENTRAL PROCESSOR MODELS 4-70/75

Document No. PS 4.10.70

Page A.6

** (a) Number < 10,000,000
(b) Number ≥ 10,000,000

FIXED-POINT INSTRUCTIONS

Instruction	For-mat	Mnem-onic	Op Code	Timing (μs)		
				4-70	4-75 Non-Paging	4-75 Paging ∅
Add Halfword	RX	AH	4A	2.28	2.37	2.46
Add Logical	RR	ALR	1E	1.04	1.07	1.10
Add Logical	RX	AL	5E	1.89	1.98	2.07
Add Word	RR	AR	1A	1.04	1.07	1.10
Add Word	RX	A	5A	1.89	1.98	2.07
Subtract Halfword	RX	SH	4B	2.28	2.37	2.46
Subtract Logical	RR	SLR	1F	1.04	1.07	1.10
Subtract Logical	RX	SL	5F	1.89	1.98	2.07
Subtract Word	RR	SR	1B	1.04	1.07	1.10
Subtract Word	RX	S	5B	1.89	1.98	2.07
Multiply Halfword	RX	MH	4C	6.39	6.48	6.57
Multiply Word	RR	MR	1C	5.39	5.43	5.45
Multiply Word	RX	M	5C	6.24	6.33	6.42
Divide	RR	DR	1D	10.44	10.47	10.50
Divide	RX	D	5D	11.29	11.38	11.47
Compare Halfword	RX	CH	49	2.28	2.37	2.46
Compare Word	RR	CR	19	1.04	1.07	1.10
Compare Word	RX	C	59	1.89	1.98	2.07
Convert to Binary**	RX	CVB	4F	(a)10.02 (b)17.46	(a)10.17 (b)17.61	(a)10.32 (b)17.76
Convert to Decimal+	RX	CVD	4E	9.52	9.70	9.88
Load and Test	RR	LTR	12	0.76	0.79	0.82
Load and Complement	RR	LCR	13	0.76	0.79	0.82
Load Halfword	RX	LH	48	2.0	2.09	2.18
Load Multiple	RS	LM	98	1.91 + 0.48R	1.97 + 0.54R	2.51 + 0.60R
Load Negative	RR	LNR	11	0.76	0.79	0.82
Load Positive	RR	LPR	10	0.76	0.79	0.82
Load Word	RR	LR	18	0.76	0.79	0.82
Load Word	RX	L	58	1.61	1.70	1.79
Shift Left Single	RS	SLA	8B	1.08 + 0.08P	1.14 + 0.08P	1.20 + 0.08P
Shift Left Double	RS	SLDA	8F	1.60 + 0.08P	1.66 + 0.08P	1.72 + 0.08P
Shift Right Single	RS	SRA	8E	1.08 + 0.08P	1.14 + 0.08P	1.20 + 0.08P
Shift Right Double	RS	SRDA	8E	1.60 + 0.08P	1.66 + 0.08P	1.72 + 0.08P
Store Word	RX	ST	50	1.32	1.38	1.44
Store Halfword	RX	STH	40	3.06	3.12	3.18
Store Multiple	RS	STM	90	1.44 + 0.52R	1.50 + 0.58R	2.04 + 0.64R

English Electric Computers

PRODUCT PLANNING SPECIFICATION

Issue	Date	Auth'y	Log
4	16.2.68	MW	743

Title
CENTRAL PROCESSOR
MODELS 4-70/75

Document No.
PS 4.10.70

Page A.7

English Electric Computers

PRODUCT PLANNING SPECIFICATION

FLOATING POINT INSTRUCTIONS

Instruction		For-mat	Mnem-onic	Op Code	Timing (μs)		
					4-70	4-75 Non-Paging	4-75 Paging ∅
Add Normalized (Long)		RR	ADR	2A	3.25	3.28	3.31
Add Normalized (Long)	x	RX	AD	6A	4.30	4.39	4.48
Add Normalized (Short)		RR	AER	3A	2.69	2.72	2.75
Add Normalized (Short)		RX	AE	7A	3.66	3.75	3.84
Add Unnormalized (Long)		RR	AWR	2E	3.22	3.25	3.28
Add Unnormalized (Long)	x	RX	AW	6E	4.27	4.36	4.45
Add Unnormalized (Short)		RR	AUR	3E	2.66	2.69	2.72
Add Unnormalized (Short)		RX	AU	7E	3.63	3.72	3.81
Subtract Normalized (Long)		RR	SDR	2B	3.25	3.28	3.31
Subtract Normalized (Long)	x	RX	SD	6B	4.30	4.39	4.48
Subtract Normalized (Short)		RR	SER	3B	2.69	2.72	2.75
Subtract Normalized (Short)		RX	SE	7B	3.66	3.75	3.84
Subtract Unnormalized (Long)		RR	SWR	2F	3.22	3.25	3.28
Subtract Unnormalized (Long)	x	RX	SW	6F	4.27	4.36	4.45
Subtract Unnormalized (Short)		RR	SUR	3F	2.66	2.69	2.72
Subtract Unnormalized (Short)		RX	SU	7F	3.63	3.72	3.81
Multiply (Long)		RR	MDR	2C	10.98	11.01	11.04
Multiply (Long)	x	RX	MD	6C	11.95	12.04	12.13
Multiply (Short)		RR	MER	3C	5.58	5.61	5.64
Multiply (Short)		RX	ME	7C	6.71	6.80	6.89
Divide (Long)		RR	DDR	2D	18.0	18.03	18.06
Divide (Long)	x	RX	DD	6D	19.23	19.32	19.41
Divide (Short)		RR	DER	3D	9.36	9.39	9.42
Divide (Short)		RX	DE	7D	10.49	10.58	10.67
Compare (Long)		RR	CDR	29	2.38	2.41	2.44
Compare (Long)	x	RX	CD	69	3.43	3.52	3.61
Compare (Short)		RR	CER	39	1.82	1.85	1.88
Compare (Short)		RX	CE	79	2.79	2.88	2.97
Halve (Long)		RR	HDR	24	1.52	1.55	1.58
Halve (Short)		RR	HER	34	1.00	1.03	1.06
Load (Long)		RR	LDR	28	1.52	1.55	1.58
Load (Long)	x	RX	LD	68	2.57	2.66	2.75
Load (Short)		RR	LER	38	1.00	1.03	1.06
Load (Short)		RX	LE	78	1.73	1.82	1.91

/continued.....

FLOATING POINT INSTRUCTIONS (continued)

Instruction	For-mat	Mnem-onic	Op Code	Timing (µs)		
				4-70	4-75 Non-Paging	4-75 Paging ø
Load and Test (Long)	RR	LTDR	22	1.52	1.55	1.58
Load and Test (Short)	RR	LTER	32	1.00	1.03	1.06
Load Complement (Long)	RR	LCDR	23	1.52	1.55	1.58
Load Complement (Short)	RR	LCER	33	1.00	1.03	1.06
Load Negative (Long)	RR	LNDR	21	1.52	1.55	1.58
Load Negative (Short)	RR	LNER	31	1.00	1.03	1.06
Load Positive (Long)	RR	LPDR	20	1.52	1.55	1.58
Load Positive (Short)	RR	LPER	30	1.00	1.03	1.06
Store (Long)‡	RX	STD	60	1.84	1.90	1.96
Store (Short)+	RX	STE	70	1.46	1.52	1.58

LOGICAL INSTRUCTIONS

Instruction	Format	Mnemonic	Op Code	Timing (µs)		
				4-70	4-75 Non-Paging	4-75 Paging ∅
AND	RR	NR	14	1.04	1.07	1.10
AND	RX	N	54	1.89	1.98	2.07
AND	SI	NI	94	3.55	3.63	3.82
AND	SS	NC	D4	3.01 + 0.98L	3.09 + 1.03L	3.89 + 1.07L
OR	RR	OR	16	1.04	1.07	1.10
OR	RX	O	56	1.89	1.98	2.07
OR	SI	OI	96	3.55	3.63	3.82
OR	SS	OC	D6	3.01 + 0.98L	3.09 + 1.03L	3.89 + 1.07L
Exclusive OR	RR	XR	17	1.04	1.07	1.10
Exclusive OR	RX	X	57	1.89	1.98	2.07
Exclusive OR	SI	XI	97	3.55	3.63	3.82
Exclusive OR	SS	XC	D7	3.01 + 0.98L	3.09 + 1.03L	3.89 + 1.07L
Compare Logical	RR	CLR	15	1.04	1.07	1.10
Compare Logical	RX	CL	55	1.89	1.98	2.07
Compare Logical	SI	CLI	95	3.06	3.18	3.30
Compare Logical	SS	CLC	D5	2.56 + 0.88B	2.62 + 0.91B	3.40 + 0.94B
Edit	SS	ED	DE	2.88 + 1.02K + 1.46D	2.96 + 1.05K + 1.51D	3.76 + 1.08K + 1.55D
Edit and Mark	SS	EDMK	DF	4.22 + 1.02K + 1.46D	4.30 + 1.05K + 1.51D	5.10 + 1.08K + 1.55D
Insert Character ‡	RX	IC	43	3.94	4.06	4.18
Load Address +	RX	LA	41	1.08	1.14	1.20
Move	SI	MVI	92	3.55	3.63	3.82
Move	SS	MVC	D2	3.01 + 0.55L	3.09 + 0.58L	3.89 + 0.61L
Move Numerics	SS	MVN	D1	3.01 + 0.98L	3.09 + 1.03L	3.89 + 1.07L
Move Zones	SS	MVZ	D3	3.01 + 0.98L	3.09 + 1.03L	3.89 + 1.07L
Store Character ‡	RX	STC	42	3.39	3.47	3.54
Test Under Mask	SI	TM	91	3.06	3.18	3.30
Translate	SS	TR	DC	3.01 + 1.7L	3.09 + 1.79L	3.89 + 1.88L
Translate and Test	SS	TRT	BD	4.48 + 1.71B	4.54 + 1.79B	5.32 + 1.86B
Shift Left Double Logical	RS	SLDL	8D	1.6 + 0.08P	1.66 + 0.08P	1.72 + 0.08P
Shift Left Single Logical	RS	SLL	89	1.08 + 0.08P	1.14 + 0.08P	1.20 + 0.08P
Shift Right Single Logical	RS	SRL	88	1.08 + 0.08P	1.14 + 0.08P	1.20 + 0.08P
Shift Right Double Logical	RS	SRDL	8C	1.6 + 0.08P	1.66 + 0.08P	1.72 + 0.08P

INTERRUPT EXECUTION TIMES APPENDIX 'B'

Time required to staticise one or more interrupts and also time required to scan the Interrupt Flag Register in conjunction with the current mask register after the latter has been addressed, is 1.6 microsecs if no change of state is involved.

Time required to perform either of the above operations plus a change of state is dependent on the priority of the interrupt actual executed and is as follows :

Priority	Interrupt Condition	State Initiated	Execution Time(usecs)
1	Power Failure	4	6.62
2	Machine Check	4	6.98
3	External Signal No.0/Channel No.0	3	7.34+CRT*
4	External Signal No.1/Channel No.1	3	7.70+CRT*
5	External Signal No.2/Channel No.2	3	8.06+CRT*
6	External Signal No.3/Channel No.3	3	8.42+CRT*
7	External Signal No.4/Channel No.4	3	8.78+CRT*
8	External Signal No.5/Channel No.5	3	9.14+CRT*
9	Channel No.6	3	7.58+CRT
10	Channel No.7	3	7.94+CRT
11	Channel No.8	3	8.30+CRT
12	Channel No.9	3	8.66+CRT
13	Channel No.10	3	9.02+CRT
14	Channel No.11	3	9.38+CRT
15	Channel No.12	3	9.74+CRT
16	Channel No.13	3	10.10+CRT
17	Channel No.14	3	8.54+CRT
18	Channel No.15	3	8.90+CRT
19	Elapsed Time Clock	3	9.26
20	Console Interrupt Request	3	9.62
21	Supervisor Call Instruction	3	9.98
22	Privileged Operation	3	10.34
23	Op-Code Trap	3	10.70
24	Address Error (execute, addressing, specification, protection, paging)	3	11.06
25	Data Error	3	9.50
26	Exponent Overflow	3	9.86
27	Divide Error	3	10.22
28	Significance Error*	3	10.58
29	Exponent Underflow*	3	10.94
30	Decimal Overflow*	3	11.30
31	Fixed Point Overflow*	3	11.66
32	Debug Mode	3	12.02

*CRT is applicable to channel use only.

Continued....

APPENDIX 'B'(Continued)

Note these times are increased by 0.06 microseconds and by 0.12 microseconds on a 4-75 that is operating in the paging and non-paging modes respectively.

The 'Page Turning' interrupt execution requires 4 microseconds.

Note these times include an allowance for the time lost as a result of disturbing the instruction prefetch system.

APPENDIX C

PERIPHERAL CONTROL SYSTEM PERFORMANCE

C.1 Introduction

I/O Devices are connected through Device Control Units (DCU's) to I/O Channel Standard Interface Trunks. The Channels controlling the Trunks are connected to the Processor via Channel Control Units and have priorities assigned to them dependent upon their demand in the system. The 4-70/75 is provided with three types of Channel Control Unit, the Single Channel, Multichannel and Multiplexor Channel Control Units. The assemblage of Channel Control Units, Channels and Trunks comprises the Peripheral Control System of the 4-70/75.

In this Appendix are defined the parameters which specify the performance of the Peripheral Control System (Section C.3) and also those parameters causing degradation to its performance and to the performance of the Central Control Unit (Section C.4, C.5 respectively). A complete list of the values of these parameters is given in Section C.6.

Section C.2 defines the set of rules governing the configuration of Peripheral Control Systems for the 4-70/75.

This Appendix should be read in conjunction with Section 6.

C.2 Peripheral Control System Configuration Rules

C.2.1 Connection of Channel Control Units

The general rules for the connection of Channel Control Units to a 4-70/75 are as follows:-

(a) The 4-70/75 can accommodate a maximum of SIX Channel Control Units.

(b) The maximum number of Multiplexor Channel Control Units is THREE.

(c) Normal priority order is Single Channel Control Units, Multichannel Control Units, Multiplexor Channel Control Units. The order may be changed for special circumstances.

C.2.2 Connection of Channels

(a) The 4-70/75 can accommodate a maximum of SIXTEEN-n* Channels.

(b) A Single Channel Control Unit can handle ONE Selector Channel.

(c) A Multichannel Control Unit can accommodate a maximum of EIGHT Selector Channels.

* n is the number of direct control lines.

cont/.....

(d) A Multiplexor Channel Control Unit can handle ONE Multiplexor Channel.

C.2.3 Connection of Trunks

(a) The 4-70/75 can handle a maximum of FIFTY-SIX trunks. This limit is imposed by the number of Trunk platters available in a maximum configuration.

(b) Trunks are provided in Groups of 4. Trunks contained in a Group must all be assigned to the same CCU. Trunks associated with a CCU must be in the same cubicle as the CCU.

(c) A Multiplexor Channel Control Unit can handle a maximum of SIXTEEN Trunks.

(d) A Multichannel Control Unit can handle a maximum of EIGHT trunks - ONE for each Selector Channel.

(e) A Single Channel Control Unit can handle a maximum of FOUR trunks - servicing is non-concurrent.

(f) A Trunk can handle only ONE Device Control Unit.

Table C1 summarises the Connection rules:-

Table C1

Control Units (Max 6)	Single Channel CU	Multi Channel CU	Multiplexor CU (Max 3)
Channels (Max 16)	1 Selector	8 Selectors	1 Multiplexor
Trunks (Max 56)	4	8 (One/Channel)	16
Servicing	Non-concurrent	Concurrent	Concurrent
Devices	256	256/Trunk	256
Maximum Throughput Rate	1 MB/sec	660 KB/sec Single Device TPR 500 KB/sec	140 KB/sec
Maximum No. of Groups per Control Unit	1	2	4

Pictorial representations of TableC1 are given in the Configurator (page C4) and the Layout of Platters (page C5).

From the latter it can be seen that the limit of 56 Trunks arises from the maximum of 7 platters with logic for 8 Trunks each.

C.2.4. Connection of Device Control Units

C.2.4.1 General Rule

DCU's of devices having highest data transfer requirements are connected to the channel with highest priority; the remaining DCU's are connected in descending order of data transfer requirements to descending priority sequence of channels.

C.2.4.2 Multichannel

In addition to a System Priority the Multichannel has its own internal priority. Again the general rule is:-

DCU with the highest data transfer rate are connected to the highest priority trunks, and the remaining DCU's in descending sequence.

EXCEPTION: Random Access DCU's should always take priority over other devices regardless of relative transfer rates. This is because Random Access Devices have a critical Command Chain time.

C.2.4.3 Multiplexor

The Multiplexor Channel Control Unit also has its own internal priority but the rules for connection are not as straightforward. For example, fully buffered devices (Printers and Card Punches) can be assigned to a low priority whereas the Typewriter, which does not give an error indication on loss of an input character, must be assigned a high priority. Table C2 lists devices normally assigned to a Multiplexor Channel in order of priority requirements.

Table C2

1. Typewriters, Multi-purpose Device Control Unit.
2. Flow devices e.g. sorter/readers, in order of available pocket selection time.
3. Data Transmission other than Data Exchange.
4. Lector.
5. Card Reader.
6. Paper Tape equipment, Plotters etc.
7. Peripheral Switch Controllers.
8. Fully buffered devices (Card Punch, Line Printers)and Data Exchange.

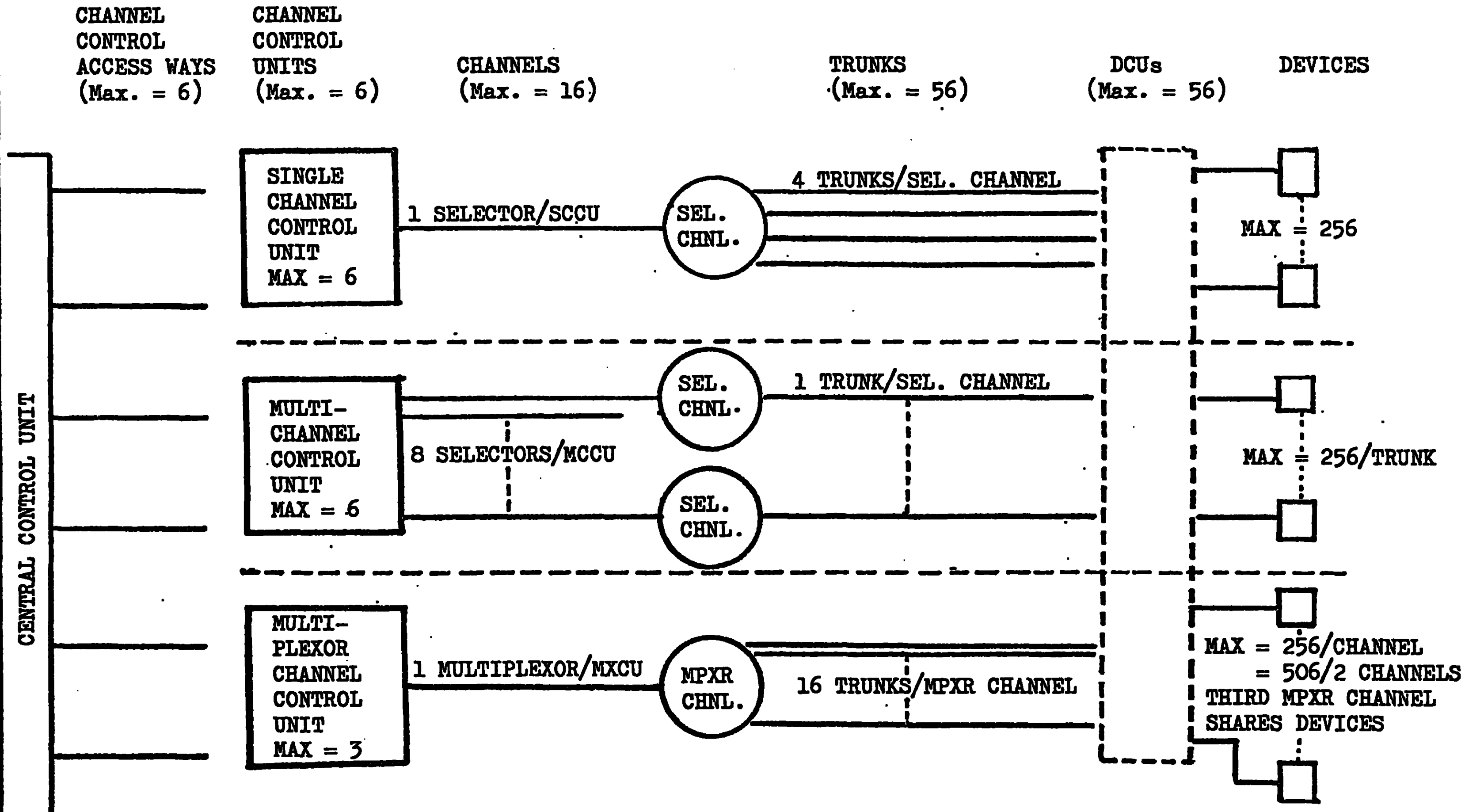
C.2.5 Connection of Devices

(a) A Single Channel Control Unit can accommodate 256 Devices.

(b) A Multichannel Control Unit can accommodate 256 Devices/ Trunk

(c) A Multiplexor channel can accommodate 256 Devices. Main Store reservation is made for two Multiplexor channels but the expected maximum of 512 devices cannot be achieved because the first 24 words of the reserved area are used for Supervisor. Because each device on a MXCU requires 4 words of Main Storage, the maximum number of devices on two MXCU's is 506. (See Section 3.5).

If a third Multiplexor Channel is added to the system, it must share reserved store with one of the other MXCUs.



4-70/75 I/O CONFIGURATOR

Issue	Date	Auth'y	Log
4	16.2.68		743
4/1	10.7.68		809

Title
CENTRAL PROCESSOR
MODELS 4-70/75

Document No.
PS 4.10.70

Page C.4

English Electric Computers

PRODUCT PLANNING SPECIFICATION

*

Issue	Date	Auth'y	Log
4	16.2.68	CAS	743

Title

CENTRAL PROCESSOR
MODELS 4-70/75

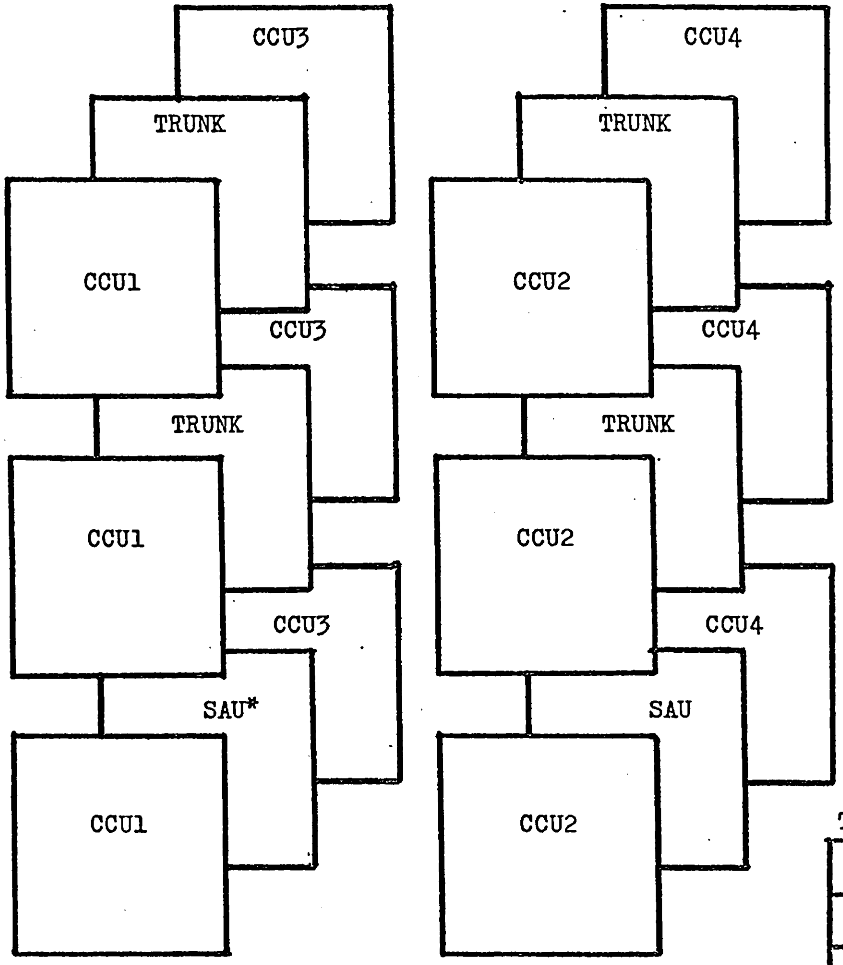
Document No.

PS 4.10.70

Page C.5

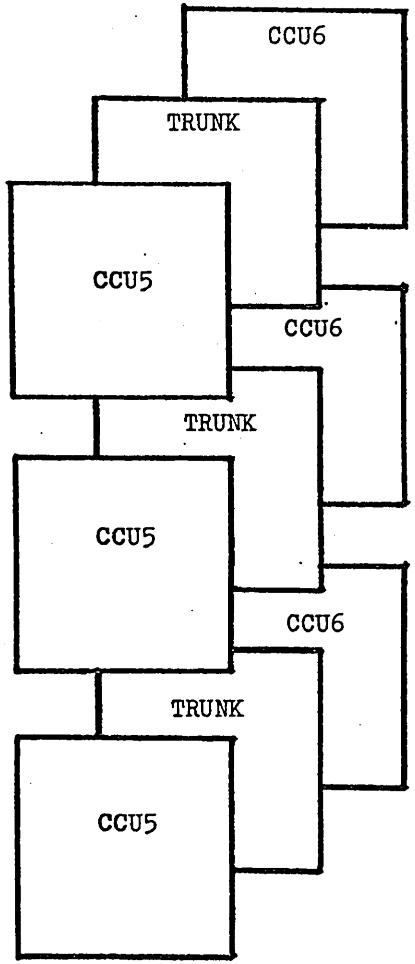
English Electric Computers

PRODUCT PLANNING SPECIFICATION



DOUBLE CUBICLE

*SAU = Store Access Unit



SINGLE CUBICLE

TRUNK PLATTER LAYOUT

TRUNK	TRUNK
TRUNK	TRUNK
TRUNK	TRUNK
TRUNK	TRUNK
GRP O/HD	GRP O/HD
PLATTER OVERHEAD	

4-70/75 I/O CUBICLE LAYOUT

C.3 PERIPHERAL CONTROL SYSTEM PERFORMANCE

C.3.1 Introduction

This section defines the performance of the Peripheral Control System in terms of the maximum throughput rates achievable by the total System and by the individual Channel Control Units.

C.3.2 Definition of Throughput Rate

Maximum Throughput rate of a System or Unit is defined as the maximum number of bytes which can be transferred through the System or Unit between one or more Device Control Units and Main Store. The maximum System throughput rate is determined by the rate at which bytes can be accepted by Main Store. The Maximum throughput rate of a Channel Control Unit is determined by the minimum time to service successive bytes.
Minimum Byte Servicing Times for the various Channel Control Units are given in Section C.6. Maximum Throughput rates (TPR) of the Channel Control Units are given in Table C.1.

C.3.3 Qualifications to Figures quoted in Section C.3.

It must be noted that the throughput figures quoted in this section are best case, based on minimum successive Byte Servicing Times. No allowance whatsoever has been made for degradations to these rates resulting from interference from other Input Output operations or the Central Control Unit.

Neither has any allowance been made for tolerances on throughput rates of individual devices. In configuring a device to any part of the Peripheral Control System, these tolerances must be taken into account to avoid exceeding the performance limits of that part.

C.3.4 Maximum Throughput Rate of Peripheral Control System

The maximum throughput rate of the overall Peripheral Control System is quoted here as 4 Megabytes/sec. The derivation of this figure assumes that all available Main Store cycles are used for Input Output transfers. The figure is an approximation only; it is based on an approximate Store Cycle time of 1 usec for 4 Bytes and makes no allowance for the possible benefits of store interleaving. However, it is justified in that Systems configured within this limit are unlikely to experience overload conditions.

To ensure that the attached Channel Control Units do not exceed the 4 MByte limit, the rates they impose on Main Store for byte servicing must be taken into account. The Multiplexor Channel Control Unit requires 5 Main Store accesses per byte or 20 per word, whereas the Single Channel and Multi-channel Control Units require only one store access per word. The following equation therefore defines the limitation on the System Throughput rate :-

C.3.4 (continued)

$$20 \times M + S + MU \leq 4\text{MBytes/sec}$$

where M = Sum of throughput rates of DCUs on Multiplexor Channel Control Units.
S = Sum of throughput rates of fastest DCUs on Single Channel Control Units.
MU = Sum of throughput rates of DCUs on Multi Channel Control Units.

C.3.5 Maximum Throughput Rate of the Single Channel Control Unit (SCCU)

The maximum throughput rate of the Single Channel Control Unit is 1 Megabyte/sec.

This figure is the reciprocal of the fastest byte servicing time available - 1 usec.

If the throughput rate of the SCCU is 700 KB/sec or above, the SCCU sequences go into a fast loop in which no time is available for sequencing service requests from the Central Control Unit. Simple condition code responses to Start Device, Test Device and Check Channel Instructions can be dealt with by non-sequenced hardware but Halt Device and the hardware instruction 'Service Interrupt' have to be serviced by SCCU sequences and consequently hold up the Central Control Unit until the throughput rate of the SCCU falls below 70O KB/sec. The effect of this restriction on Central Control Unit performance is described in Section C.5. *

C.3.6 Maximum Throughput Rate of the Multi channel Control Unit (MCCU)

A special feature of the MCCU is that servicing of bytes from different trunks can be overlapped, with a consequent reduction in average byte servicing time. The advantage of overlapping is lost, however, if one trunk requires two or more bytes to be serviced in succession.

With full overlapping, the MCCU has a maximum throughput rate of 660 KB/sec. This rate is only achievable if more than one trunk is connected and no Device Control Unit has a throughput rate greater than 330 KB/sec. The former restriction arises from the need to overlap services; the reason for the latter is that a DCU with a throughput rate greater than 330 KB/sec will pass more bytes in a given time than all the other DCUs together and will, therefore, at some time, require more than one byte serviced in succession, with consequent loss of the overlap benefit. As the throughput rate of the fastest DCU increases above 330 KB/sec, so the loss of the overlap advantage increases until the limit is reached where one DCU has a throughput rate of 500 KB/sec and the MCCU cannot service any other DCUs. The maximum single device throughput rate is therefore 500 KB/sec.

The performance of MCCU is summarised graphically on page C.9.

C.3.7 Maximum Throughput Rate of the Multiplexor Channel Control Unit(MXCU)

C.3.7.1 Normal Byte Servicing Mode

The Multiplexor Channel requires five Store Accesses for every byte processed in normal operation and its throughput rate is consequently heavily dependent on Main Store Access Time as seen from the Channel. The relationship between Throughput Rate, T, and Store Access Time is given by the equation

$$T = \frac{1000}{5A+2.7} \quad \text{KB/sec}$$

where A is Main Store Access Time, seen from the Channel. A plot of this equation is given in the Graph on page C.10.

The nominal Multiplexor Channel Control Unit throughput rate is 140 KB/sec. This requires a Store Access Time seen from the Channel of 0.9 usecs. (Increases in this time will cause a further degradation of MXCU TPR at a rate of 1 KB/10 nsecs).

The sum of TPRs of DCUs attached to a MXCU must not exceed 140 KB/sec.

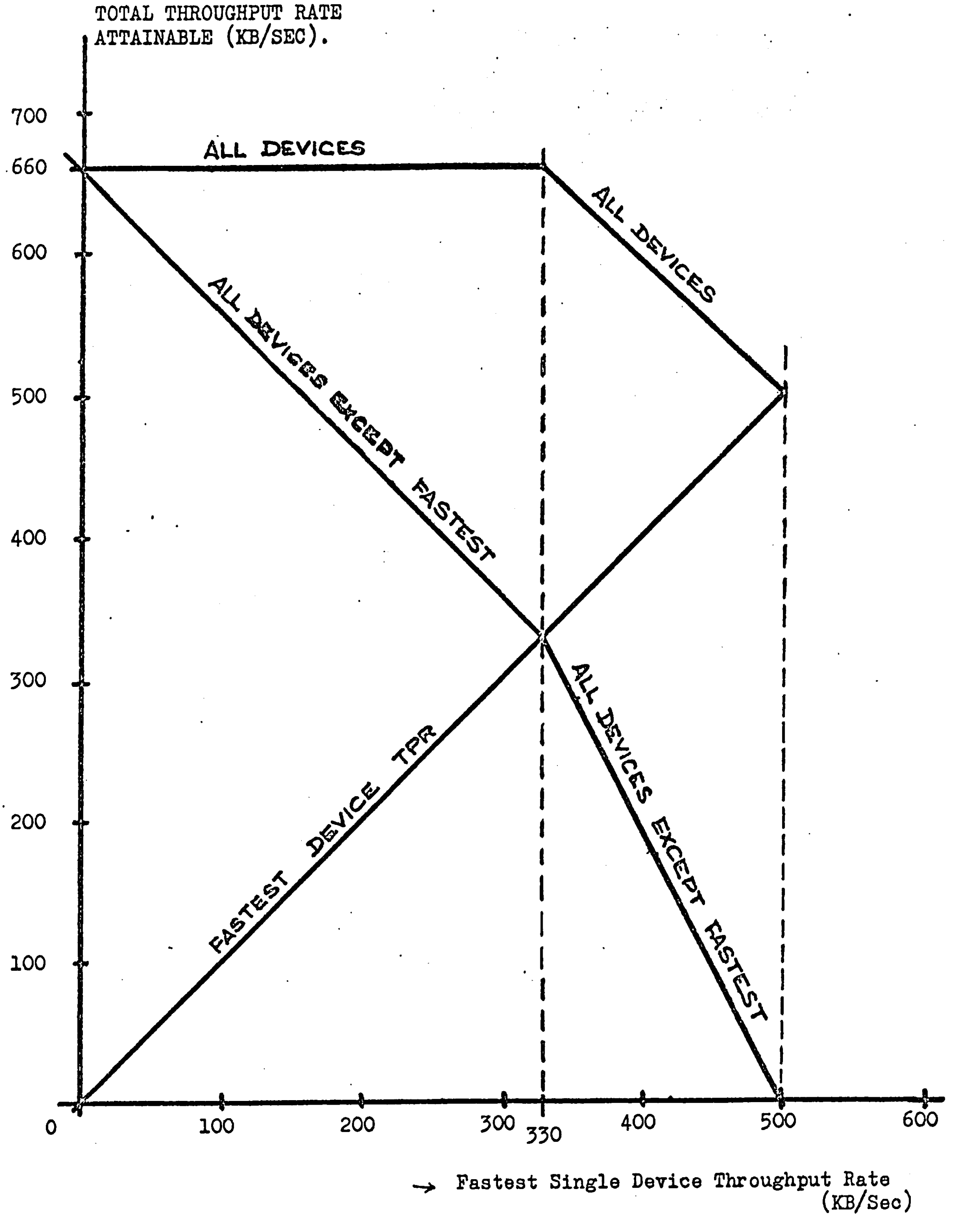
C.3.7.2 Miniburst Operation

Besides working in the normal Multiplexing mode, the MXCU can work in a mini-burst mode in which all the Channel facilities can be dedicated to one device for two or more successive byte services. In this mode, byte transfers require only one store access instead of five, and MXCU throughput rate is increased. The relationship between throughput rate, Tm, and Store Access Time, A, seen from the Channel, is given by the equation

$$Tm = \frac{1000n}{(5A + 2.7) + (n - 1)\ (A + 1.3)}$$

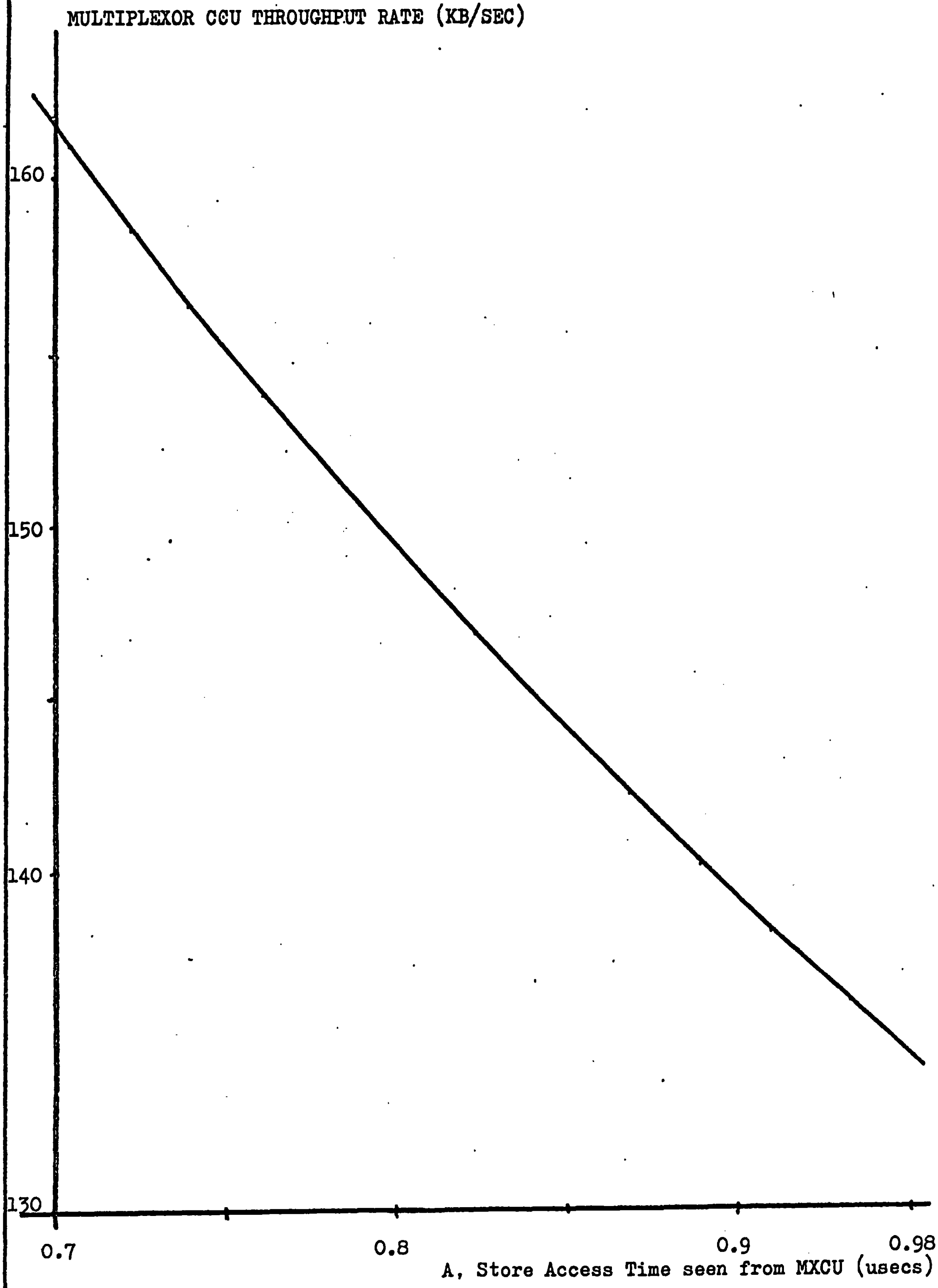
where n is the number of bytes processed in a mini-burst. Depending on the length of mini-burst, throughput rate can reach a maximum of about 450 KB/sec.

It should be noted that a miniburst can only be requested by a *
DCU and not by program or the MXCU. *



MULTICHANNEL CONTROL UNIT PERFORMANCE GRAPH

Issue	Date	Auth'y	Log	Title	Document No.
4	16.2.68	CRS	743	CENTRAL PROCESSOR MODELS 4-70/75	PS 4.10.70
					Page C.9



MULTIPLEXOR CCU THROUGHPUT Vs STORE ACCESS TIME

Issue	Date	Auth'y	Log	Title	Document No.
4	16.2.68	CAS	743	CENTRAL PROCESSOR MODELS 4-70/75	PS 4.10.70
					Page C.10

C.4 SOURCES OF DEGRADATION TO INPUT OUTPUT PERFORMANCE

C.4.1 Introduction

This section gives a mainly qualitative account of those operations of the Channel Control Units and of the Central Control Unit which can degrade the performance figures of a Channel Control Unit below the maxima defined in Section C.3.

C.4.2 Priority for Store Accessing

When Channel Control Units require access to main store they signal their requests to the Input Output Store Multiplexor. The Store Multiplexor records the requests and services them according to the priority of the originating Channel Control Units. If the two highest priority requests to Input Output Store Multiplexor require to access opposite halves of the store, their servicing may be overlapped. Request from lower priority Units cannot 'jump the queue' even if all higher priority requests are waiting to access the other half of the store.

Once a request is recognised as the highest priority request it still has to compete with store access demands from the Central Control Unit; if the latter has already claimed the store module which the Channel requires, a further delay can occur.

Channel Control Units' requests for store access can therefore be delayed both by requests from more prior Channel Control Units or by requests from the Central Control Unit; the delay due to the former will depend on the loading of the more prior CCUs. The result of such delays is that any Channel operations requiring store accesses can take longer than the minima listed in Section C.6. Most significant is the possible increase in the average byte servicing time with the consequent decrease in Channel throughput rate. The effect is complex and no quantitative evaluation of the degradation to Channel Control Unit is attempted here.

C.4.3 Chaining

C.4.3.1 Introduction

Data Chaining and Command Chaining operations require significant action by the Channel Control Units during which action byte servicing cannot take place. Sufficient byte storage must therefore be provided in the attached Device Control Units (DCUs) to allow the DCU to continue servicing its device during the chaining operation. If a DCU does not have sufficient buffering, an 'overrun' condition occurs and the Service Request Not Honoured bit is set in the DCU's Secondary Indicators. Depending on the type of device, this error condition may be recoverable and

C.4.3.1 (continued)

can act as a safety valve for shedding peak overloads; some occurrences of this error may be tolerable but system efficiency will suffer if persistent overloads are likely.

The sequence times for chaining operations in the three types of Channel Control Unit are given in Section C.6.

C.4.3.2 Effect of Chaining on the Single Channel Control Unit

C.4.3.2.1 Data Chaining

Data Chaining is the Channel operation most likely to cause an overrun on DCUs attached to the SCCU. This is because data can continue to pass between the DCU and its device while the SCCU is executing the Chain.

The number of buffers of byte storage required in the DCU to prevent an overrun during chaining can be determined approximately by the equation

$$N > Cs\ (D + t) + B \qquad C(i)$$

where N is the number of buffers
Cs is the Data Chaining time (usecs)
D is the throughput rate of the device (bytes/usec)
t is the tolerance on D
B is the number of buffers full (reading) or empty (writing) at the beginning of chaining.

Example: Consider a Data Chain on an 820KB/sec Drum. The time for a Data Chain with Transfer in Channel is 6.96 usecs (see Section C.6). Equation C(i) therefore shows that

$$N > 6.96 \times 0.82 = 5.7$$

This assumes t = 0 and B = 0.

The number of buffers available in a Drum DCU is 6. The calculated value of N is dangerously close to this limit and may easily exceed it if the tolerance on the 820KB/sec is included or if main store access delays increase the length of the chaining operation.

N.B. The conclusion drawn is that Data Chaining on an 820KB/sec Drum is not recommended.

Having completed a Data Chain, the SCCU can resume normal byte servicing and must also clear/fill the extra DCU buffers filled/emptied during chaining. If sufficient time is not allowed for this 'catch-up' between Data Chains, then the value of B in equation C(i) is increased for subsequent chaining operations and N is more likely to exceed the number of buffers available.

C.4.3.2.1 (continued)

'Catch-up' time can be determined approximately from the following :

Let x be the 'catch-up' time measured from the beginning of the Chaining operation.

In time x, (D + t)x bytes are transferred between the DCU and its device.

Of the period x, (x - Cs) usecs are available for byte servicing

$$\therefore \frac{(x - Cs)}{S} = x(D+t) + B$$

where S is the time required to service one byte (usecs) and D, t, B and Cs are as defined for equation C(i).

$$\therefore x = \frac{Cs + SB}{1 - S(D+t)} \qquad C(ii)$$

A graph of x against (D+t) is plotted on page C18 assuming that B = 0 and using values of Cs and S quoted in Section C.6.

For a given device throughput rate, the minimum separation of Data Chains can be determined approximately from this graph. Any increase in Cs or S by store interference will produce a corresponding increase in x.

C.4.3.2.2 Command Chaining

A Command Chaining operation is less likely to lead to a DCU overrun condition on the SCCU. A Command Chain occurs between data blocks when data flow between the device and the DCU is considerably diminished.

If the Command Chain operation is lengthened or delayed by queues for store access, however, the time taken to complete the operation may exceed that allowed by the DCU. The time allowed for Command Chains depends on the DCU and figures are quoted in the appropriate DCU specifications.

C.4.3.3 Effect of Chaining Operations on the Multichannel Control Unit(MCCU)

Whereas servicing of trunks on the SCCU is non-concurrent, the trunks on an MCCU can be serviced concurrently and the effects of chaining operations are more complex. A Data or Command Chain performed on behalf of any trunk takes precedence over normal byte servicing for all the other trunks, regardless of their relative priorities; furthermore it is possible for chaining operations to occur on two or more trunks in succession, in which case an overrun of buffers in the fastest DCUs is more likely.

C.4.3.3 (continued)

Having completed a chaining operation, the MCCU must clear/fill the DCU buffers filled/emptied during the operation. This 'catch-up' operation begins with the highest priority trunks and servicing of lower priority trunks is held up until more prior DCUs require no more byte transfers between themselves and the MCCU.

The effect of chaining on a particular MCCU trunk therefore depends on the priority of the trunk, the total throughput rate of more prior trunks, the duration of the chaining operation(s) and the number of buffers the attached DCU has available.

It can be shown, by the method used in Section C.4.3.2.1, that the (n + 1)th priority trunk must wait approximately W_n usecs from the start of chaining before it is serviced, where :

$$W_n = \frac{Cm + Sm \sum_{1}^{n} Br}{1 - Sm \sum_{1}^{n} (Dr + tr)} \qquad 1 \leq r \leq n \qquad C(iii)$$

Br is the number of DCU buffers full (reading) or empty (writing) at the start of chaining.
Dr is the throughput rate of device on trunk r (bytes/usec).
tr is the tolerance on Dr
Sm is the average byte servicing time for the MCCU (usec)
Cm is the duration of the chaining operation(s) (usec); two or more may occur in succession.

By calculating the number of bytes transferred by the device on trunk (n + 1) in time W_n and comparing this with the number of buffers available in its DCU, equation C(iii) can be used to make a rough assessment of the likelihood of an overrun on DCU (n + 1). However, the equation is probably more useful in showing the relationship between parameters. The following observations can be made :

(a) Lower priority DCUs are more likely to overrun on heavily loaded MCCU systems. In such systems the term Sm $\sum^{n} (Dr + tr)$ can approach '1', increasing the value of W_n.

(b) Data Chains anywhere in the system are likely to have the most serious effect because data can still be transferred between all the DCUs and their devices during Data Chaining. In this case, the term $\sum^{n}$ (Dr + tr) is the sum of throughput rates of all more prior devices.

(c) If a Command Chain is performed for a trunk, the amount of data passed by that trunk during chaining and for a period subsequent to chaining is diminished. If the Command Chain is performed for a trunk more prior than the (n + 1)th, the (Dr + tr) of the attached device can be subtracted from the

C.4.3.3 (continued)

term $\sum^{n}$ (Dr + tr) and the value of W_n is reduced. This relaxation does not apply if the Command Chain is performed for a less prior trunk.

(d) The likelihood of overrun is increased if two or more chaining operations occur in succession, i.e. Cm is significantly increased.

The time to 'catch-up' after chaining, i.e. resume normal servicing of a trunk up to and including the (n + 1)th is given approximately by equation C(iii) summed over n + 1 instead of n, i.e. W_{n+1}. By setting n + 1 equal to the number of trunks attached to the MCCU, W_{n+1} could give a measure of the 'catch-up' time of the whole MCCU system. Unlike equation C(ii), however, this cannot be used to define a minimum separation of chaining operations if more than one trunk is fitted; the incidence of chaining operations for one trunk can be independent of the incidence of other trunks and their separation is therefore indeterminate. The equation for W_{n+1} does show, however, that heavily loaded systems can take a long time to recover (in the limit of a fully loaded system, i.e. $\frac{1}{Sm} = \sum^{n+1}$ Dr + tr, W_{n+1} is infinite) increasing the chances of an overrun during subsequent chaining. The conclusion drawn is that chaining, particularly Data Chaining, on MCCUs loaded at or near their performance limit, is likely to lead to overruns.

C.4.3.4 Effect of Chaining Operations on MXCU

The time the MXCU requires to perform a chaining operation is comparable with the average byte servicing time; the number of bytes processed by the attached DCUs during chaining and 'catch-up' is therefore much less than on the SCCU and MCCU. Unless used excessively, the effect of chaining on MXCU performance is much less significant than on the SCCU and MCCU.

C.4.3.5 Conclusion

Some quantitative measure of the effect of chaining on the SCCU has been given and the effect on MXCU performance is not likely to be significant, unless used to excess.

Meaningful quantitative descriptions of chaining on the MCCU are difficult because chains can occur on a number of trunks in succession or at indeterminate intervals. The conclusion drawn is that although Command Chaining has to be employed, Data Chaining should be used sparingly on heavily loaded MCCU configurations.

C.4.4 End Servicing without Chaining

C.4.4.1 Introduction

A DCU reports the completion of an operation by raising Service Request and End. The Channel Control Unit responds to this

C.4.4.1 (continued)

condition by executing a number of sequence steps during which time byte servicing is interrupted. If Command Chaining is to follow, the End Service time is added to the Chaining time. End Servicing times without chaining are given in Section C.6.

C.4.4.2 Effect on SCCU

End servicing has no effect on SCCU performance. While End Servicing is being performed no byte servicing is required.

C.4.4.3 Effect on MCCU

End servicing has the same effect on MCCU performance as a chaining operation. Servicing of bytes from other trunks is interrupted. However because one trunk has completed an operation its data throughput rate has fallen to zero and the MCCU can more easily cope with the remaining throughput rate after the End Service sequence.

C.4.4.4 Effect on MXCU

The effect of End Servicing on the MXCU is not significant because the time required for an End Service is small compared with the byte servicing time.

C.4.5 Effects of Servicing Central Control Unit Requests

C.4.5.1 Introduction

Channel Control Units have two hardware sequences for responding to the privileged instructions Start Device (SDV), Test Device (TDV), Halt Device (HDV) and the hardware instruction Service Interrupt. Execution of the two sequences is separated by a period in which the Channel waits for a 'Ready' response from the appropriate DCU. The sum of the sequence times and the 'waiting for Ready' time is termed the Channel Response Time, CRT (see Section C.5.3.2).

The first hardware sequence broadcasts device addresses or, for Service Interrupt, the hardware command Who Are You. The second sequence transmits a command to the DCU which has responded with 'Ready'. Response to the instructions SDV, TDV and HDV under certain conditions (see Section 6.8) and to privileged instruction Check Channel (CKC) only involves the first sequence.

A Channel Control Unit can continue to service bytes in the 'waiting for Ready' period but not while its sequences are broadcasting addresses or commands. Times for these sequences are listed in Section C.6.

C.4.5.2 Effect on SCCU

If the SCCU is busy servicing a trunk, response to SDV, TDV and CKC only involves the setting of a condition code and is short.

Response to Halt Device involves both addressing of a DCU and transmission of a command to the DCU. The response time is long and could cause DCU overrun but because the DCU is being halted, this is irrelevant.

Response to Service Interrupt is also long and could cause an overrun on a fast DCU. While servicing DCUs with a throughput rate greater than 700KB/sec, however, SCCU sequences are fully occupied with byte servicing and the sequences for responding to any processor requests cannot be started; byte servicing is therefore not interrupted and an overrun of the DCU cannot occur.

C.4.5.3 Effect on MCCU

Full Channel Responses to privileged instructions (except CKC) and Service Interrupt can be longer than Command Chain time and the resulting interruption to byte servicing could lead to overruns on attached DCUs. However, a restriction similar to that on the SCCU may limit the interruption of byte servicing on heavily loaded MCCUs; if the total throughput rate of the MCCU is greater than 500KB/sec the MCCU sequences can be exclusively occupied with byte servicing and sequences for response to processor requests are then unable to start. The effect of this restriction on the Central Control Unit is discussed in Section C.5.

C.4.5.4 Effect on MXCU

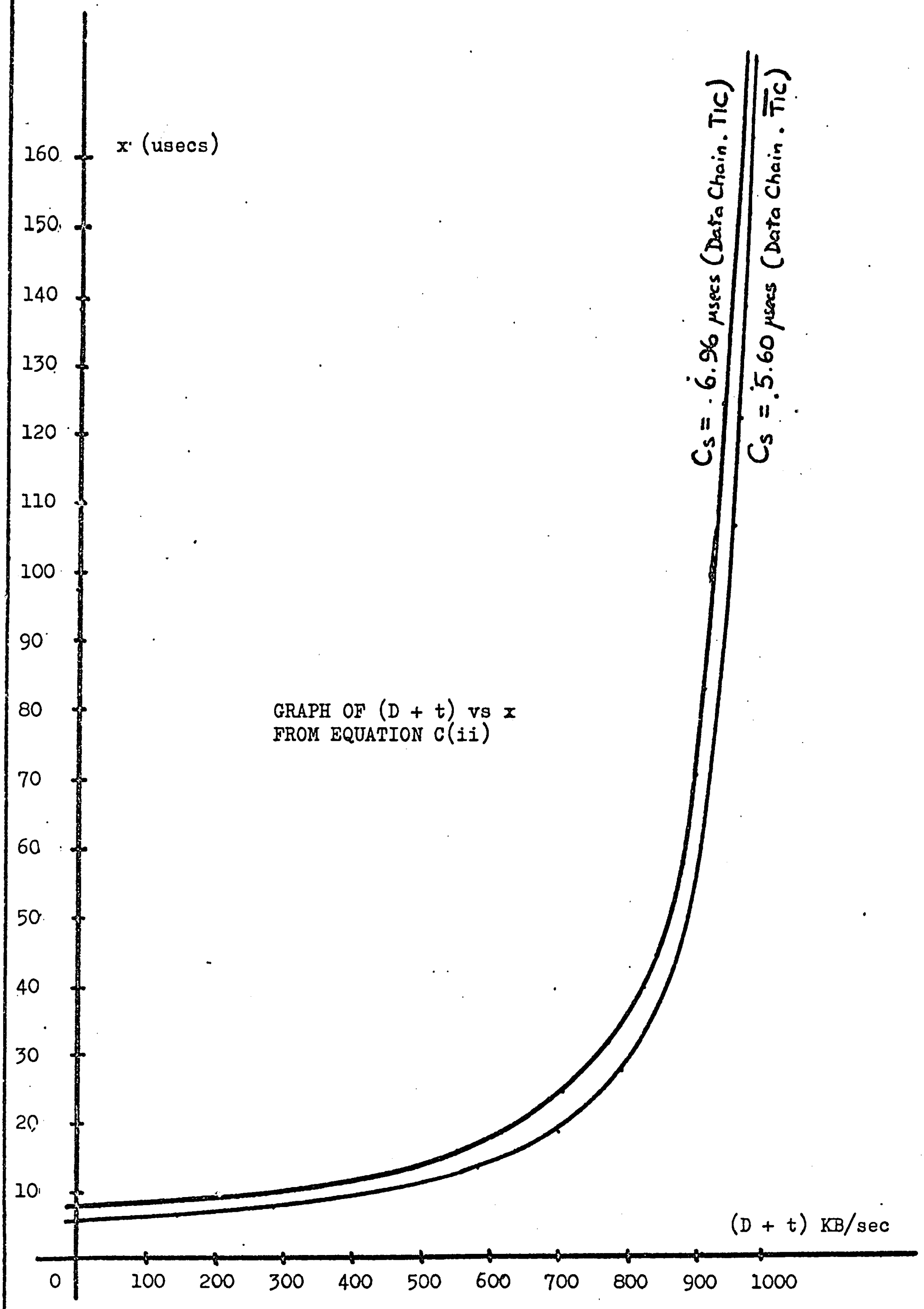
Response times to processor requests on the MXCU are comparable with the byte servicing time and single processor requests will not have a significant effect on MXCU performance. As requests increase in frequency then the throughput of bytes in the MXCU will show a corresponding degradation.

C.4.5.5 Ready

Channel Control Units respond at a lower priority to Ready than to Service Requests; hence on a busy Channel Control Unit, the processor delay can be extended and may last until the termination of one of the current operations. (See Section C.5.3).

C.4.5.6 Program Controlled Interrupt

The detection of a Program Controlled Interrupt results in the execution of a Service Interrupt command for which the duration of Channel response can exceed that for chaining operations (See Section C.6.) with the consequent risk of overruns. Excessive use of the PCI should therefore be avoided.



GRAPH OF (D + t) vs x
FROM EQUATION C(ii)

C.5 DEGRADATIONS TO CENTRAL CONTROL UNIT PERFORMANCE FROM PERIPHERAL CONTROL SYSTEM

C.5.1 Introduction

Operation of the Central Control Unit can be delayed by operation of the Peripheral Control System. The delays can arise from conflict in main store accessing or from delays in Channel response to Processor originated requests. This section describes the sources of delay and the values of some factors contributing to the delays are listed in Section C.6.

C.5.2 Store Access Delays

Central Control Unit requests for store accesses are routed through a Store Access Unit. This Store Access Unit competes with the I/O Store Multiplexor for access to the Main Store and in cases where there is a conflict for the same store module, the I/O system is given priority, thus delaying the Central Control Unit. No quantitative measure of the delay is available.

C.5.3 Channel Response Times to Control Unit requests

C.5.3.1 Definition of Channel Response Time (CRT)

Channel Control Units have two hardware sequences for responding to Central Control Unit requests. The execution of the sequences is separated by a 'waiting for Ready' period. The first sequence, termed pre-Ready, broadcasts addresses or the hardware command Who Are You. The second sequence, termed post-Ready, responds to the Ready return and transmits a command to the responding DCU.

The Central Control Unit is interlocked to the Channel Control Unit from initiation of the processor request until the latter has completed its response. The duration of the interlock is defined as the Channel Responśe Time, CRT.

Full CRT is comprised of five factors in the following sequence :-

a) An indeterminate delay before entering the Channel Control Unit pre-Ready sequence

b) A defined time, for execution of the pre-Ready sequence

c) A defined time for the appropriate DCU to return a Ready signal; this time is termed the Device Response Time, DRT.

d) An indeterminate delay before entering the Channel Control Unit post-Ready sequence

e) A defined time for execution of the post-Ready sequence

The duration of delays (a) and (d) depends on the loading of the Channel Control Units; the reasons for the delays are discussed in the following sub-sections.

C.5.3.1 Continued.

Device Response Times are defined in the appropriate DCU specifications.

Channel Control Unit pre-Ready and post-Ready sequence times are listed in Section C.6.

Note: Response to Start Device, Test Device and Halt Device under certain conditions (see Section 6.8) and to privileged instruction Check Channel, only involves factors (a) and (b).

C.5.3.2 Single Channel Control Unit

If the SCCU is not busy and if there is no other reason to reject a processor request, Channel Response Time is the sum of the appropriate SCCU sequences and the Device Response Time. If the SCCU is servicing a trunk, Channel response to Service Requests (i.e. byte servicing) takes precedence over response to both processor requests and Ready returns. Indeterminate delays are therefore incurred before entering the SCCU sequences for handling the processor requests and the Ready return.

If the SCCU is servicing a DCU with a throughput rate greater than 700 KB/sec it is fully occupied with byte servicing and is unable to deal with some processor requests at all. Simple condition code responses to SDV, TDV and CKC can be made but response to HDV and Service Interrupt, which require action by SCCU sequences, is delayed until the throughput rate falls below 700 KB/sec. This means, for instance, that a Halt Device or Service Interrupt instruction addressed to an SCCU servicing an 820 KB/sec Drum can delay Central Control Unit operation until the end of the current data block is reached.

C.5.3.3 Multichannel Control Unit

If the MCCU is not busy, responses to processor requests are not delayed. CRT is therefore the sum of the appropriate sequence times and the Device Response Time.

If the MCCU is busy, byte servicing takes precedence over servicing processor requests and Ready returns. As for the SCCU, indeterminate delays will be incurred in entering the appropriate MCCU sequences.

If the MCCU is servicing a load of 500 KB/sec or greater and if Service Requests are received from the load at regularly spaced intervals the MCCU can be locked into the byte servicing sequence, excluding response to processor requests. However, if more than one trunk is being serviced, the incidence of Service Requests is spasmodic and byte services are bunched leaving some time available for starting processor request responses. Thus, although it is unlikely that these responses are completely inhibited, the delay to the Central Control Unit can be extended significantly as the MCCU load increases from 500 KB/sec.

C.5.3.4 Multiplexor Channel Control Unit

As for the SCCU and MCCU, if the MXCU is not busy, Channel Response time depends on the length of the appropriate MXCU sequences and the response time of the addressed DCU in sending Ready.

If the MXCU is busy, Service Requests take precedence over processor requests and Ready returns and the Central Control Unit can be delayed an indeterminate length of time depending on the loading of the MXCU.

C.6 **Peripheral Control System performance figures**

C.6.1 **Section Introduction**

The figures quoted in this section are calculated from the shortest possible execution times of the appropriate Channel Control Unit sequences. Any sequences involving store access can be lengthened by interference at store from other Channel Control Units or the Central Control Unit.

C.6.2 **Single Channel Control Unit**

C.6.2.1 **Byte servicing**

C.6.2.1.1 **Introduction**

Two SCCU sequences are involved in byte servicing. For fast devices, a fast servicing loop is provided which is traversed once for every byte transferred. If the rate of Service Requests is not high enough to fully occupy the fast servicing loop, an Idle sequence is entered between byte services and average servicing time is increased.

C.6.2.1.2 **Byte servicing times and throughput rate**

Fast Service Request loop	800 nsecs
Additional time required for Store accesses	800 nsecs
Average byte servicing time (800 + ¼ x 800)	1000 nsecs
Maximum SCCU Throughput rate (reciprocal of 1,000 nsec)	1 Mb/sec
Average byte servicing time if Idle loop is entered (i.e. time to traverse Idle loop + fast servicing loop)	1430 nsecs

Note: Certain processor requests, requiring action by CCU sequences, are only honoured in the Idle loop and only then if a Service Request is not detected while traversing the Idle loop. Therefore, if the rate of Service Requests is greater than or equal to 1 every 1430 nsecs, then Processor Requests are delayed. This rate of Service Requests is equivalent to a throughput of 700 KB/sec.

C.6.2.2 **Data Chaining times**

Note: Data Chaining time is the time required to service the last byte of a block and then execute the chain. The times below do not include the byte servicing time.

Data Chain times (extra added to byte servicing times):	usecs
Basic Data Chaining sequence (3 store accesses)	4.65
Additional time required for Transfer in Channel (1 store access)	1.36

*

C.6.2.3 **End Servicing**

End Servicing with no Command Chaining	usecs
Basic Time (Write)	2.52
Basic Time (Read - 1 Store access)	3.28

C.6.2.3 Continued.

Additional times for Command Chaining:

	usecs
Basic time (2 store accesses)	3.04

Additional time if:

Previous transfer was a Read (1 store access)	0.76
Next transfer is a Write (1 store access)	0.72
Status Modifier is set	0.24
Transfer in Channel (1 store access)	1.36

C.6.2.4 Response to Central Control Unit Requests

C.6.2.4.1 Introduction

The figures quoted in this subsection are the sums of the pre-Ready and post-Ready sequence times (see Section C.5.3.1).

C.6.2.4.2 SCCU Sequence times

Start Device:

If Condition Code (CC) = 0 and Read (3 store accesses)	8.54
If Condition Code (CC) = 0 and Write (4 store accesses)	9.38
CC = 1 (4 store accesses)	8.86 *
CC = 2	0.60

Test Device:

CC = 0	5.18
CC = 1 (1 store access)	5.50
CC = 2	0.60

Check Channel:

CC = 0,1	0.84
CC = 2	0.60

Halt Device:

CC = 0	0.84
CC = 1 (1 store access)	5.48
CC = 2	4.92

Service Interrupt:

2 store accesses	6.60

C.6.3 Multichannel Control Unit

C.6.3.1 Byte servicing

C.6.3.1.1 Introduction

Two MCCU sequences are involved with byte servicing. The MCCU has the ability to overlap servicing of bytes from two or more trunks; while servicing bytes in this mode, the MCCU sequences in a Service Request loop. If overlapping of services is not possible (e.g. two or more bytes from the same trunk require consecutive services) or if the rate of Service Requests is not high enough to fully occupy the Service Request loop, the MCCU enters an Idle loop.

C.6.3.1.2 Byte servicing times and Throughput rate

	usec
Length of Idle loop	0.48
Bytes of Service Request loop	1.26
Extra time required for store access	1.02

Average byte servicing time with full overlapping
(1.26 + ¼ x 1.02) = 1.515 usec

Maximum MCCU throughput rate with full overlapping
(reciprocal of 1.515) = 650 KB/sec

Average byte servicing time with no overlapping
(1.26 + ¼ x 1.02 + 0.48) = 1.995 usec

Maximum MCCU throughput rate with no overlapping
(reciprocal of 1.995) = 500 KB/sec

Note: Central Control Unit requests, requiring action by MCCU sequences, are only honoured in the Idle loop and only then if a Service Request is not detected while traversing the Idle loop. Therefore, if Service Requests are received at regular intervals and at a rate greater than or equal to every 1.995 usecs, processor requests are delayed. This rate of Service Requests is equivalent to a throughput rate of 500 KB/sec.

C.6.3.2 Data Chaining times (additional to byte servicing times)

		usec
Basic		4.42
Extra for storing part of word)	i.e. Chain does not start/	1.02
Extra for fetching part of word)	finish on word boundary on output input	1.04
Extra for Transfer in Channel		1.26

C.6.3.3 End Servicing

End servicing with Device End not set	3.72

End servicing with Device End set but no Command Chaining:

Basic (no store access)	4.46
Extra for storing part of last word (only on input if do not end on word boundary)	1.36

C.6.3.3 Continued.

Additional times for Command Chaining:	usec
Basic	4.18
Extra for fetching part of word (only on output if do not start on word boundary)	1.04
Extra if Status Modifier set	0.82
Extra if Transfer in Channel	1.26

C.6.3.4 Response to Central Control Unit requests

Times are quoted separately for the pre-Ready sequence and the post-Ready sequence (see Section C.5.3.1).

Pre-Ready

	usec
Start Device (CC = 0)	3.06
Start Device (CC = 2)	1.40
Test Device (CC = 0)	3.06 *
Test Device (CC = 2)	1.40
Halt Device (CC = 0)	1.40
Halt Device (CC = 2)	3.06
Check Channel	1.40
Service Interrupt	4.08

Post-Ready

Start Device (CC ≥ 0)	11.92
Test Device (CC = 0)	4.76
Halt Device (CC = 2)	4.76
Service Interrupt	5.54

C.6.4 Multiplexor Channel Control Unit

C.6.4.1 Byte servicing time

	usec
All times assume a store access time seen from MXCU	0.9
Normal mode	7.02
Miniburst mode - first byte	4.86
Miniburst mode - subsequent bytes	2.02

C.6.4.2 Data Chaining times (additional to byte servicing times)

Basic	5.04
Additional for Transfer in Channel	1.26

C.6.4.3 End Servicing times

	usec
End with no Device End or Device End and no Command Chain	5.48
Extra for Command Chain	5.40
Extra for Transfer in Channel	1.26

C.6.4.4 Response to Central Control Unit requests

Times are quoted separately for the pre-Ready and post-Ready sequences. (See Section C.5.3).

Pre-Ready

Start Device	1.44
Test Device	1.44
Halt Device	1.44
Check Channel	1.50
Service Interrupt	8.12

Post-Ready

Start Device:

CC = 0	9.86
CC = 1	10.34
CC = 2	7.28
CC = 3 (Ready not received or two Readies)	6.00
CC = 3	7.04

Test Device:

CC = 0	3.74
CC = 1	6.80
CC = 2	3.74
CC = 3 (No Ready or two.Readies)	2.70
CC = 3	3.50

Halt Device:

CC = 0	3.74
CC = 1	6.80
CC = 2	3.74
CC = 3 (No Ready or two Readies)	2.70
CC = 3	3.50

Service Interrupt	2.9

D. PRIVILEGED INSTRUCTIONS

D.1 Introduction

The instruction set of the 4-70 processor includes a number of instructions which have processor control functions and which may only be executed when the processor is operating in the Privileged Mode. These instructions are specified in this Appendix.

D.2 Instructions

D.2.1 INSTRUCTION NAME : Start Device

OPERATION CODE : 9C

FORMAT : SI

DESCRIPTION :

The Start Device instruction is used to initiate all peripheral operations. The rightmost twelve bits of the sum of the contents of the register specified by B_1 and of the D_1 field specify the channel (upper four bits) and device on that channel (lower eight bits) on which the operation is to take place. Other bits of this sum, and the contents of the I_2 field, are ignored.

Prior to the operation, the Channel Command Words (CCW) to be executed by the sub-channel involved, and the Channel Address Word (CAW - bytes 72-75), containing the protection key to be used by the sub-channel and the address of the first CCW, must be loaded.

The instruction is complete when the peripheral operation has been either initiated successfully or suppressed, which will be indicated by the setting of the Condition Code on completion. The CAW may be altered as soon as the instruction is completed.

The formats of the CAW and CCWs are given in Section 6.2.

Status information may be stored in bytes 92 and 93 of the Channel Status Word (CSW) if the peripheral operation is not successfully initiated. This is indicated by CC=1. All peripheral interrupts must be inhibited while the instruction is being executed, until the status information (if any) stored by the instruction has been removed from the CSW, to guard against the possibility of the latter being overwritten by an interruption.

CONDITION CODES :

CC = 0 : Operation initiated and proceeding.

CC = 1 : Operation not initiated.
Status stored in SDB and CSB fields of CSW.

CC = 2 : Operation not initiated, because sub-channel is executing commands or has a termination interrupt pending.

CC = 3 : Operation not initiated because channel/sub-channel is inoperable.

NOTES :

(1) In these notes, expressions such as "DB = 1" or "PGC = 0" refer to the presence or absence of the appropriate conditions in the addressed device/sub-channel rather than to the current values of these status bits in the Channel Status Word. These conditions are described in Section 6.4.

D.2.1 (continued)

(2) CC = 3 may be set for any of the following reasons :-

- No channel response (e.g. because incorrect channel number was specified).
- No response from device control unit (e.g. because incorrect device number was specified) or more than one response.
- The 'freak' condition where TIP, DB, CB, DE and INOP all = 0.

INOP = 1 will not set CC = 3.

(3) CB = 1 is a sufficient condition for CC = 2; DB = 1, CB = 0 is not.

(4) For successful execution, setting CC = 0, it is necessary for all of MR, TIP, DB, CB, CCC and PGC to be zero, and, unless the first command is Sense, DE must = 1 and INOP = 0. If this is not the case, and the conditions for CC = 2 or 3 are absent, SDB and CSW are stored and CC set = 1.

(5) CC may be set = 1, with MR = 1 in the SDB, if a Manual Request condition is present in another device attached to the same sub-channel or control unit. The identity of this device cannot be determined until the Manual Request interrupt occurs; meanwhile peripheral operations cannot be initiated on other devices on the same sub-channel or control unit.

(6) All CSB conditions are cleared at the start of the instruction. Possible reasons for CCC or PGC to be set in the course of executing the instruction, setting CC = 1, are listed in Sections 6.4.4.4 and 6.4.4.7. A first command code which is acceptable to the sub-channel, but not to the device, will not set CC = 1, but will subsequently terminate the operation with the SI bit set on interruption.

(7) It is recommended that, following a Start Device which sets CC = 1, the status bits should be examined in the following sequence :-

CCC; PGC: MR (see Note (5)); INOP/SI; DB.

D.2.2 INSTRUCTION NAME : Halt Device

OPERATION CODE : 9E

FORMAT : SI

DESCRIPTION :

The Halt Device instruction is used to forcibly terminate a peripheral operation on a specified sub-channel. The rightmost twelve bits of the sum of the contents of the register specified by B_1 and of the D_1 field specify the channel (upper four bits) and device on that channel (lower eight bits), which are used to identify the sub-channel. Other bits of this sum, and the contents of the I_2 field, are ignored.

On a selector channel, the specified device number is not required to identify the (single) sub-channel, and is ignored. Where a selector CCU requires to use a device number (e.g. in order to signal a control unit) it uses the number of the last device involved in a peripheral operation on the channel.

No control information (CAW,etc) is required by the instruction.

The.instruction is complete when the attempt to terminate the peripheral operation has been made; the success or failure of the attempt is indicated by the Condition Code setting. Termination of the operation will subsequently lead to a normal termination interrupt.

Status information may be stored in bytes 92 and 93 of the Channel Status Word by the instruction. This is indicated by CC = 1. This should only occur on a multiplexor channel, and only byte 92 (SDB) will be significant. All peripheral interrupts must be inhibited while the instruction is being executed, until the status information (if any) stored by the instruction has been removed from the CSW, to guard against the possibility of the latter being overwritten by an interruption.

CONDITION CODES :

CC = 0 : No operation terminated because specified sub-channel was not involved in an operation, or has a termination interrupt pending.

CC = 1 : No operation terminated. Consult SDB. (On a selector channel CC = 1 indicates a hardware fault condition).

CC = 2 : Operation on specified sub-channel terminated.

CC = 3 : No operation terminated because channel/sub-channel is inoperable.

D.2.2 (continued)

NOTES :

(1) Notes (1) and (2) of Start Device (D.2.1) apply.

(2) The interpretation of SDB following CC = 1, when a multiplexor channel is involved, is as for Start Device.

(3) No special indication of the cause of termination is given when a peripheral operation is terminated by Halt Device. IL may be present even if the Suppress Length Indicator flag is set in the last CCW.

D.2.3 INSTRUCTION NAME : Test Device

OPERATION CODE : 9D

FORMAT : SI

DESCRIPTION :

The Test Device instruction is used to examine the status of the specified device, in particular to determine what its response would be to a Start Device instruction. The rightmost twelve bits of the sum of the contents of the register specified by B_1 and of the D_1 field specify the channel (upper four bits) and device on that channel (lower eight bits) whose status is to be examined. Other bits of the sum, and the contents of the I_2 field, are ignored.

No control information (CAW, etc) is required by the instruction.

The instruction is complete when the Condition Code has been set and, if CC = 1, status information has been stored in bytes 92 and 93 of the Channel Status Word. Only byte 92 (SDB) is significant. All peripheral interrupts must be inhibited while the instruction is being executed, until the status information (if any) stored by the instruction has been removed from the CSW, to guard against the possibility of the latter being overwritten by an interruption.

CONDITION CODES :

CC = 0 : Device is available and can accept another operation.

CC = 1 : Device is not available. Consult SDB.

CC = 2 : Device is not available because sub-channel is executing commands or has a termination interrupt pending.

CC = 3 : Device is not available because channel/sub-channel is inoperable.

NOTES :

(1) Notes (1)-(5) of Start Device (D.2.1) apply.

(2) If CC = 1, CSB should be ignored. SDB bits should be examined in this order : MR (see Note (5) of Start Device); INOP/SI; DB.

(3) The instruction differs from the IBM 360 "Test I/O" in that it cannot be used to clear pending interruptions, and when CC = 1 it does not store the complete CSW.

D.2.4 INSTRUCTION NAME : Check Channel

OPERATION CODE : 9F

FORMAT : SI

DESCRIPTION :

The Check Channel instruction is used to examine the status of a channel. The channel number is specified in the leftmost four out of the rightmost twelve bits of the sum of the contents of the register specified by B_1 and of the D_1 field. All but these four bits of the sum, and the contents of the I_2 field, are ignored.

No control information (CAW, etc) is required.

The instruction is complete when the Condition Code has been set. The contents of the CSW are not altered under any circumstances.

CONDITION CODES :

CC = 0 : Channel is available.

CC = 1 : (Selector channel only) Operation complete; termination interrupt pending.

CC = 2 : (Selector channel only) Channel is executing commands.

CC = 3 : Channel is inoperable (or non-existent channel specified.)

Issue	Date	Auth'y	Log	Title	Document No.
4/1	10.7.68	MRW	809	CENTRAL PROCESSOR MODELS 4-70/75	PS 4.10.70
					Page D5

D.2.5 INSTRUCTION NAME : Load Scratchpad

OPERATION CODE : D8

FORMAT : SS

DESCRIPTION :

Execution of 'Load Scratchpad' causes consecutively-addressed 32-bit words to be transferred from main store to consecutively-addressed locations in the Scratchpad. The first and second operand addresses specify the initial locations in Scratchpad and main store, respectively. The number of words transferred is specified in the L field.

CONDITION CODE : Unaltered.

NOTES :

(1) The number of 32-bit words transferred is one greater than the quantity (0-255) in the L field.

(2) All but the least significant 8 bits of the first address are ignored; they specify a Scratchpad word location number in the range 0-255.

(3) The addressing of non-existent Scratchpad words with addresses below 128 has no effect.

(4) On a 4-70, and on a 4-75 with the engineers switch reset to inhibit paging, reference at any stage of execution to a Scratchpad address above 127 will cause a Data Error interrupt. The instruction is terminated.

(5) On a 4-75 with the engineers switch set to permit paging, reference to any Scratchpad address in the range 128-255 will cause access to the Associative Memory (see Section 19.2.2.3). Should the count of words still not be exhausted after reference to address 255, "wrap around" occurs, the next Scratchpad word accessed being location 0. The state of the Paging Mode indicator bit is irrelevant.

(6) The second address is a byte address and must be a multiple of 4, otherwise Address Error interrupt occurs and execution is terminated.

(7) If a location outside the available main store is addressed at any time during execution, Address Error interrupt occurs and execution is terminated.

D.2.6 INSTRUCTION NAME : Store Scratchpad

OPERATION CODE : DO

FORMAT : SS

DESCRIPTION :

Execution of 'Store Scratchpad' causes 32-bit words to be transferred from consecutively-addressed locations in the Scratchpad to consecutively-addressed locations in main store. The first and the second operand addresses specify the initial location in Scratchpad and main store, respectively. The number of words transferred is specified in the L field.

CONDITION CODE : Unaltered.

NOTES :

(1) (2) As for 'Load Scratchpad' (D.2.5)

(3) The addressing of non-existent Scratchpad words with addresses below 128 causes zero words to be loaded into main store.

(4)-(7) As for 'Load Scratchpad' (D.2.5).

D.2.7 INSTRUCTION NAME : Program Control

OPERATION CODE : 82

FORMAT : SI

DESCRIPTION :

Execution of the 'Program Control' instruction causes the termination of program execution in the present state and initiation of another state. The action is controlled by the contents of the Immediate Field. The address specified by the instruction is placed in the Program Counter of the terminated state so that when the latter is re-initiated it will resume at that address.

CONDITION CODE :

The code of the terminated state is preserved in the Program Counter storage register of that state. The code of the initiated state is loaded from the Program Counter storage register of that state.

NOTES :

(1) The 8-bit Immediate Field is partitioned into 4 sub-fields. From left to right these are :-

(a) A 3-bit unused portion which must be zero. (This is a program restriction).

(b) The 1-bit Test Mode Control Flag : see Section 5.3.5 for the definition of the action of this flag.

(c) The 3-bit Direct State Initiation Field : if the rightmost bit of the Immediate operand is zero, the state initiated is P1, P2, P3 or P4 according to whether this field contains x11, x10, x01 or x00, respectively (x = 0 or 1; programs should use x = 0).

(d) The 1-bit Specification Control Flag :
F=1 specifies that the new state is determined by the ISI field of the ISR of the terminated state.
F=0 specifies that the new state is determined by bits 5 and 6 of the Immediate Operand, according to (c) above.

(2) The effect that the execution of this instruction has on the Interrupt Control System is described in Section 5.3.5.

(3) An 'Address Error' interrupt will occur if the specified address is not on a halfword boundary.

D.2.8 INSTRUCTION NAME : Idle/No-Op

OPERATION CODE : 80

FORMAT : SI

DESCRIPTION :

Execution of the 'Idle/No-Op' instruction effects an Idle Mode within the Central Control Unit by continuously branching back to itself. The contents of the Immediate Field are displayed on the Engineer's Console.

CONDITION CODE : Unaltered.

NOTES :

(1) When this instruction is being executed the Idle Indicator on the Console is lit.

(2) Any interrupt requested whilst the Idle Mode is in effect will be executed if it is not masked. The Idle Indicator is then removed and the Central Control Unit is no longer in the Idle Mode.

(3) The B and D fields of this instruction are ignored. *

D.2.9 INSTRUCTION NAME : Diagnose

OPERATION CODE : 83

FORMAT : SI

D.2.9.1 <u>The Central Control Unit</u>

<u>General</u>

Execution of the 'Diagnose' instruction affects the subsequent action of either the Central Control Unit or a Multichannel Control Unit. In the former case, which applies whenever the instruction is not followed by an input-output instruction, the Central Control Unit is set into the 'Diagnose Mode', and a counter is loaded with the contents of the least significant 12 bits of the instruction-specified address. Subsequently the next instruction is fetched and, as it is executed, the previously loaded counter is decremented by one after each sequence step of the instruction is executed. When the counter reaches zero, the instruction is caused to terminate after the next step, when the 'Diagnose Snapshot' operation is initiated. As a result of this operation the contents of a specified set of registers are dumped into main store locations 32 to 60. The subsequent operation of the Central Control Unit is normal and commences with the next instruction. The Immediate field of this instruction and the bits in the instruction-specified address above the rightmost twelve, are not used.

CONDITION CODE : Dependent upon the object instruction.

PROGRAM INTERRUPTIONS :

Interrupts may be requested as a result of the execution of both the 'Diagnose' and the object instruction

NOTES :

(1) The initial count must be at least 3 for an SS instruction and at least 2 for any other type of instruction.

(2) If the 'Snapshot' occurs after the very last step of the instruction, one or two succeeding instructions may be skipped. To get around this, two 'dummy' (not SS) instructions should be placed after the one diagnosed.

(3) If interruption does not occur, and the count is still non-zero when the last step of the instruction is executed, the snapshot is inhibited, the Diagnose Mode is reset and the next instruction is fetched and executed.

(4) If an unmasked interrupt occurs before the count reaches zero, the interrupt is serviced up to, and including, the sequence step during which the count equals zero. The 'Diagnose Snapshot'is then initiated. If the count is still non-zero when the last sequence step in the interrupt sequence is executed, the 'Snapshot'

D.2.9.1 (continued)

operation is inhibited, the Diagnose Mode is reset, and the next instruction is fetched and executed. If the count becomes zero before one of the first five steps of the interrupt sequence, an error is likely to occur and the Snap-Shot may not be executed.

(5) P4 interrupts should always be masked when 'Diagnose' is executed. Unless all other interrupts are masked, 'Diagnose' should only be executed in P3 state.

(6) The format of the information placed in main store by the diagnostic snapshot operation, is as follows:-

Word 1 (bytes 32-35) contains the following 1 bit markers: CYIN, CYO, ADD, FO, F1, AMF, NIC(0-2), CAM, CC1(0-1), PAR, AE(0-4), CTI(0-2), MK1-6, MK11-15. These may be recorded incorrectly if they were changed during the sequence step immediately preceding 'Snapshot'.

Word 2 (bytes 36-39) contains the contents of AR10.

Word 3 (bytes 40-43) contains the contents of AR11 (8 bits) and AR9 (last 24 bits).

Word 4 (bytes 44-47) contains the OP Code (8 bits), C1, C2, L1, L2, L3 (all 4 bits), 2 unused bits, AR9 (first 2 bits).

Word 5 (bytes 48-51) contains the Program Counter (32 bits).

Word 6 (bytes 52-55) contains Scratchpad Address (7 bits), one unused bit, Shift Counter (8 bits), Interlocks (16 bits).

Word 7 (bytes 56-59) contains CR3.

Word 8 (bytes 60-63) contains CR4.

D.2.9.2 The Multichannel Control Unit

D.2.9.2.1 General

The execution of the 'Diagnose' instruction followed by the execution of a 'Start Device', 'Halt Device' or 'Test Device' instruction, causes certain diagnostic operations to be performed by the Multichannel Control Unit which controls the specified channel. The channel control unit functions in the normal manner when the 'Diagnose' instruction execution is followed by that of either a 'Check Channel' instruction or a channel interrupt. A channel that is not a multichannel which is addressed by the diagnostic operation will ignore the 'Diagnose' instruction and respond to the input-output instruction in the normal way.

The count specified by the 'Diagnose' instruction has the following effect:-

(a) count < 4 Central Control Unit executes 'Diagnostic Snapshot' operation. *

(b) count = 4 or 5 as for (a) but followed by Multichannel Control Unit action. *

(c) count > 5 immediate Multichannel Control Unit action.

D.2.9.2.1 (continued)

The input-output initiation procedure is modified as follows :-

(a) The instruction specified address must address a channel controlled by a Multichannel Control Unit. The device address is used as an operation modification indicator.

(b) The format of the Channel Address Word is reorganised as follows :-

bits	0-3	Protection Key
	4-7	Least significant 4 bits of appropriate command code
	8-31	Address of first byte of a selected main store location.

The word is followed by a test and control word which occupies positions 76 to 79.

D.2.9.2.2 The Start Device' Instruction

The execution in the previously defined circumstances of a 'Diagnose' instruction followed by that of a 'Start Device' instruction causes the following actions to occur.

(a) The channel control unit staticises the control information and fetches the test word.

(b) It then reads into a hardware register the word in its Scratchpad selected by the combination of channel number and the least significant four bits of the device number. This register is overwritten by the test word if the channel command is a 'Write', 'Write Control' or 'Erase'.

(c) Finally the word in the register is placed in the Scratchpad by means of a sequence of manipulations controlled by the setting of the most significant four bits of the instruction specified device number. The same word is also stored in main store at the address specified by the Channel Address Word, in this case parity errors may cause some of the bytes to be set to all '1's.

The above operation causes the Condition Code to be set to 1 and the second Channel Status Word to be stored with the byte count replaced by the least significant two bytes of the stored word.

The normal operation of both the non-addressed channels and in certain circumstances of the addressed channel, are not effected by this diagnostic operation.

D.2.9.2.3 The 'Test Device' Instruction

The execution of a 'Test Device' instruction in the circumstances previously described for the 'Start Device' instruction, causes

D.2.9.2.3 (continued)

the channel control unit to staticise the control information and fetch the test word. In its subsequent operation, the control unit transfers the test word through its register and then stores it in the main store at the address specified by the Channel Address Word. The Condition Code is set to '1' and the Channel Status Word is set as for the 'Start Device' instruction.

The operation of both the non-addressed channel, and, if the most significant four bytes of the device number are zero, the addressed channel, are not effected by the operation.

D.2.9.2.4 The 'Halt Device' Instruction

The execution of a 'Halt Device' instruction, following that of a 'Diagnose' instruction, causes the Multichannel Control Unit controlling the addressed channel to perform a general reset operation on all its channels. As a result the Condition Code is set to 2 and the addressed channel will not appear to be either busy or operating. The non-addressed channels will be reset as if they were still busy or operating, if this was previously the case. There are no subsequent interrupts to reset these conditions.

D.2.9.2.5 Further Information

Further information concerning the diagnostic operation of a Multichannel Control Unit may be found in the 'Engineering Description' of the 4-70.

D.2.10 INSTRUCTION NAME : Set Storage Key

OPERATION CODE : 08

FORMAT : RR

DESCRIPTION :

Execution of the 'Set Storage Key' instruction causes a new value to be loaded either into a single main store Reservation Key or, if both the Reservation Key Control Indicator (see note below) and the engineer's switch which controls paging, are set, into a group of eight reservation keys.

The address of the 512 (4,096) byte reservation area associated with the key(s) is held in the second register specified by the instruction. The value loaded is that held in the first register specified by the instruction.

CONDITION CODE : Unaltered.

NOTES :

(1) The second register specified by the instruction contains the address of the 512 (4,096) byte main store area whose key(s) is (are) to be set. The format of the register must be as follows :-

0 | 0 000

ignored	area address	ignored	0000
0	8	23	28 31

Programmers are recommended to ensure that bits 28-31 are zeroes. If bit 29 = 1 only the keys on the odd half of store are affected. When a group of 8 keys are involved, the main store area does not necessarily have to start at a multiple of 4096. When setting the obligatory zero key on the first block of store, an address less than 96 must be specified, otherwise 'Address Error' interrupt occurs.

(2) The first register specified by the instruction contains the six bit reservation key in bits 24-29. Bits 0-23 and 30-31 are ignored. The function of the storage key fields are specified in Section 3.4.

(3) If a location outside the available main store is specified the operation is terminated with unpredictable results.

(4) The Reservation Key Control Indicator is loaded during a change of state with the contents of bit 23 of the new Interrupt Status Register.

Issue	Date	Auth'y	Log	Title	Document No.
4	16.2.68	MRW	743	CENTRAL PROCESSOR MODELS 4-70/75	PS 4.10.70
4/1	10.7.68	MRW	809		Page D.14

D.2.11 INSTRUCTION NAME : Insert Storage Key

OPERATION CODE : 09

FORMAT : RR

DESCRIPTION :

Execution of the 'Insert Storage Key' instruction causes the specified main store Reservation Key(s) to be loaded ('OR'ed together and then loaded) into the first register specified by the instruction. The first action applies unless both the Reservation Key Control Indicator and the engineer's key which controls paging, are set. The address of the 512 (4,096) byte reservation area with which the key is associated is held in the second register specified by the instruction.

CONDITION CODE : Unaltered.

NOTES :

(1) All notes to Set Storage Key are applicable to Insert Storage Key.

(2) Reservation Keys are not changed by the execution of this instruction.

(3) Bits 0-23 of the first register are unchanged.

(4) When multiple keys are stored, they are 'OR'd without a check being made to ensure that the keys are identical.

D.2.12 INSTRUCTION NAME : Write Direct

OPERATION CODE : 84

FORMAT : SI

DESCRIPTION :

Execution of the 'Write Direct' instruction causes an eight-bit byte to be staticised on all Direct Control Trunks. The byte remains staticised either until the execution of another 'Write Direct' instruction or until a 'Power Failure' interrupt causes it to be set zero, whichever occurs first.

At the same time the Signal Out lines of one or more trunks, as selected by bits in the Immediate Field, are pulsed.

CONDITION CODE : Unchanged.

NOTES :

(1) Trunks are selected by the bits of the Immediate Field as follows :-

I. Field	Trunk pulsed.
0	Six
1	Five
2	Four
3	Three
4	Two
5	One
6	Initial Program Load 1
7	Initial Program Load 2

A trunk is only selected if its associated bit is set to '1'.

(2) Selection of a non-existent trunk has no effect.

(3) If the Direct Control Option is not installed an 'Op Code Trap' interrupt will be requested.

(4) It is a program restriction that bits 0 - 5 must not be set if either or both bits 6 and 7 are set.

(5) Further information is given in PS 4.6.10. *

D.2.13 INSTRUCTION NAME : Read Direct

OPERATION CODE : 85

FORMAT : SI

DESCRIPTION :

Execution of the 'Read Direct' instruction causes the sampling of the Static In lines of the Direct Control trunk selected by the instruction's Immediate Field. If the sampling is successful, the eight-bit byte sampled is stored in the location specified by the instruction specified address.

CONDITION CODE : Unchanged.

NOTES :

(1) Selection of trunks is in accordance with Note (1) for 'Write Direct'.

(2) The remote Initial Program Load bits are ignored.

(3) If more than one trunk is selected the results are unpredictable. If a non-existent trunk is selected a zero byte is stored.

(4) If bits 0-5 of the I_2 field are zeros, no trunks are sampled; instead a set of engineers' breakpoint switches are sampled.

(5) If the Direct Control Option is not installed, an 'Op Code Trap' interrupt will be requested unless all 8 bits of the I_2 field are zeros, in which case the action described in Note (4) takes place.

(6) Further information is given in PS.4.6.10. *