THE CR8/16 MICROCOMPUTER
USER REFERENCE MANUAL

CHRISTIAN ROVSING A/S
Cophenhagen Denmark

[illegible]

THE SCOPE OF THIS DOCUMENT

This User Guide is intended to answer most of the questions that a potential user of the CR8/16 microcomputer might ask. It aims at breadth rather than depth of coverage - in other words it is considered more important to provide the user with a "Bible" to give at least some guidance about any CR8 topic, rather than to supply every detail on some topics and omit others altogether. In particular, the sections on the CP/M and MP/M operating systems are an overview rather than a comprehensive guide since these operating systems are very widely used in the computing industry and many books have already been written about them.

Incidentally, the same thing applies to the Z80 and 8088 microprocessors which play host to those operating systems.

It is intended that the amount of technical knowledge required to understand this manual will gradually increase as one reads through it. For instance someone with a bare minimum of computer knowledge should be able to understand section 1, whereas section 5 is aimed at people who want to interface their own hardware or system software to the CR8.

The questions that this document attempts to answer are outlined below. Note that the numbering of the questions corresponds to the numbering of the sections that supply the answers. (See contents.)

1. What is it ?
2. Why should I choose a CR8 ?
3. How do I start it up & make it go ?
4. How do I run my application programs and manage my data and programs ?
5. How does it work ?

Please note that the name "CR8" mentioned throughout this document is an abbreviation for "CR8/16 Microcomputer".

Trademarks

The following names referred to in this document are the trademarks of the companies mentioned below:

CP/M	Digital Research
MP/M	Digital Research
Z80A	Zilog
i8088	Intel
Multibus	Intel
Multi Module	Intel
dBASE II	Ashton-Tate
MDBS III	M.D.B.S.
CBASIC, CBASIC-86	Digital Research
CB, CB-86	Digital Research
PASCAL MT+, SPP	Digital Research
RM/COBOL	Ryan McFarland Corporation
COGEN	Bytek

Document Reference: : CSD-MIC/001/USM/0010
1st Edition : April, 1983
Published by : CHRISTIAN ROVSING A/S
2, Lautrupvang
DK-2750 Ballerup

[illegible]

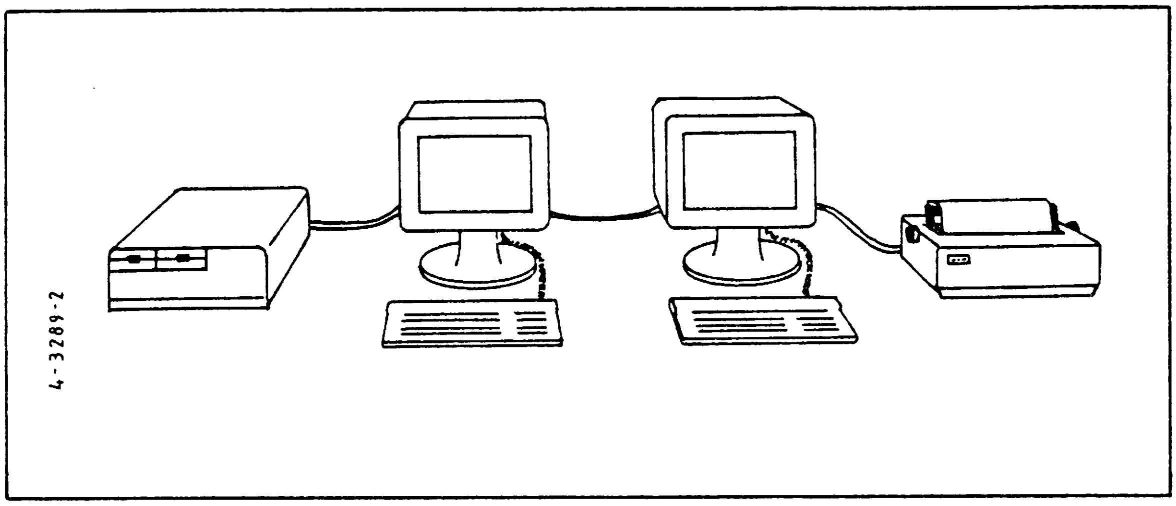
[illegible]

[illegible]

1. INTRODUCTION TO THE CR8

1.1 BRIEF DESCRIPTION OF THE CR8

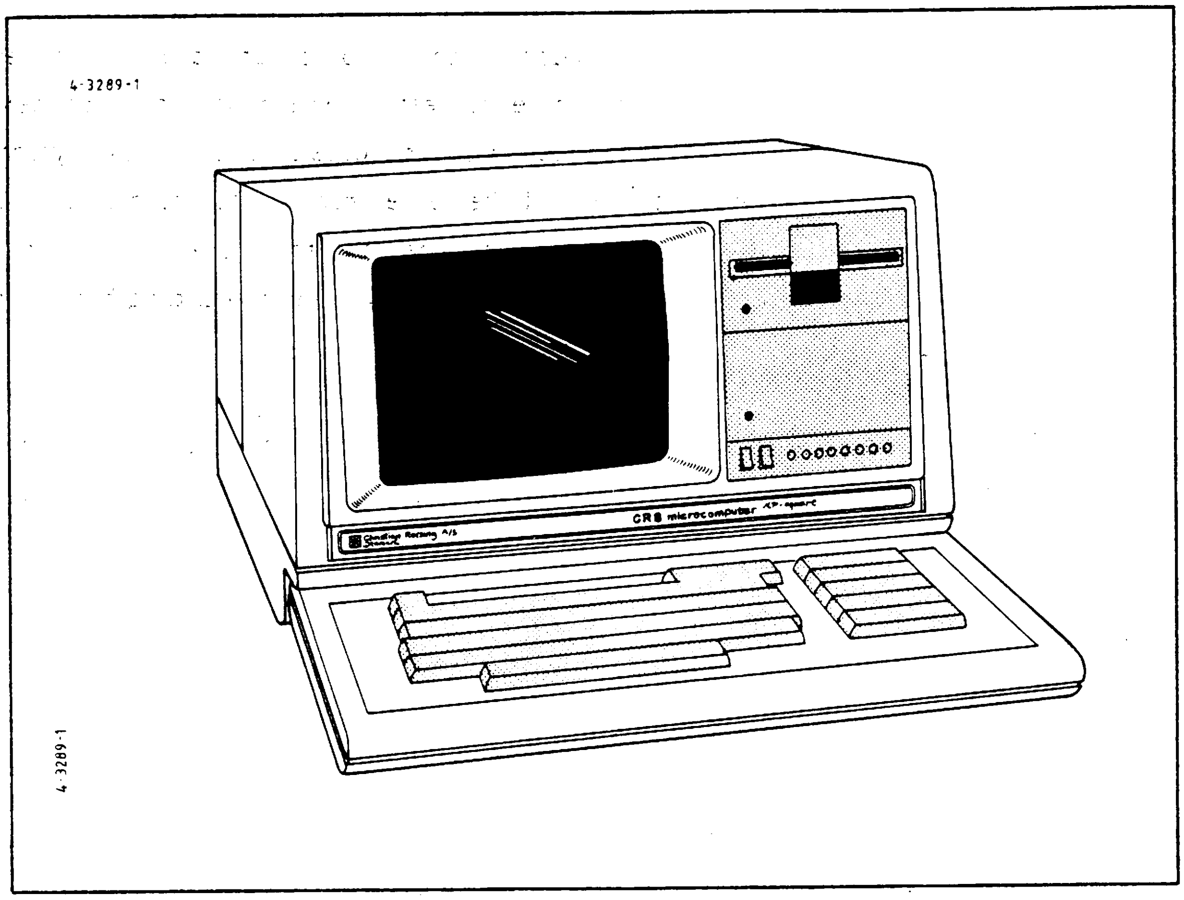
The CR8 is a complete general-purpose microcomputer system which is available in a variety of hardware and software configurations. It comes in two basic forms: the cabinet model and the integrated workstation. The integrated work-station looks like a normal VDU but it also contains the main processor(s) and disc drives - in fact a complete stand-alone computer system. The cabinet model contains much the same components, but the terminal(s) is (are) separate.



CR8 CABINET MODEL (A TYPICAL CONFIGURATION)

At the heart of every CR8 is the MP^2 (Multi Purpose Multi Processor) board. This is the part that controls the whole CR8 computer system and does the 'thinking'; it is here that programs 'run'. This single printed circuit board houses a Z80A single chip microprocessor plus the option of using an 8088 microprocessor as well. Up to 128K bytes of on-board RAM is accessible; a floppy disk controller and two serial interfaces (for connecting terminals or printers) are also included.

Every CR8 also includes 2 disk drives (apart from some special-purpose models). A disk drive is used for bulk storage of information - very often in the form of (an electronic representation of) text which can be displayed on a terminal or printer. Programs that define and control the duties to be performed are also stored on disk.



CR8 INTEGRATED WORKSTATION

There are two types of disk drive available - floppy and hard. A floppy disk can hold less information than the hard (otherwise known as Winchester) disk, but has the advantage that the disks may be removed. Thus the upward limit of available storage depends only upon the available cupboard space for floppy disks.

The third vital component of the CR8 is one or more terminals. A terminal has two parts - a keyboard with which the user communicates with the processor board and a VDU screen or printer with which the processor communicates with the user.

There are also various optional circuit boards which are needed for certain applications. These include the SCI board which allows the connection of four more terminals to the CR8; the XMA board which allows connection to any number of other terminals or computers via an X-Net network; and the PAM bus driver board which turns the CR8 into a PAM process control system. The next section describes the various options in more detail.

All this equipment would be useless without some **software.** If the reader will excuse a small digression into some homespun philosophy, a computer without software is like a man without life. In other words a dead man is physically much the same as a live one, he consists of the same organs assembled in the same way. However, he cannot do anything at all, he is totally useless. So it is with a computer without software. Software is the invisible and intangible driving force which causes the computer to work.

The CR8 (in some configurations) is in effect two computers since it has two processors ("brains"). Thus it has two bundles of software to co-ordinate and control its activities. These are known as Operating Systems. One of these would be CP/M or MP/M, and the other is called SCRIMP/M. Various varieties of these operating systems are used on the CR8, depending on the exact configuration - more details appear later. CP/M is for single user systems and MP/M is for multi-user systems. Both are very popular and commonly found within the microcomputer industry, which means that a large variety of application programs are available for the CR8. Application programs or packages are those pieces of software which actually do something useful like word processing; operating systems are programs that coordinate and control the useful programs. The range of uses to which the CR8 can be put, of course, is limited only by the programmer's imagination. However, ready-made programs are already available for, for example, word processing (including spelling correction), report generation, data base management, sorting & merging, project planning, and many more - for more details, see sections 1.3 and 2.2.3.

1.2 CONFIGURATION OPTIONS - HARDWARE

The two basic types of CR8 - cabinet and workstation models - each come in a variety of forms. Furthermore, each of the models can be configured using the options described in section 1.2.1.

Standard Features of the CR8 Cabinet Models

Model No.	Description of Features Included
CR8/007/1	4-Unit cabinet (size: width 48 x height 17.7 x depth 55 cms) containing: o MP^2 board (8088 processor, Z80A Processor, 128K bytes RAM, Floppy Disk Interface, 2 V24 interfaces, Multibus interface). o 5 Multibus slots o Power Supply (150 watts) o Fans o Disk drives - one of the options 11, 21 or 31. o Facility to connect 1 terminal and 1 printer (options 40 to 49). Single user system.
CR8/007/4	Four user version of the above. In addition it houses: o 128K bytes extra RAM (option 93) o 1 SCI board (option 97) o Facility to connect up to 4 terminals and 2 printers in all (options 40 to 49)

Model No.	Description of Features Included
CR8/007/8	Eight user version of CR8/007/1. In addition it houses: o 256K bytes extra RAM (option 94) o 2 SCI boards (option 97) o Facility to connect up to 8 terminals and two printers in all (options 40 to 49)

Standard Features of the CR8 Integrated Workstation Models

Model No.	Description of Features Included
CR8/017/1	Integrated Workstation (microcomputer and disks built into a VDU with a 12" screen and detachable keyboard). The unit contains: o MP^2 board (see description of Cabinet model) o 8 Multibus slots (6 in older examples) o Power Supply (165 Watts) o Fans o Disk drives - one of the options 11, 21 or 31 o Facility to connect one printer (Options 40 to 46)
CR8/017/4	Four user version of the above. In addition it houses: o 128K bytes of extra RAM (option 93) o One SCI board (option 97) o Facility to connect up to 3 terminals and two printers (options 40 to 49)

1.2.1 Optional Extras

A list of CR8 optional extras is given below, followed by short descriptions of each one. This list was correct on 1st March, 83; new options are added from time to time - for instance, larger capacity disks are shortly to be announced.

Option Number	Description
01	Single floppy drive, 5-1/4", 1 Mbyte (dual side, dual density)
11	Dual floppy drive, 5-1/4", 2 x 1 Mbyte (dual side, dual density)
21	6.38 Mbyte Winchester disk + 1 floppy drive (opt-01) + Winchester disk controller board.
31	12.76 Mbyte Winchester disk + 1 Floppy drive (Opt-01) + Winchester disk controller board
40	Matrix printer, 80 cps
41A	Matrix printer, 120 cps
41B	Matrix printer, 200 cps
42A	Daisy wheel printer, 40 cps (letter quality)
46A	Tractor feeder for Opt-42A
42B	Daisy wheel printer 55 cps (letter quality)
42C	Daisy wheel printer 35 cps (letter quality)
44B,C	Double sheet feeder for Opt-42B,C
45B,C	Single sheet feeder for Opt-42B, C
46B,C	Tractor feeder for Opt-42B, C

Option Number	Description	
48B	CR5 Compass, X3.64 subset terminal	English ch.
		Danish ch.
48C	CR7 Comfort, X3.64 subset terminal	English ch.
		Danish ch.
48D	CR7 Comfort-X, direct X-Net attachable X3.64 subset terminal	English ch.
		Danish ch.
49	ANSI X3.64 Compatible Terminal (Tandberg TDV)	English ch.
		Danish ch.
93	128K byte memory board	
94	256K byte memory board	
97	SCI-board, 4K universal serial comm. I/F (V24/RS232, X21, current loop) incl. on board Z80A and 32K RAM.	
o	XMA, X-Net interface for MP-square	
o	MP-square incl. 8088, Z80, 2x64K RAM, floppy disk interface 2xV24 and Multibus interface	
o	CR8 Cabinet, 2U, incl. 5 Multibus slots, Power Supply (150W, 5V 20A, +12V 6A, -12V 1.5A) and fan units. (W48cms (19"), H8.85 cms (2U) L55cms)	
o	CR8 Cabinet, 4U, incl. 5 Multibus slots, Power Supply 150W, 5V 20A, +12V 6A, -12V 1.5A) and fan units (W48 (19"), H17.7 (4U), L55cms.)	

- CR8 Cabinet, 6U, incl. 10 Multibus slots, one power supply (150W, 5V 20A, +12V 6A, -12V 1.5A) and fan units (W48 (19"), H26.5 (6U), L55)

- CR8 Cabinet, 6U as above, but with two power supplies.

- PAM Interface

- Mathematics Co-processor for MP^2

Memory Extra RAM can be added - 128, 256 or 512 Kbytes. This does not include memory on MP^2 or SCI boards.

SCI Board This serial communications interface board can be set up to provide various interface types, (V24, X21, current loop) for communication with other devices such as terminals, printers or mainframes. Each board supports 4 interfaces which do not have to be of the same type. The SCI board houses its own Z80A microprocessor and 32K bytes of RAM.

Mathematics Co-processor In dual processor systems, the 8088 CPU may be replaced by a small board which houses both the 8088 plus an 8087 mathematics co-processor. It is used to speed up processing for applications requiring a lot of floating-point operations. It extends the normal instruction set and range of data types.

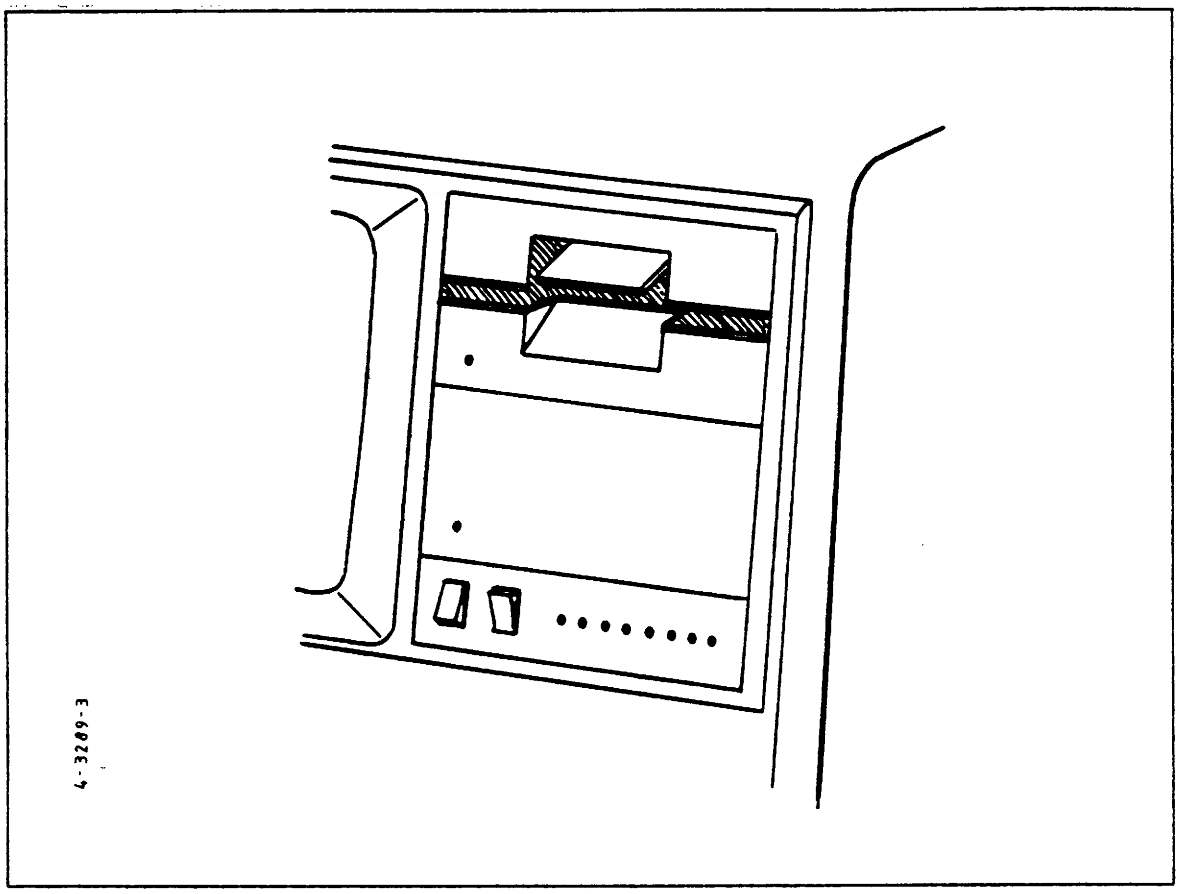
X-Net Interface Board X-Net is Chr. Rovsing A/S's local area network which is used for connecting various computing resources within, say, a factory or university campus. X-Net will connect to any manufacturer's equipment; however, this is simplified in the case of the XMA X-Net interface board, which precludes the use of the separate terminal adapter required by other manufacturer's terminals. Furthermore, the CR8 may take on a special role within X-Net as the X-Net Administrator (XNA). Please refer to X-Net Local Area Network System Description (ref. X-NET/RFM/0001) for further details. The X-Net interface board occupies one Multimodule slot in the MP^2 board.

PAM Interface. The Process Addressable Monitor is Chr. Rovsing's process monitoring & control system, which is used in many applications in for example, production supervision, energy management, and building security. It uses a single twisted pair cable for both data transmission & power distribution to the monitoring/control points, at distances of up to 4 kilometres. The CR8 Interface board handles up to 160 PAM interface modules, each of which in turn carries signals for up to 8 sensors or output devices. The interface board occupies one of the 3 available Multimodule slots on the MP^2 card.

Other Multimodule boards. The user may of course plug any Multimodule-compatible boards into the Multimodule slots. Many such boards are commercially available, for example serial or parallel programmable peripheral interfaces; alternatively the user may build his own.

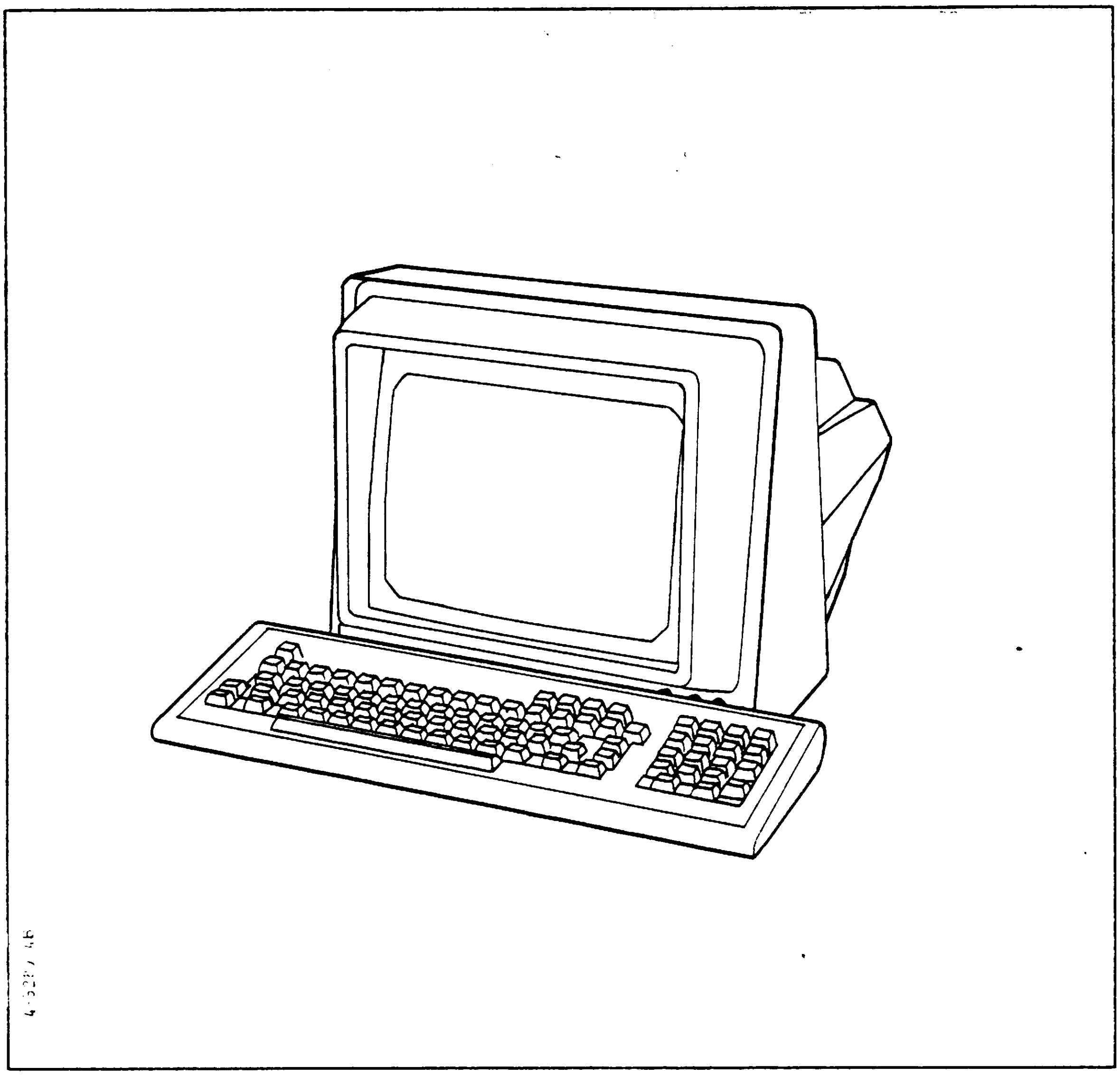
Disk Drives

All CR8 models (apart from the 2 Unit Cabinet Model) have space for two disk drives of standard size ((5.88 x 3.38 x 8.00 inches) or four in cases where the newer type of disk is installed which is only 1.69 inches high. The capacities quoted in the list of options are all unformatted capacities, i.e. a small proportion of these capacities are used by the system for 'housekeeping' and is not available to the user. Note that the Winchester disk requires a controller board whereas up to 2 floppy drives are catered for as standard within the MP^2 board.



CLOSE-UP OF CR8 WORKSTATION SHOWING FLOPPY DISK ABOVE AND WINCHESTER BELOW

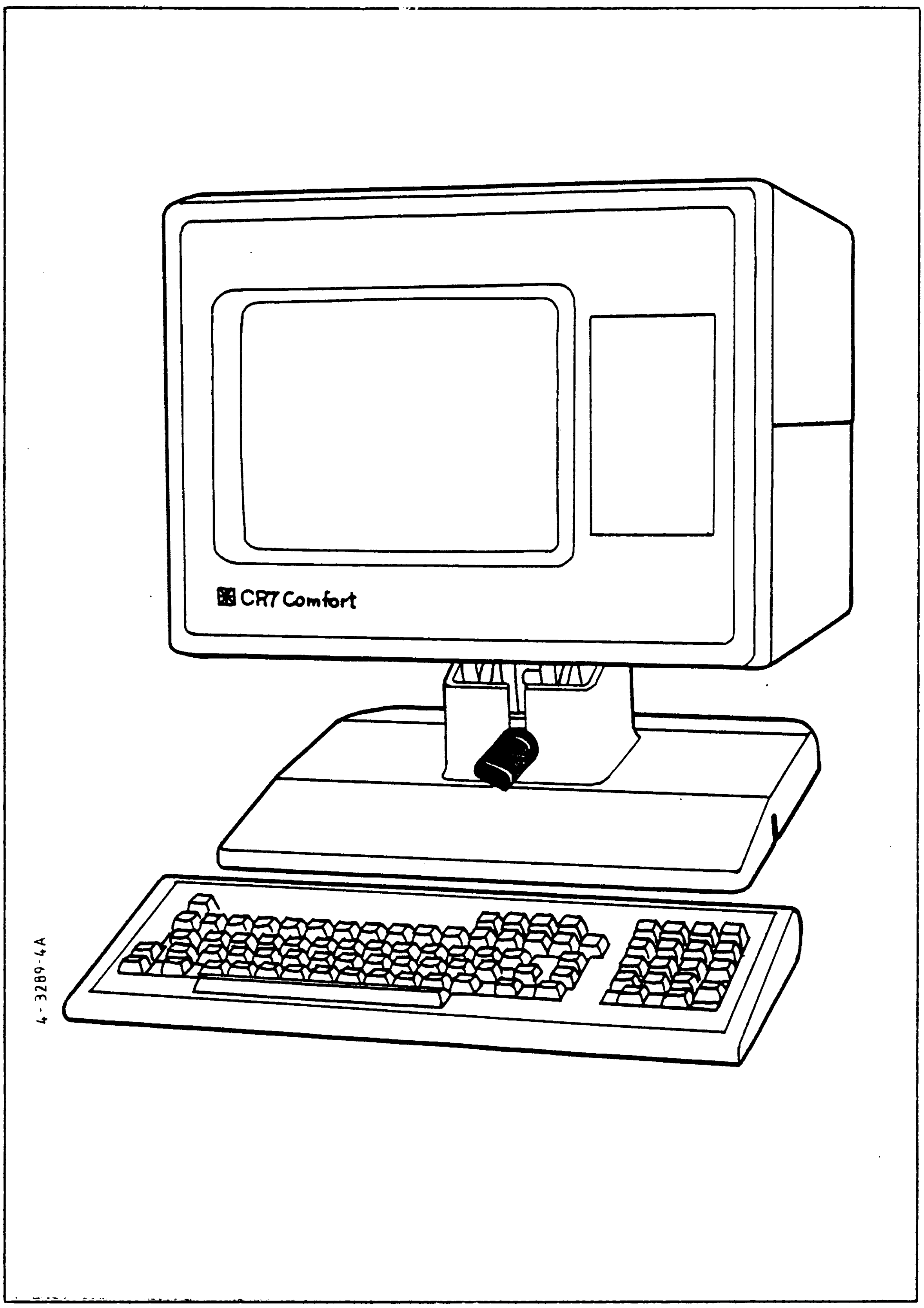
VDU's There are several VDU types that may be supplied with the CR8. One of these is included as the basis of the integrated workstation - in other words the VDU contains the disc drives, processors, etc. Any of the types supplied by CR may be used in multi-user CR8 systems. All are asynchronous terminals supporting ANSI X3.64. All have detachable keyboards and cursor addressing.



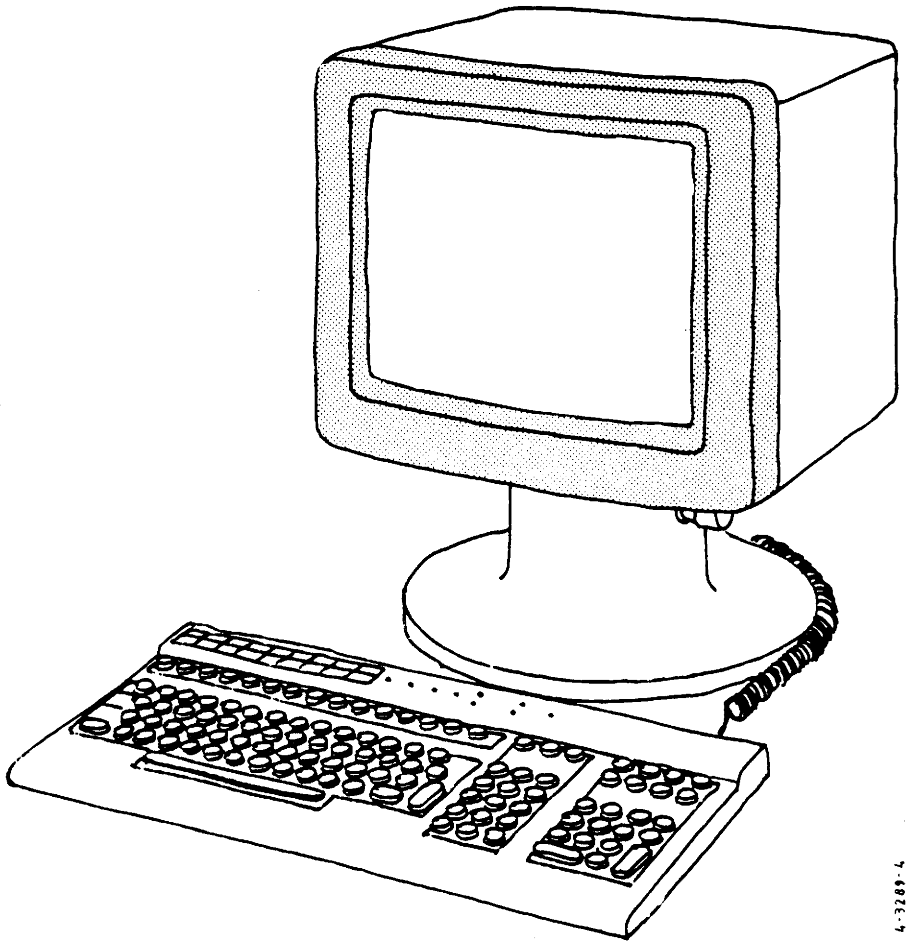
THE CR5 COMPASS TERMINAL

The three terminal types that do not contain disc drives etc. are the CR5 Compass, the CR7 Comfort, and the Tandberg TDV.

Note that any terminal to be used with the CR8 should support the ANSI X3.64 standard.



THE CR7 COMFORT TERMINAL



THE TANDBERG TDV TERMINAL

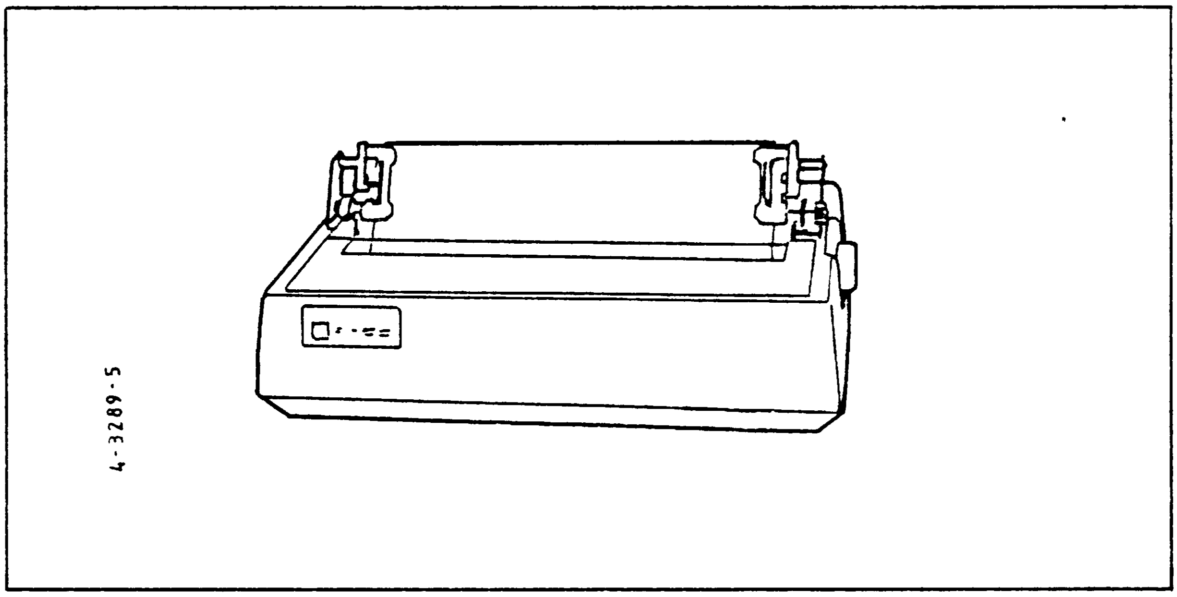
Printers There are 2 main types of printer that may be supplied with the CR8 - dot matrix and daisy wheel.

The dot matrix printers used continuous stationery which is fed through by a tractor feed mechanism (engaging with the holes in the edges of the paper). These printers are intended for higher speed of output though the print quality is not as good as that of the daisy wheel printer. The dot-matrix printers come in three versions (which outwardly appear very similar):

Option 40:
Speed: 80 characters per second. 80 columns of characters across the paper.

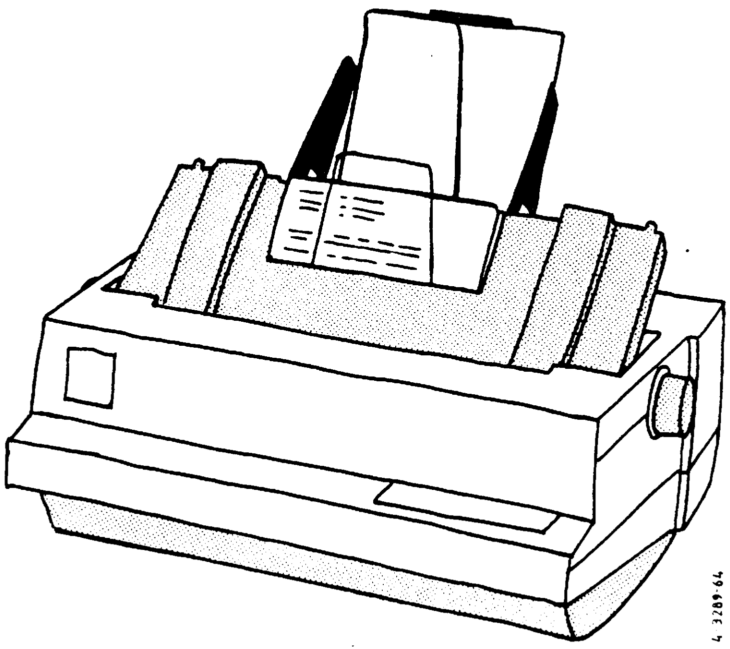
Option 41A:
Speed: 120 characters per second. 132 columns of characters across the paper.

Option 41B:
Speed: 200 characters per second. 132 columns of characters across the paper. This model has the option of running at half speed but with a higher print quality.

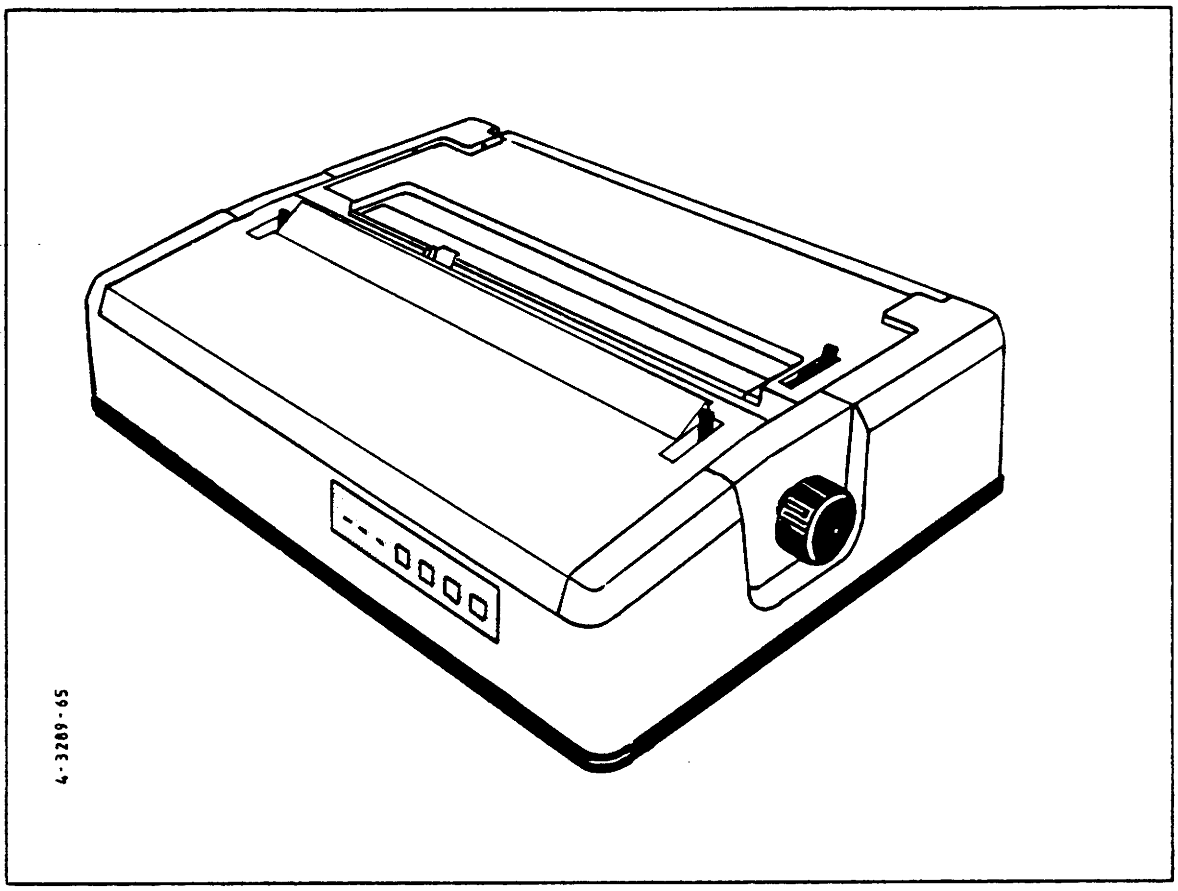


DOT-MATRIX PRINTER

The other type of printer that may be supplied with the CR8 is a **daisy wheel** printer. This is in effect an electric typewriter without a keyboard, so the print is of letter quality. It is intended for word processing applications. There is a choice of three adaptors - a tractor feed for continuous forms, a single sheet feeder, or a double sheet feeder. When the double sheet feeder is fitted, the user may select one of two paper styles when printing different parts of his document. This feature is particularly useful when the first page of a letter requires pre-printed headed paper whereas the continuation sheets use plain paper. It takes only few seconds to swap one adaptor for another.



DAISY-WHEEL PRINTER WITH SINGLE SHEET FEEDER (OPTION 42B, C)



DAISY-WHEEL PRINTER (OPT. 42A)

1.3 CONFIGURATION OPTIONS - SOFTWARE

The CR8 microcomputer supports a range of CP/M-based operating systems, as follows:-

	Systems using Z80 Only	Systems using both Z80 and 8088	
		Z80	8088
Single user	CP/M 2.2	SCRIMP/M	CP/M-86
Multi User	MP/M II (supports 4 users)	SCRIMP/M	MP/M-86 (supports 8 users)

CP/M 2.2 and **CP/M-86** are single user systems, suitable for dedicated use in small offices or industrial control applications. As "single user" systems, there is only one system console, and only one application program may be executing at any time.

MP/M II and **MP/M-86** are multi-user, multi-programming systems. That is, they support a number of system consoles (interactive terminals) - up to 4 for MP/M II, 8 for MP/M-86, each console having full control of the system facilities. It is possible to run several programs simultaneously from one (in fact, any) terminal.

SCRIMP/M - Subset Comparable Real-time Implementation MP/M - is designed to handle the input/output functions both on the MP^2 processor board and SCI serial interface board (Z80A processors in both cases).

The following gives a glimpse of the **applications packages** for the CR8 which are available from Chr. Rovsing; more details of them are given in section 2. A full list of the available software is given at the end of this section; it does not include specialist software such as X-Net or protocol conversion.

Word-processing system for text entry and amendment. This optionally includes a **mailing program** to simplify the production of many letters that have sections in common, and a **spelling checker.**

Financial Modelling This is a sophisticated screen-oriented calculating package which may be programmed so that numeric fields in the display may be defined as mathematical functions of other fields. Thus the effects of changing a parameter are instantly displayed at all stages through the calculation. The package has many applications including: balance sheets, tax estimation, market share analysis, or statistics calculation in scientific experimentation.

Planning Tool This package is an aid to planning and handles report generation. It can also do calculations as part of its report. Its function is similar to the Financial Modelling package described above. The report is programmed in a very high-level English-like language.

File Sorting CR can provide a program for sorting files. It allows a wide variety of sorting keys and input and output formats, and includes facilities for merging up to 32 files.

A **Project Management** tool is available to aid in the management of small projects, using the technique of critical path analysis. The tradeoffs between time, money and manpower may be investigated : which activities may be delayed without delaying the whole project and which may not.

Data Management Packages:

A **data entry and retrieval system** is available to facilitate the storage of information on user defined forms. Records may be selectively retrieved using a mask of the record to be matched; for example if the data were customer names and addresses, all customers called Smith or all customers in a certain town could be selected.

dBASE II is a simple data base management system that allows for a flexible user designed input form. All functions are specified in a command language that can be used interactively or combined into programs to be run as a job (for more complex report generation for example).

MDBS III is a comprehensive data base management system for the storage and retrieval of information on a large scale. Not only the data but the relationships between them are stored. Data can be retrieved using the English-like MDBS-QRS query language or via the user's own programs which can be in a variety of programming languages.

Programming Languages

The CR8 will support the following programming language options: Basic (CBASIC interpreter or compiler), Pascal, Cobol, and Fortran. Z80 assembler and 8088 assembler come as standard with the CR8.

Communications

The CR8 may be provided with software to solve many communications problems, including emulators for IBM 2780 and 3274, Burroughs TC500, ICL CO3, NCR 801, X25 protocol and others.

Others

This brief overview of the available software is not complete, and will quickly become out of date since CR frequently release new software packages and options. Please contact the Business Group, CR Hotline for a current list. The address is at the front of this manual.

SOME OF THE SOFTWARE AVAILABLE FOR THE CR8

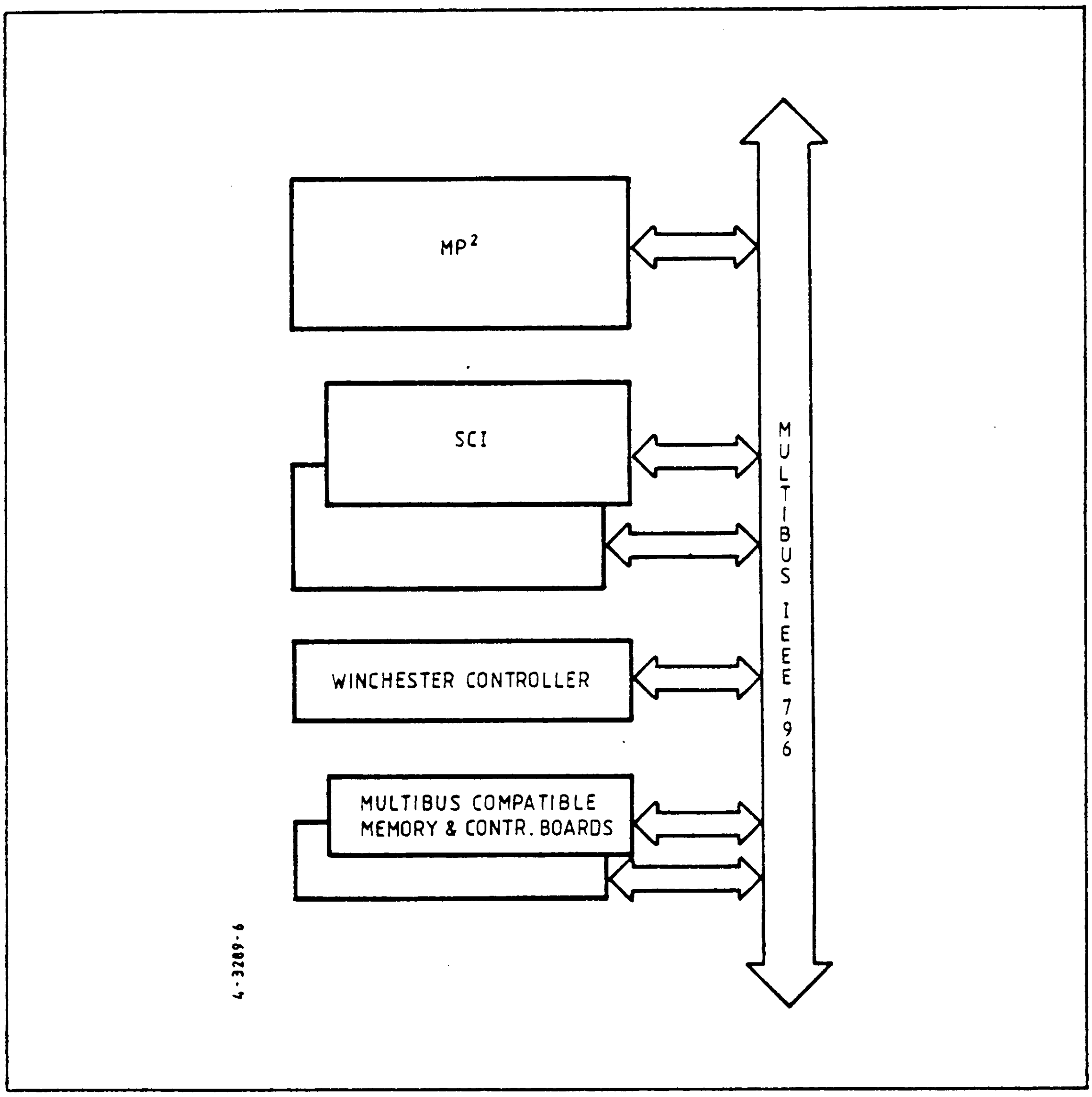
Please consult CR for the latest list.

Option No.	Name	Z80A single user	Z80A multi user	8088 single user	8088 multi user
	Operating Systems				
200	CP/M 86			X	
201	MP/M 86				X
202	CP/M 2.2	X			
203	MP/M II		X		
	SCRIMP/M	X	X		
	Languages				
60	FORTRAN	X	X		
61A	C-BASIC	X	X	X	X
61B	BASIC compiler	X	X	X	X
62	PASCAL MT+ (incl. SPP)	X	X	X	X
63	COBOL	X	X	X	X
	Applications				
64A	Word Procesing, English	X	X		
64B	Word Processing, Danish	X	X		
65	Proof Reading (English)	X			
	Calculation & Financial Planning	X	X	X	X
67	Data Input	X	X		
69	Sorting	X			
70	Project Management	X	X	X	X
71A	Mailing, English	X	X		
71B	Mailing, Danish	X	X		
72	Indexing				
73	MDBS III database	X	X		
74	dBASE II	X	X		
75	Finanssystem med budget-modul, Danish	X	X	X	X
76	Debitorsystem, Danish	X	X	X	X
77	Lagersystem, Danish	X	X	X	X
78	Kreditorsystem med Finansintegration, Danish	X	X	X	X
79	Kundeordresystem, (forudsætter Debitor/ lager) Danish	X	X	X	X
80	Totalsystem incl. opt.75-79	X	X	X	X
81	Planning & Report Generation, English	X			
82A	Textproc. English	X	X	X	X
82B	Textproc. Danish	X	X	X	X

2. FEATURES OF THE CR8

2.1 Hardware

The CR8 was designed to be not merely a stand-alone desk-top computer, nor merely a multi-user microcomputer system but as the starting point for building communications systems, local networks, process control systems, and many other applications.



CR8 FUNCTIONAL DIAGRAM

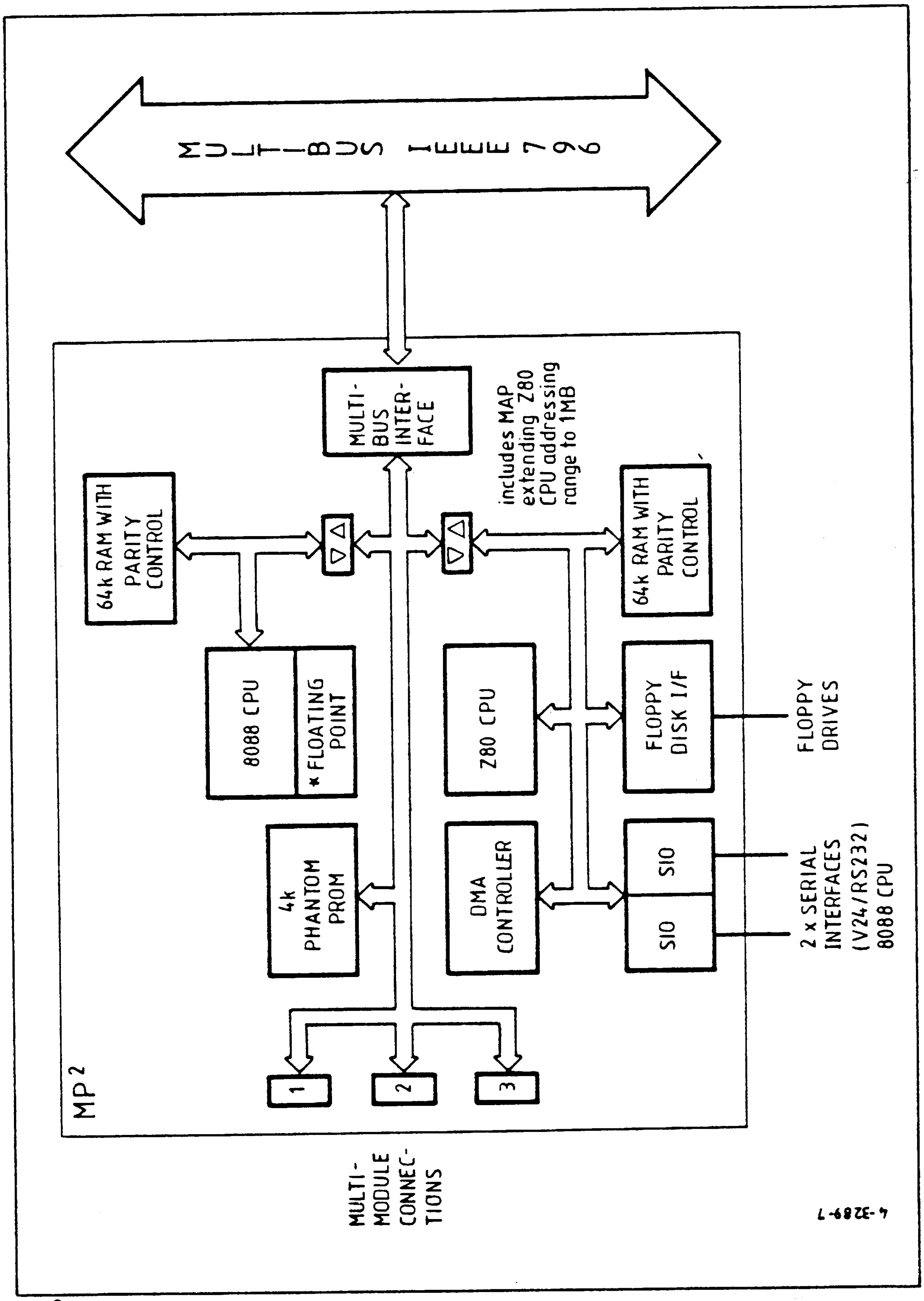
This expandability is in turn a function of various CR8 design features - in particular its MP^2 dual processor board, its System Communications Interface board, its Multimodule and Multibus compatibility, its 1-megabyte addressing range, and its CP/M, MP/M and SCRIMP/M operating systems.

2.1.1 MP^2 Processor Board

This is the heart of the CR8 microcomputer. It is a highly integrated dual processor microcomputer system on one board. This board is mechanically and electrically compatible with the INTEL Multibus Interface (IEEE796 standard). This means that the user may combine any of hundreds of commercially available boards with the MP^2 in his CR8. Commercially available boards include parallel I/O, analog interfaces, memory, processors, hard disc interfaces, and so on. In addition the board has connectors for 3 standard INTEL iSBX Multimodule cards. Here again, many applications are catered for in the market place, or the user may design and make his own interface cards to that standard for his own special peripherals. Furthermore, you could use these Multimodule slots for CR's interface cards to the X-NET local area network or PAM monitor and control systems.

The two processors in the CR8 are the Zilog Z80A 8-bit cpu and the INTEL 8088 16 bit CPU. In simpler CR8 models using one processor, the Z80A does everything; in dual CPU versions, the user's applications generally run in the 8088 while the I/O overhead is off-loaded onto the Z80. This greatly improves the performance compared with single processor microcomputers, especially in the case of multi-user systems. Furthermore, a mathematics co-processor (INTEL 8087) may be added for further increase in performance for "number crunching" applications. (See section 5.3.3.6 for further details of this mathematics co-processor).

The MP^2 board houses 128 Kbyte of RAM, half of this is local memory for each processor though most is globally accessible by both processors. Note also that extra memory may be added on separate Multibus boards up to an overall maximum of 640 Kbytes. The addressing range of the Z80A, normally 64 Kbytes, has been extended to a megabyte in the MP^2 card by means of mapping registers. Thus both processors can address the full megabyte. The PROM area on the MP^2, normally 4 or 8 Kbyte, may optionally be increased up to 64 Kbyte.


MULTIBUS IEEE 796
MULTI-BUS INTER-FACE
includes MAP extending Z80 CPU addressing range to 1MB
64k RAM WITH PARITY CONTROL
64k RAM WITH PARITY CONTROL
8088 CPU
* FLOATING POINT
Z80 CPU
FLOPPY DISK I/F
FLOPPY DRIVES
4k PHANTOM PROM
DMA CONTROLLER
SIO
SIO
2 x SERIAL INTERFACES (V24/RS232) 8088 CPU
1
2
3
MP 2
MULTI-MODULE CONNEC-TIONS
4-3289-7

MP^2 FUNCTIONAL DIAGRAM

The MP^2 board also contains a built-in interface for two floppy disk drives.

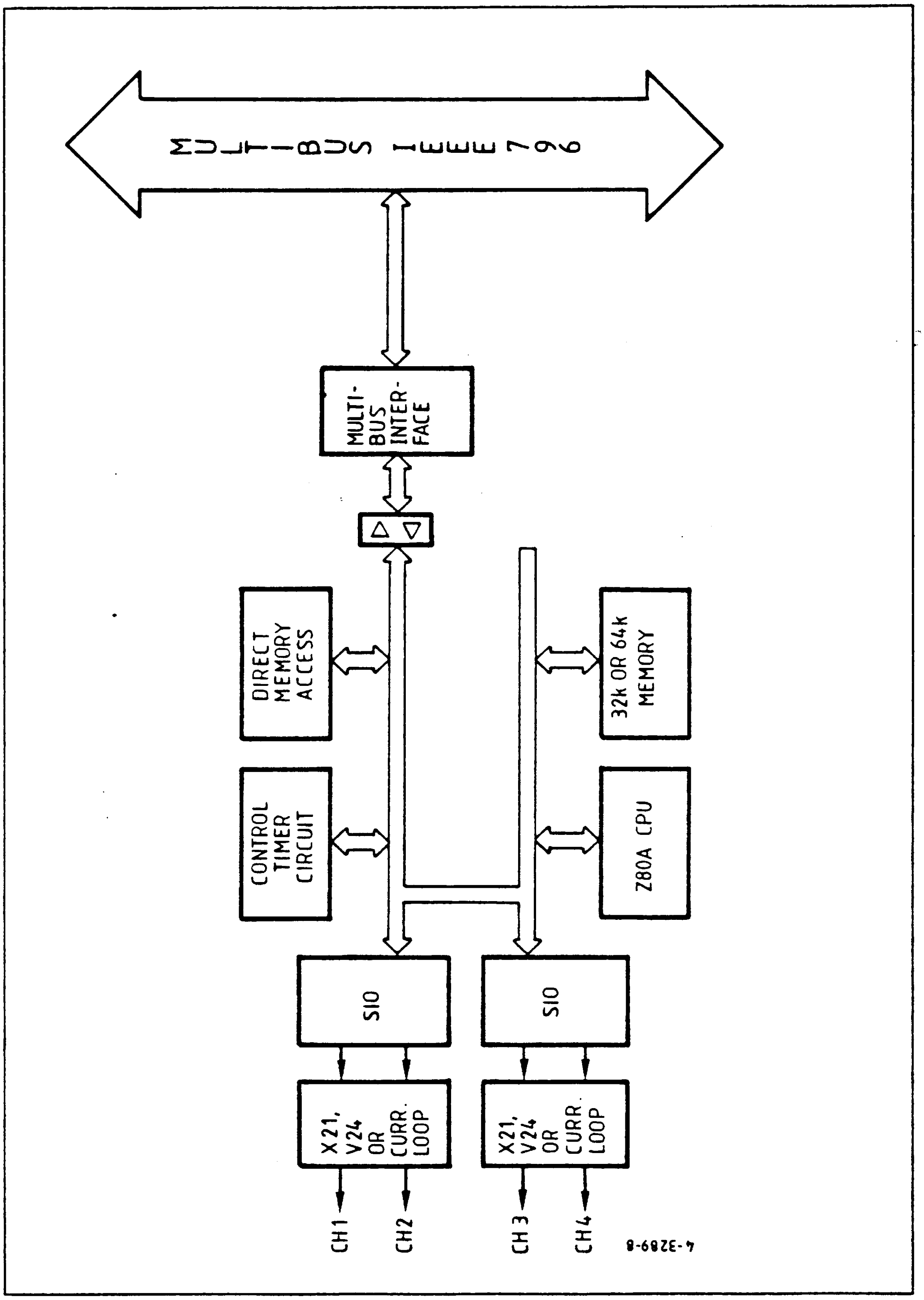
Two serial interfaces are included on the MP^2. One is a full modem interface, capable of synchronous or asynchronous operation with HDLC, SDLC, BSC, etc. The other is a limited interface for use with a terminal. In the integrated workstation this latter interface connects to the VDU section of the CR8 workstation. The signals and timing on both interfaces adhere to CCITT V24 recommendations.

2.1.2 System Communications Interface (SCI)

One feature of the CR8 which may be of interest particularly to systems houses is that by merely plugging in one or more of these SCI boards and loading new software from a floppy diskette, a simple CR8 desk-top computer is instantly converted to a sophisticated multi-processor communications device. With the help of the SCI it can now handle a variety of communications protocols such as X25, using software that has already been developed.

Similarly, a user of the simple single-user CR8 in, say, an accounting or word processing application can very simply upgrade to the multi-user model. By merely inserting this (same) SCI board and loading new software, the single-user system becomes a system with 6 serial channels, 4 of which are independently strap-selectable to X21/DTE, V24/DTE/DCE, or 20 mA current loop. By adding another SCI he can then add a further four serial channels. Thus in the maximum configuration, 10 terminals could be attached to the CR8 running the multi-user MP/M operating system, though generally the users would require a printer or two, leaving room for 9 or 8 terminals in all.

The SCI board fills one Multibus slot. It contains a Z80A microprocessor which runs software downloaded to it when the CR8 is booted. Certain details of the channels that it supports are programmable - baud rate, data length, number of stop bits per character and parity details. Other options are selected by switches or jumpers.



SCI FUNCTIONAL DIAGRAM

2.2 Software

2.2.1 Operating Systems

The merits of the CP/M and MP/M operating systems that are available on the CR8 have been described elsewhere in this and many other documents. Suffice it to say here that they are well-proven operating systems which are very widely used throughout the industry. Thousands of programs are available for them, and therefore, for the CR8.

What is different about the CR8 is the SCRIMP/M operating system, and the way that work normally done by CP/M or MP/M is offloaded to SCRIMP/M which runs on a separate processor. Thus the amount of useful work that can be done on the main processor with CP/M or MP/M is greatly increased compared with the more usual single processor/operating system combination. Furthermore, SCRIMP/M can handle real-time applications that simply could not be implemented under MP/M, since the latter slices time too slowly for fast real-time interaction. A process under MP/M might typically have the CPU for about 15 msec. A time slot of much less than this would lead to 'thrashing' (perpetual swapping with little useful work being done) since there is a fixed overhead of 2 or 3 msec which is required to perform the swap. Because SCRIMP/M is a lot simpler than MP/M - it does no file handling (such tasks can be handed over to CP/M or MP/M if required) - then it has a much smaller overhead when swapping between processes, so can afford to slice time into periods of around 2 or 3 msec with an overhead per swap of around 200 microseconds. Thus one could say it is very roughly 10 times more responsive than MP/M.

SCRIMP/M can be used either as an autonomous operating system, or it may be considered as a job running under CP/M or MP/M (and of course in turn has jobs running under itself). It should be stressed that SCRIMP/M is a multi-programming operating system which can manage several real time processes simultaneously. It is used 'internally' within the CR8 to handle the time-critical parts of device-driving (terminals, floppy discs etc.), and at the same time can be running real-time applications such as process control. In addition it is particularly suited to handling the lower levels of protocol in communications applications. SCRIMP/M generally runs in the Z80s - on the MP^2 board for level 3 X25 protocol, floppy disc

driving, 2 serial lines, PAM, X-NET, and applications; and simultaneously on the SCI board Z80 for 4 more serial lines, and X25 levels 1 and 2.
Finally, users of CR's CR80 minicomputer should note that device drivers and communication protocol software written for XMOS (which is the central part of SCRIMP/M) in the LTU (Line Termination Unit) can be transferred to the CR8 with only very minor modifications. Further details of SCRIMP/M are given in section 5.9.

2.2.2 Communications Software

The CR8 can be supplied with software that implements various protocols in order to allow communication between IBM mainframes and ANSI X3.64 terminals.

2.2.2.1 IBM 2780 Emulation

The CR8 may be supplied with emulation software so that it acts as an IBM 2780 Data Transmission Terminal. The card reading, card punching and printing operations of the IBM 2780 are simulated on the CR8 as disc read or write operations. The BSC (Binary Synchronous Communications) protocol used by the IBM 2780 is implemented by this software. Furthermore, the CR8 can of course be used to create or edit the JCL or data files needed for the IBM batch job.

2.2.2.2 IBM 3274 Emulation

Software also exists to emulate IBM's 3274 end-to-end protocol which defines how an IBM mainframe communicates with a terminal through a Control Unit. The CR8 acts in the role of an IBM Control Unit. Several asynchronous terminals are connected via SCI boards within the CR8. This IBM terminal emulation can take place via CR's X-NET local area network if required.

2.2.3 Applications Software

The following applications packages have been 'tailored' for use on the CR8 and are available from CR; of course, there are hundreds of other programs available for CP/M or MP/M from other suppliers. Furthermore CR's range of software is continuously being added to and updated. See the list at the end of section 1.

2.2.3.1 WORD PROCESSING PACKAGE (OPTION 64)

Virtually any typing task can be carried out with this powerful editor. The main features of the editing functions include: automatic buffering, on-screen text formatting, word-wrap, flexible find & replace commands, on-screen help, dynamic page break display, precise control of text format and hyphen help.

The part of the document being entered or corrected is always shown on the terminal screen, additions and corrections being immediately displayed. The text entry is accomplished merely by typing the desired text. Command functions such as cursor motion or deletion of text are immediately visible on the screen.

Automatic Disk Buffering (Temporary Automatic Storage). The size of a document is not limiteed by the amount RAM in the CR8, but by the disk capacity (approx. 200 A4 pages). The text is brought into RAM as required, with no operator intervention or concern.

On-Screen Formatting. Text is displayed on the screen as it will appear when printed, allowing review and correction before printing. On-screen text formatting is accomplished with the aid of these features: word-wrap, automatic margin setting, justification, variable line spacing, centring and paragraph re-form.

Word-wrap. Paragraphs may be entered at high speed without hitting the RETURN key. When a word exceeds the right margin, word-wrap automatically moves the whole word to the next line and justifies and re-displays the line just completed. The RETURN key is only used to indicate the end of a paragraph, blank line, or other points where a permanent line break is desired. The user need not pause to think about the end of the line, nor feel the necessity of looking up from the material being entered.

Flexible "Find" and "Replace" Commands. Searching operations can be done as many times as required, globally (on the entire document), on whole words only, ignoring case (upper or lower), and selectively (you will be asked whether replacement is to be performed in each instance).

On-Screen Help. During editing, a menu of commands will appear at the top of the screen, which may be withdrawn to give additional text display area. If a character is entered, after a short pause the menu automatically changes to show all commands that begin with that character. Furthermore the user can select one of 4 levels of help offering different amounts of detail.

Dynamic Page Breaks. Within certain limitations, the page breaks that will occour at printout are indicated on the screen during editing, and change appropriately in response to every insertion or deletion. Thus it is not necessary to print in order to review page layout or page break positions.

Printing. A useful feature of the printing function is that whilst one document is being printed, another can be edited - i.e. concurrent printing is provided. The print formatting features also include page formatting, pagination control, special effects, microspace justification and operator options.

Automatic Margin Setting, Justification, Line Spacing and Centring. With word-wrap in effect, each completed line is automatically adjusted to fit the left and right margins, justified (right-aligned) unless you selected ragged-right format, and optionally double or triple spaced. A line of text can be centred between the current left and right margins with a keystroke command.

Paragraph Re-form. The text from the cursor position to the end of the paragraph may be "re-formed" on command to change the margins or line spacing, to change from ragged right to justified or vice versa, or to clean up after alterations.

Hyphen Help. Text Master can identify desirable places to divide a word between lines with a hyphen. At each occurrence, the user may decide whether to hyphenate, and optionally adjust the hyphen position. Hyphen-help inserts temporary hyphens which do not print if moved to mid-line by later reforming.

Special Print Effects. In response to special characters embedded into the document, Word Star can <u>underline</u>, double strike, **boldface**, superscript, and subscript. These special characters that are used to invoke enhancements are displayed at the beginning and at the end of the affected text, during editing.

Type Ahead Buffer. This feature enables the user to type faster than the screen display can respond; the keyboard strokes are saved until they can be processed.

Scrolling. The document being created can be longer or wider than the VDU screen; the screen can be used as a movable "window" onto the text.

Column Mode. This feature is very useful for moving, copying or deleting columns of text and numbers. Column mode is particularly useful in creating and editing tables.

Screen Highlights. The CR8 uses reverse video to differentiate menus and special control characters from the main text being entered.

In addition to these standard features, four optional programs are available. The proof-reading and mailing options cannot be used "stand alone".

Mailing (Option 71). This is used to produce standard documents. It is designed to save a great deal of time with repetitive documents, standard form letters etc. Some examples are: - a report or proposal that goes to different organisations where standard text is used in a variety of situations adding different introductory material and arranged in different order; or where the name and address change, while the rest of the letter stays the same; and mailing lables, where everything changes except the format.

The Mailing program also makes it possible to chain several files during printout, with continuous page numbering, and to print both letters and envelopes from the same mailing list with a single command. The mailing program also has all the normal print functions available with the main word processing program. The one exception is that it cannot print a file while editing another.

Proof-reading (Option 65). The Proof-reading program is designed to check for spelling errors and any typing mistakes in a file. It does this by taking each word within the file and matching it to the words in its dictionary. In the event that the word isn't listed in the dictionary (or dictionaries), the program assumes the word is misspelled.

The program's dictionary contains over 20,000 words. However, it does not include all the words one might use, such as names of people, places, technical terms and obscure words. Thus it is bound to pitch these out as well as words that are genuinely misspelled. Dictionary entries may be added or removed, or whole new dictionaries may be created.

Data Input (Option 67) and **Sorting (Option 69).** These two word processing options are covered elsewhere (sections 2.2.3.5 & 2.2.3.3), since, unlike the other two, they can be used independently of the word processing package.

2.2.3.2 Calculation & Financial Planning.

This is a very flexible and generally-applicable "spreadsheet" program which can be used for report generation, complex calculations and cost planning. It is in fact a programmable program, in that the user can set it up to perform certain calculations (including conditional statements). The calculations may then be repeated using different parameters.

The user designs his own report or form - a large sheet of text and/or numbers. It can have up to 63 colunms and 254 lines of entries (though the overall total is 600 entries), where an entry is a text string of up to 116 characters or a number of up to 16 significant figures (maximum $\pm 10^{63}$). Entries can be defined as fixed text or numbers, or variables. In the latter case the value can be accepted by typing it in each time a new calculation is done or it can be extracted directly from a data file; alternatively it can be some mathematical function of other entries on the form. These mathematical functions can call any of the following built-in functions:

Addition, subtraction
Multiplication, division
Raising to a power
Sum of arguments
Number of values in argument list

Mean value
Maximum, minimum value
Logarithms (base e or 10)
Square root
Absolute value
Exponential function
Logical expressions (AND, OR, IF, etc.)
Integer value of
Table lookup
Net Present Value of cash returns at a given discount
PI
Trigonometric function (SIN, ASIN, etc.)

So, using these functions to do a calculation in stages, it is easy to see on the screen **how** the calculation was done - giving greater credibility to the final report! For large or complex calculations that do not fit onto the screen, it is possible to "roam" over the report so that the screen is used as a window onto a part of the large report. Furthermore, whole lines or columns of particular interest may be optionally "fixed" into the display so that these do not disappear when the display is scrolled.

The package is so flexible and powerful that it is impossible to give a comprehensive list of possible applications; however the following gives a taste of the possibilities:

Balance sheets & accounting generally
Tax estimation
Market share analysis
Statistic calculation in scientific experimentation such as analysis of variance, correlation.
Data storage and retrieval (especially where there is a lot of different "facts" relating to each item).
Any application that could be done slowly with a packet calculator and reams of paper.

Planning & Report Generation Package (Option 81)

This package serves much the same function as the one just described (Option 66). The user does not create the report "on the screen" as he would with the other package. This package acts more like a traditional programming language, where a separate program is written which describes the desired report. It does however, have a few more mathematical accounting functions such as Internal rate of return, and four different types of depreciation calculation. It can also do a table lookup (a useful programming feature). It lacks the Linear Regression function that Option 66 has. This package is more flexible in the style and format of final report that it can generate; the 66 Option is more 'User Friendly' but gives a more limited range of reports as a result. The 81 Option could in fact be considered as an inexpensive alternative to Cobol.

2.2.3.3 FILE SORTING (Option 69)

This is a sort/merge package which can be used either as a stand-alone program, or in a form which in accessible from the user's own software (in BASIC, FORTRAN, COBOL or assembler). It can sort up to 32 input files and simultaneously merge these into a single output file. Alternatively, by merging one file into one file, use can be made of the extensive file conversion features without sorting.

The data format is very flexible. Files can be ASCII, binary, BCD (Cobol packed decimal), or EBCDIC. The user may even define his own collating sequence. It can handle signed and unsigned integers, or single and double precision floating point in both binary and character form. Moreover in the latter case the numbers can be in a variety of Exponential or floating point formats. Characters can have their case ignored or can even be read backwards.

The file format is equally flexible. The program supports records of up to 4096 characters, maximum 65K records per file. Records may be fixed length, variable length (with byte count or carriage-return-delimited). COBOL "relative" file type is also supported. Fields within a record may be of fixed length, or of variable length delimited by commas.

The keys for sorting can be up to 32 in number and each key's attributes may be independently defined (from the large variety of formats mentioned previously).

Another feature of this package is the "Exclude" and "Select" commands. By using these the user can, for example, ignore input data within a certain range of values, or data that is not in ASCII, or numbers less than a certain value, and so on. By this means the package can be used for data retrieval rather than (or as well as) sorting.

2.2.3.4 PROJECT MANAGEMENT (Option 70)

This package uses the technique of Critical Path Analysis as an aid in the management of small projects. The tradeoffs between time, manpower and money may be investigated. Parts of the project that constitute "bottlenecks" may be identified. That is, the program highlights those activities which if completed late will delay the whole project. Furthermore, it can generate a report on the project that can help to clarify what has to be done. Unlike many critical path analysis packages, this package is interactive and simple to use (and does not require a vast mainframe in which to run). So its aim is both to clarify the project to the manager and to produce easy-to-read reports on the project so the manager can more easily explain to others where the "vital links" in the project lie. It is intended for the analysis of small projects (or for a superficial view of a large project). Of course, it is ideal for use on a small subset of a large project.

As well as analysing the use of time, this package can estimate the costs of a project; also it can instantly re-estimate costs (or time) if the user specifies different starting conditions.

Calculations take place virtually instantaneously. The program can handle more than 100 activities in the project. Manpower is divided into nine different skill categories. The user can choose the time unit as being hours, days, weeks, months, quaters, or fiscal quarters; the longest project is 9999 units. It can allow for holidays, and the normal working week can be defined.

The printed output from the program consists of four reports. The first defines the initial parameters such as manpower-skill-types available and dates of public holidays. The second is a detailed description of each task into which the project has been divided in terms of timings and estimated cost. The third report is a timetable of the whole project with the critical path (and non-critical timings) clearly marked. The final report consists of a table showing job number, job name, duration, early and late timings, slack time, costs, prerequisite jobs, and number of men needed in each skill category. If all of these will not fit across the printer, the user can divide them into separate tables as required.

2.2.3.5 DATA BASE SYSTEMS FOR THE CR8

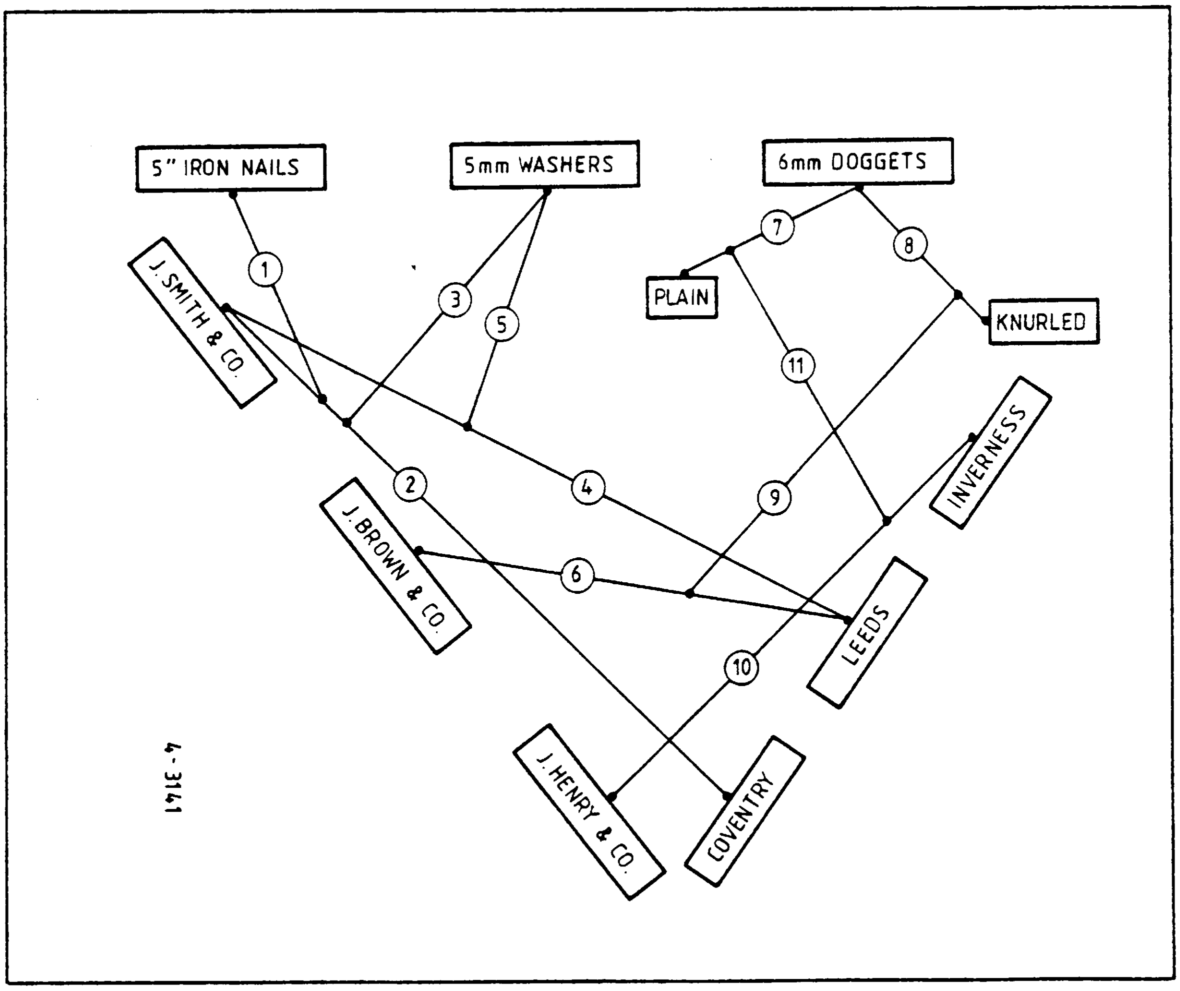
Three different packages are available from CR to cover what can loosely be termed data management - i.e. the storage and retrieval of large amounts of information in a flexible way. The three packages are: the simple data entry and retrieval system (Opt. 67), dBase II, and MDBS III. The third of these is by far the most sophisticated - it is the only one which is designed to handle many-to-many relationships. For instance, imagine a stock control application:

ITEM	SUPPLIER	LOCATION
5" iron nails	J. Smith & Co	Coventry
5mm washers	J. Smith & Co	Coventry
5mm washers	J. Smith & Co	Leeds
6mm knurled doggets	J. Brown & Co	Leeds
6mm plain doggets	J. Henry & Co	Inverness

2 DIMENSIONAL DATA REPRESENTATION

All 3 systems could store and retrieve information set out as above. Notice however that six fields have to be filled in order to record four facts, namely that 5mm washers can be bought from J. Smith in Coventry and Leeds. The duplication of "5mm washers" and J. Smith" is necessary because a 2-dimensional table such as this can only record one-to-one relationships (such as "5mm washers"/"Leeds" or "5mm washers"/"J. Smith & Co.") This duplication of information is both wasteful of disc space, and can cause much confusion. For example, J. Smith & Co. might stop selling 5mm washers, so the user deletes record 2 from the data base but forgets to delete record 3. In this simple example the duplication of information is very small; in real life it would be much greater, in fact to the point where a "table" format would be impractical for many applications.

Of the three systems, only MDBS is designed to cater for many-to-many relationships like this:



MULTI-DIMENSIONAL DATA REPRESENTATION

Our 15-entry table now can be represented by only 11 relationships. In a "real-world" example the saving would be much greater since the table might now have say 30 columns. One new item to be stocked would now require an extra 30 items to be recorded in the data base; in the multi-dimensional case only one new relationship (possibly more, but probably not as many as thirty) would need to be recorded. Furthermore, information retrieval can be made much more flexible and efficient since the database software can just 'follow the relationships'. For example (please refer to the diagram above), an enquirer might ask "Is there any point in going to Leeds?" The database system could follow the relationships 6, 9, and 8 for the reply "Yes, you could go to Browns & get knurled doggets", and relationships 4 and 5 for "You could also go to Smiths and get 5mm washers" What is more, the software would immediately know that there was nothing else in Leeds (since only 4 and 6 connect to it) so it need waste no further time in ploughing through its massive memory banks.

So, for applications where the data have few inter-relationships (or where such relationships are of no interest), or where the overall amount of data is fairly small, then the data entry system or dBase II would be the best choices. The data entry system is very convenient to use and has advanced data checking (on input). It is not very programmable in the sense that its only interaction with the user's own software is that it can make a file which can be accessed via CP/M or MP/M.

dBase II however is programmable "within itself" - that is the user can write data manipulation programs but he must use dBase II's special command language. For complete integration with a user's own software in Pascal, Cobol, etc. then the only choice is MDBS.

Further details on each of the 3 packages now follow.

The Data Entry & Retrieval System (Option 67)

This is a data-handling program that allows the user to enter, retrieve, and update data in the form of text and/or numbers. It includes a program called with which the user can create the form to his own design. That form is then filled in (and filed on disc) using the main program.

The program is easy to use since a menu of commands is always displayed on the screen; there is also a Help facility to obtain further details without having to refer to the manual. The data entry package is compatible with the word processor (opt. 64) or can easily be incorporated into existing software written in e.g. BASIC, FORTRAN or COBOL.

The form generation program is very flexible in the style of form that it can handle. The form can be several pages long (depending on system memory size) or up to three pages wide. Furthermore, it is easy to insert built-in checks for each data field. For instance one can specify that when a datum is entered in a particular field it must be 8 letters or up to 6 letters or a number between 14 and 25 or whatever. One can even specify a list of valid entries for a field against which any datum entered must be checked. It is also possible to automatically fill certain fields by performing arithmetic on the contents of other fields, much as in the calculation package (opt. 66), or by copying across character strings from one field to another.

The program supports a batch mode whereby data is entered into a temporary file which can be checked - preferably by a different person - prior to updating the data file.

On the data retrieval side, the user can supply a "mask" to be compared with every record so that he could display, for example, the records for all customers in a given town, or all customers called Smith - in short all records that conform to the user-specified "template record". Alternatively the user can "leaf through" the whole data-base in user-supplied index-number order or in an unspecified order (which is more convenient for the CR8). There are of course facilities for printing out the selected information.

Data Base Management System - dBASE II (Option 74)

This is rather more simple than MDBS. It supports a two-dimensional "table" structure of data. All communication with the system is via its own high-level language; any applications programs for the package must be written in that language. The user may interrogate the data base with a simple interactive dialogue, or combine commands (i.e. dBASEII language elements) into command files (programs) for the excecution of more complex procedures. The "form" is set up using an interative dialogue, then data is entered into the data base by filling in the blanks in the pre-defined form. The cursor automatically moves to the next box to be filled in. Simple data-type checking takes place during data entry and an audible warning given if data is of the wrong type.

One feature of the system is that it is possible to add data types or categories to an existing data base, without affecting programs already written - the new fields are simply ignored by the old programs.

dBASEII can generate reports from one or more databases; the report-writing software can do simple arithmetic also.

Data Base Management System - MDBS III

MDBS III (Micro Data Base System Version III) is a large and sophisticated data base management system which is available for the CR8. It is a **relational** data base system, that is, the relationships between data are recorded as well as the data themselves. These relationships can be one-to-many (like CODASYL or a tre-structured disc filing system), one-to-one (pairs of related items) or many-to-many (just like the real world!) Furthermore, sets may be recursive - that is, data may be indirectly related to themselves.

Up to 255 record types (fixed or variable length) are supported. Each record can have up to 65535 fields (data items) within it. Various schemes for the ordering of entries within a set are supported, such as alphabetical, FIFO, LIFO, or immaterial.

The MDBS package is divided into the following functionally separate parts:

Data Description Language
Data Manipulation Language
Query Language
Recovery & transaction logging
Design modification utility

Data description language (DDL). This is used to define the structure of the database, this definition being known as the schema. It defines both what the data types are, and the logical relationships between them. It can optionally also be used to define the physical structure of the data base - i.e. which data are placed on which areas of which disks, and define passwords and data integrity schemes. DDL is an interactive language that includes an editor. When the data description has been defined by the user, DDL "compiles" it into an initialized data base. This process can be done interactively or in batch mode.

Once the data base has been set up in this way, there are two ways of accessing it - via application programs (DMS) or directly from the end-user to the data base (QRS).

Data Manipulation Language (DML, DMS). The MDBS package includes a subroutine library called DMS which implements the Data Manipulation Language. The user can write his own application in, for instance, BASIC, PASCAL, COBOL, or FORTRAN. The application program would interact with the user, while calls to DML would handle all data retrieval storage and manipulation. Thus the application program would typically display a menu and translate user commands into DML calls. Alternatively one could use IDML which is an interactive version of DML. This program, like QRS, is also useful for debugging DML programs.

Query Language (QRS). MDBS also supports an interface directly from the "naive user" to the database. This language is similar to English. Such a user need not be a programmer but he must know the overall structure of the data base. An example of a QRS statement would be:

```
LIST NAME, PHONE FOR DEPT = 27 AND AGE >20 THRU DEPTS,
EMPLOYEES
```

This means "list out the name and phone number of all employees in Department 27 who are over 20 years old". The "THRU DEPTS, EMPLOYEES" clause is where the knowledge of the structure comes in - the user must tell MDBS that the information he wants has been defined as pertaining to employees which in term are related to departments. Notice also the use of the conditional clause:

```
FOR DEPT = 27 AND AGE >20
```

The QRS language also supports a report generation feature.
As well as providing a direct user interface to the data base, QRS is very useful for testing and debugging programs written using the DDL routines.

Recovery and Transaction Logging (RTL). This automatically records all details of all transactions since the last data base back-up operation. Thus in the event of system failure an up-to-date data base may be obtained using the back-up plus the log. In addition the log provides a useful record of day-to-day usage.

Design Modification Utility (DMU). It is possible to alter the original schema of the data base when it already contains data, using the DMU. It also shows how much of the available disc space has been used.

2.2.4 PROGRAMMING LANGUAGES FOR THE CR8

All language processors for the CR8 are optional apart from ASM and/or ASM86.

ASM. This is the simple assembler for the Z80 processor. It is a standard CP/M 2 or MP/M II utility. It accepts standard Z80 mnemonics. It includes an optional assembly directive (i.e. IF and ENDIF statements surrounding a block of code which is assembled or not according to the value of the IF expression).

ASM86. Similar to ASM but generates machine code for the 8088 processor. It runs under CP/M 86 or MP/M 86. Another version is available which runs under CP/M 2.2 or MP/M II; this is a cross assembler for the 8088.

RMAC. This is an assembler with a macro generation feature. It produces relocatable object code for the MP/M II system.

Pascal/MT+. This Pascal includes every feature specified by ISO DPS/7185, plus the following extensions:

BYTE, WORD and STRING pedefined scalars
Expressions may include the predeclared INP array.
User may make assignments to the predeclared OUT array.
The facility to permit recursion may be switched on or off.
Boolean operations on integers (using &,!,? for AND, OR, NOT respectively).
ELSE in CASE statements
External and Interrupt procedures
Bit and byte manipulation
Sophisticated file handling including random access
Move and Fill memory areas
Functions to return address and size of variables
String concatenation, copy and comparison
Heap management using ISO standard or other procedures.
Support for BCD and floating point real data types.
Ability to compile one module at a time.
Facility to send character I/O through via user's own software.

Pascal/MT+ can be used for real-time applications as well as the more usual data processing tasks (which can include systems software such as compilers). Furthermore it can generate code that can be used **without** an operating system, if desired. Note also the Speed Programming Package, described in the next section.

SPP. This Speed Programming Package comprises various program development aids for use with Pascal/MT+. SPP has the following features:

- o Its supervisory program is supplied in source code to it can be adapted to any terminal or have new functions added.
- o A screen-oriented editor which includes a syntax checker. The latter can be switched on or off as required.

- o Spelling checker for the user's identifiers. This pinpoints those that have only been used once (and are therefore probably mistakes); it gives a frequency count of all identifiers.
- o Incremental disk backup utility.
- o Logging of source modifications.

BASIC. Four types of CBASIC are available from CR for use on the CR8. All support the same language syntax, thus only one source need be maintained for either CPU. The following table defines the four types:

	INTERPRETER	COMPILER
Z80 cpu	CBASIC	CB-80
8088 cpu	CBASIC-86	CB-86

The two referred to above as interpreters are not strictly interpreters; the source must be processed by CBASIC or CBASIC-86 to produce an intermediate file in "pseudo-code". This intermediate file is then interpreted by the Run-time monitor at run-time.

CBASIC (i.e. these 4 BASICs) is the most widely used variety of BASIC in the industry so there are thousands of programs commercially available that will run with no alteration at all.

CBASIC processes floating point numbers in binary coded decimal (BCD) format which maintains accuracy to 14 significant figures. Furthermore, integer arithmetic may be specified where appropriate. These two features imply that arithmetic will be faster and more accurate in comparison with other BASIC implementations.

When used with MP/M the compiled versions support locking and unlocking of files or individual records. Also, access to multiple printers is possible.

Characters strings are dynamically allocated by all 4 CBASICs. This means that string handling is both fast and efficient in memory use. In the compiled versions they can be up to 32,000 bytes long.

Other features of CBASIC include tracing and cross reference listing for debugging, CALL statements (with parameters), overlaying code (to allow programs to run that are larger than the available memory), and error trapping.

Fortran. This compiler supports all of the ANSI X3.9-1966 standard apart from the COMPLEX data type. It also supports the following features.

- o Signed integer LOGICAL variables
- o INTEGER*4 32-bit integer variables
- o LOGICAL DO loop counters for faster execution when applicable
- o Reals and integers can be mixed in an expression.
- o Hexadecimal constants are permitted.
- o Hollerith literals are permitted.
- o Logical operations on integers
- o READ or WRITE statements can include a label to transfer to on error or end-of-file.
- o ENCODE and DECODE statements for FORMAT operations to memory.
- o IMPLICIT statement accepted.
- o INCLUDE statement to insert source code in-line from another file. Very useful for COMMON BLOCKS.
- o Supports random access of up to 65,535 records in a file larger than 8 Mbytes.

The compiler optimises the code in various ways (such as common sub-expression elimination). It also includes an extensive subroutine library.

The user may write his own I/O device driver for each Logical Unit Number in order to access his own "fancy" peripherals.

The program package includes a macro assembler, a linkage editor and a cross-reference lister.

Cobol. The Cobol package available from CR is the industry-standard RM/COBOL, developed by the Ryan-McFarland Corporation. It conforms to the ANSI X3.23-1974 standard.

The same source code can be processed by RM/COBOL to provide object code for any of the four CR8 operating systems (CP/M-86, CP/M 2.2, MP/M-86, MP/M II). This feature facilitates the transporting of code from one machine or operating system to another. Furthermore, RM/COBOL supports record locking. This allows several users to access the same file simultaneously without fear of confusion.

RM/COBOL also supports alternate keys when using indexed data files. Any field in the record can be used as the index key.

The compiler may be supplied with the COGEN utility. This allows the programmer to interactively design the desired screen or report format; then COGEN itself writes the required RM/COBOL source code to define that screen or report. This obviously cuts program development time and to some extent the level of skill required by the user or programmer.

3. SYSTEM START UP & OPERATION

3.1 INSTALLATION

Integrated Workstation Models

The CR8 may be used in any normal office environment. After unpacking it, you must check that the rated voltage matches that of your mains supply. The rated voltage is stated on a panel at the rear of the main workstation unit. In the unlikely event of it being wrong, the rear panel will have to be removed and the voltage setting changed using the switch at the top of the rear compartment. The rear panel is removed by removing the two screws at the rear then tilting the panel as though is were hinged where it joins the rest of the cabinet at the top.

Notice the two small sets of Dip switches on the left hand side at the back. These are clearly described on a label nearby. These should normally be left alone, unless you want to change the cursor shape or blinking, or switch to inverse video.

Make sure that the keyboard is connected to the main part of the workstation using the coiled lead supplied. See illustration in section 5.2.2 for positions of sockets.

Cabinet Models

Unpack and place the unit on a desk top or fit it into a 19" rack. Take care to install it where there is no obstruction to the flow of air through the fans on the right hand side of the unit (viewed from the front).

Before connecting to the mains check that the voltage setting agrees with your local supply. The setting is stated on a plate at the rear. If it is wrong, it will be necessary to open the cabinet and make internal adjustments to the power supply unit.

The illustration at the end of section 5.1.1 shows the position of the various sockets at the rear of the CR8 Cabinet. Plug into the mains, and connect printer and terminal.

Power Up and Down

With either type of CR8, it should now only be necessary to switch on the mains. When the VDU(s) has warmed up, the following message, or something similar, should appear on the screen.

```
CR8 BOOTLOADER Vers. 3.0

CHRISTIAN ROVSING A/S, 64K CP/M VER. 2.2 820105
SEAGATE ST506 HARD DISK / 5 1/4 MINI FLOPPY BIOS,V1.0
B>
```

It means that the system has 'booted' itself and is now ready for use.

Before you use any software package supplied by CR (or anyone else) be sure to make a copy of it first (see the whole of section 3.4)

The user should familiarise himself thoroughly with the whole of section 3 and 4 of this manual, plus any documentation supplied with specific software packages he might have purchased, before he types anything on the keyboard. It is possible to delete expensive software packages by typing the wrong things (especially but not only, the ERA and FORMAT commands)!

N.B. Never switch off the mains to the CR8 when the light on the front of any of the disk drives is lit. Doing so could corrupt data or programs stored on those disks.

3.2 OPERATION OF TERMINALS

3.2.1 Terminal Controls

CR8 Integrated Workstation

This is rather a special case, as the terminal is also the computer itself; the reader is therefore referred to section 5.2, in particular part 5.2.2.

CR7 Comfort

The mains switch, brightness and contrast controls are all situated at the rear of the main part of the unit.

CR5 Compass

The mains switch, brightness and contrast controls are all situated at the front of the terminal, to the right hand side of the screen.

Tandberg TDV

The mains switch and brightness controls are at the front of the terminal, to the left side of the screen.

Setting the Baud Rate

The Baud Rate of the terminal must be set to that of the CR8. This is the speed at which data is transmitted between the processor and the terminal. This will be set in the factory to the correct value before delivery. If for any reason it has been changed, or needs to be changed, refer to section 5.10. See also section 4.7.

3.2.2 Danish & English Character Sets

All CR8 Workstations and terminals will normally be capable of displaying both English and Danish characters. Keyboards are available for both languages; the layout of the special Danish keys is slightly different on the two types of keyboard, however, as far as the terminal is concerned, both types 'seem' the same. That is, no software changes need be made if swapping from one type of keyboard to the other.

Both types of keyboard will normally be in 'English' mode when first switched on (however, see section 5.10). To set it to Danish mode, press CTRL-N (hold down the CTRL key, then press N simultaneously), followed by the RETURN key. Similarly, to convert from Danish to English characters, press CTRL-O and RETURN in response to the operating system prompt (a '>') symbol.

Often the user need not be concerned with this since his software packages, if in Danish versions, will automatically send the CTRL-N sequence to the terminal without the user having to type it in.

Keys that Differ between the Danish & English Character Sets

English Mode	Danish Mode
[	Æ
{	æ
]	Å
}	å
\	Ø
|	ø

3.3 OPERATING THE PRINTERS

A few details are given below of the various printers supplied by CR. Every printer is supplied with its own handbook to which the reader is referred for full operating instructions.

3.3.1 DOT-MATIX PRINTERS

Opt. 40 Dot-Matix. This has a mains switch at the rear and one button at the front - the PRINT switch. It must be set to ON. The red lamp on the front panel indicates that the power is on.

Opt. 41A or 41B Dot-Matrix. This has a mains switch at the rear and the following switches and lights on the front panel.

POWER Light	: Indicates 'power on' when lit.
PAPER Light	: Lights when paper runs out
FORM LENGTH knob	: Set this to the length of the paper in use as follows:

Switch Position	Form Length (cm.)	Form Length (in.)
0	7.6	3
1	8.9	3.5
2	10.2	4
3	14.0	5.5
4	15.2	6
5	17.8	7
6	20.3	8
7	27.9	11
8	30.5	12
9	35.6	14

TOF SET: Use this button to set up the position of top-of-form. Move the paper to the desired position with the platen knob, set the printer off-line (see SEL button & light), then press TOF.

SEL. This toggles between on- and off-line each time the SEL-button is pressed. The SEL light is lit when the printer is on-line, i.e. ready for action. The SEL light will also go out if the paper runs out.

FORM FEED and **LINE FEED.** These only work if the printer is off-line. They move the paper up one page or one line respectively.

3.3.2 DAISY WHEEL PRINTERS

OPT. 42A DAISY-WHEEL

The mains switch is at the rear. When on, the POWER lamp is lit on the front panel. The other lights and buttons are as follows (left to right).

PRINT ON light is lit when printer is 'on-line'.

ALERT light is lit if:

The front cover is open, or
Paper has run out (if paper-out mechanism fitted), or
The ribbon has run out.

POWER Light. Lit when mains switched on.

STOP CONTINUE button. This toggles between on- and off-line.

LF Button. Moves the paper up one line, or of held down for more than 2 seconds, it causes continuous line feeds to occur.

PAGE ADV Button. Moves the paper to the next top-of-form.

SET PAGE. When pressed, the line at which the printer's head rests becomes the top-of-form position. The printer acknowledges by moving the head briefly to the right and back again.

Setting the paper length: Directly behind the lamps on the front panel (lift the cover) are two sets of tiny switches. Switches 6 to 9 in the right-hand bank of switches control the paper length. Set them to OPEN as per this table:

Switch Position	Form Length (cm.)	Form Length (in.)
6	2.5	1
7	5.1	2
8	10.2	4
9	20.3	8

The effect of the switches is additive; thus for example, switches 6 and 7 closed, and 8 and 9 open would give a form length of 12 inches. The printer must be turned off and on again after setting the paper switches.

OPT. 42B OR 42C DAISY-WHEEL

Mains switch is at the rear of the printer. The front panel has a numeric display which shows the number of the print column currently under the print head. It also displays error codes (if the first digit is 'E').

SELECT. Press the upper part of this switch to toggle between on- and off-line. When on-line, the light below this switch will be lit.

RESET/BREAK. If the upper portion is pressed, certain error conditions will be corrected. If the lower section is pressed, a special code is sent to the CR8.

GO TO TOF/SET TOF. If the top section is pressed, the paper will move to the next top-of-form. If the lower part is pressed, the current paper position will become the top-of-form.

MICRO LINE. This will move the paper up or down in 1/48 inch increments.

There are some other switches for setting the form length and so on. They are well labelled; they lie to the left of the buttons described above, though they are under the front cover.

3.4 DISKS, FLOPPY AND HARD

3.4.1 Use of Floppy Diskettes

A diskette, or floppy disk, is a thin plastic disc coated with magnetic recording material, used for storing quite large amounts or information.

The diskette is sealed in a cardboard envelope. The inner plastic disk rotates constantly inside the envelope when the diskette is in use.

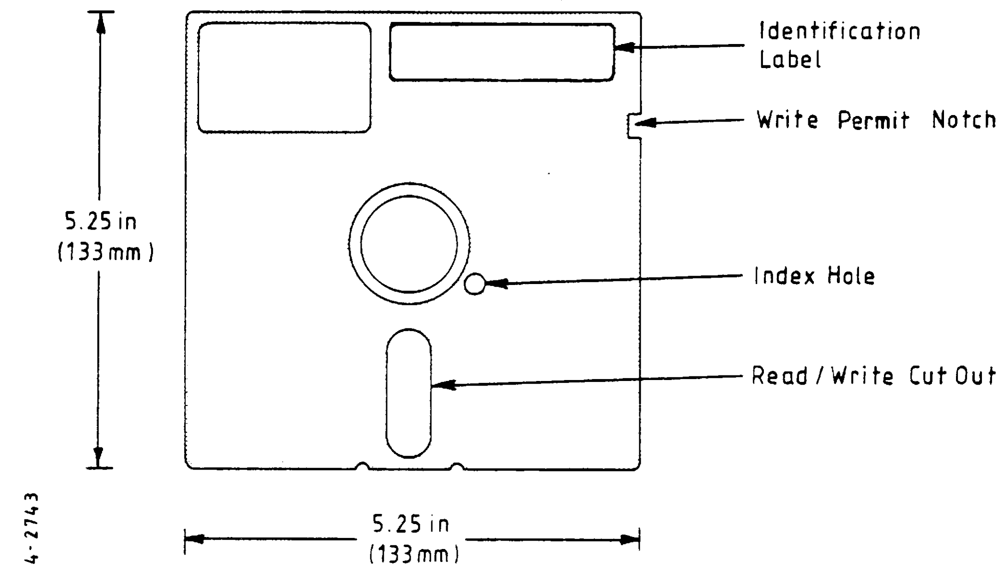


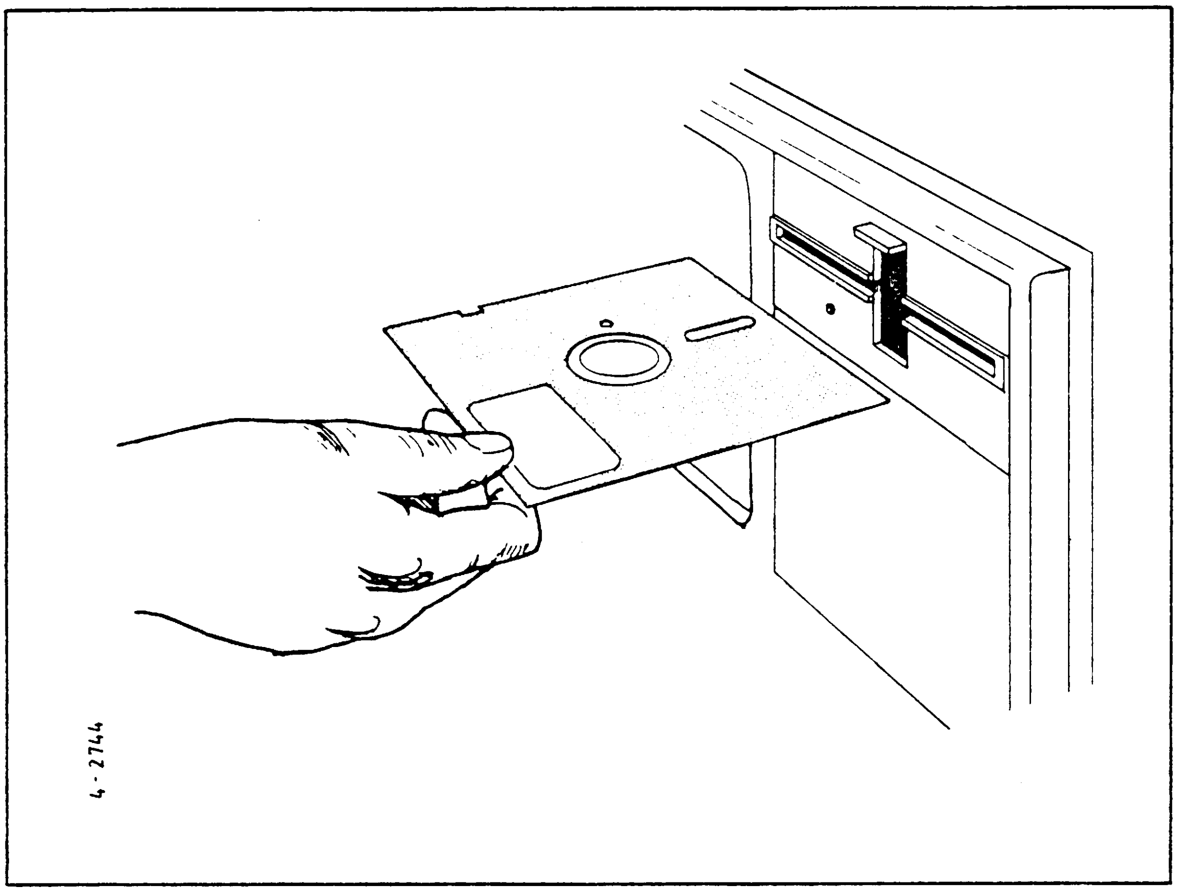
ILLUSTRATION OF A CR8 DISKETTE

Both sides of the diskette are used by the CR8, but the user must always insert it as shown on page 3-9.

The write permit notch may be used to prevent the diskette from being accidentally overwritten. When this notch is covered (with a piece of hard tape), the diskette can only be read from, not written to.

Inserting a Diskette

Lift the small lid on the disk drive up (it should be already up if the drive is empty). Then, holding the diskette the right way up - write protect notch on the left - slide the diskette into the drive slot completely to the end stop. Never force the diskette in. Now close the lid by pushing it gently down. Again, do not force it down. The diskette is now ready for use.



INSERTING A DISKETTE

A diskette is removed by simply lifting the lid and withdrawing the diskette. Remember to remove all your diskettes before the computer's power supply is turned off.

If the diskette is put in the wrong way round (or if it has not been formatted - see next section) then an error message with error code 10 will appear on the screen, but no harm will have been done.

Care and Storage of Diskettes

Diskettes last almost indefinitely if handled carefully. The following is a useful guide to the care of diskettes.

o Hold the cardboard envelope. When the diskette is removed from the protective paper sleeve do not touch the exposed surface or allow dust to settle on it.

o Do not bend diskettes.

o Insert gently. Do not force diskettes into drive slots.

o Avoid heat. Direct heat from strong sunlight or radiators may warp the diskette.

o Do not switch the computer ON/OFF with diskettes in the drives.

o Store safely. Specially made diskette storage boxes are available. These boxes hold the diskettes vertical, and may be locked for security.

o Use felt-tip pens. The identifying label on the diskette should never be written on with ball-point pens or sharp pencils. Whenever possible, write on the label before sticking it to the envelope.

3.4.2 FORMATTING DISKS

Before any disk (Winchester or floppy) can be used for the first time, it must be formatted. This means that certain signals are "written" onto the disk to define where the different areas to be used start and finish. If the disk has been used before, formatting it again causes the information on it to be lost, so, . . .

When to format a disk

1. Before using a brand new floppy diskette.
2. Before using a used diskette or disk which has no files on it that are of any further interest.
3. As an attempt to rectify a disk that is giving hardware errors (though all data will be lost).

Note that the Winchester disk in the CR8 will already be formatted since it contains the system files, thus in normal circumstances you will never need to format a Winchester.

How to format a disk

The formatting program supplied with the CR8 is called FORMAT. It can only be used to format floppy diskettes.

1. Check that the diskette is not write-protected. The notch must be present (see illustration at the beginning of section 3.2). Insert the diskette in the drive. Close the lid.

2. Type FORMAT and press RETURN after the prompt from CP/M (or MP/M).

3. A menu of choices appears on the screen. Enter option E to format a floppy disk in drive E, or A or B for the dual floppy system.

4. Type "Y" to the next question, then "G" to the next.

5. Disk formatting will now begin. During floppy formatting, the track number currently being formatted is displayed on the screen, counting up from 00 to 99 (hexadecimal).

```
TRACK**

FORMATTING COMPLETE
```

6. You may now format another diskette by entering "Y" in response to the next message. The program will return to step 3. Enter "N" to exit from the formatting program.

Be careful not to format a diskette which already has important information on it. Formatting will completely erase a diskette.

3.4.3 COPYING DISKS

Copying a complete floppy disk on the dual floppy system

The following procedure should be followed when making a copy of a complete floppy diskette, for example when saving your original system diskette. Assuming that the operating system is ready (displaying "A>") with the system diskette in its normal position in drive A. . . .

1. Type "FCOPY" and press RETURN.
2. Insert discs as instructed and answer the questions appropriately.

Copying single or multiple files
(Winchester/floppy or dual floppy systems)

To copy on a file-by-file basis, use CP/M's PIP utility, for example the command:

```
PIP E: = B: * . *
```

will copy all files from drive B (part of the Winchester) to drive E (the floppy in a Winchester/floppy system).

An example of copying a single file would be:

```
PIP E: = B: LETTER.DOC
```

This would copy the file LETTER.DOC from the Winchester to a floppy disk.

3.5 RUNNING A PROGRAM

The user must refer to the documentation applicable to his particular piece of software. However, generally speaking, his default disk is set to the disk on which the program lies (see section 4.2), and the program exists in a suitable form in a file of type .CMD, then by typing the file (program) name (excluding .CMD) and pressing RETURN, the program should run. Files of type .COM, .PRL may also be run in this way.

3.5.1 SYSTEM RESET

Integrated Workstation Models

If at any time you or one of your programs gets stuck, you can re-start the entire operating system as though it had just been switched on. For the dual floppy system, insert the system diskette in drive A and close the lid. Then for either system, press the Reset button on the front of the CR8 (below the disc drives).

Note that all information stored in memory (but not disk) prior to pressing the Reset will be lost; thus for example if you had been editing a file, then all changes since the last "save to disk" command would be lost.

For a more detailed description of what happens in the hardware when 'reset' is pressed, see section 5.7.

Cabinet Models

There is no 'reset' button on the cabinet model CR8, but switching the power off and on again will have the same effect on the operating system. Remember that the system diskette should be in Drive A (dual floppy systems) and the lid open <u>before</u> operating the ON/OFF switch.

For a more detailed description of what happens in the hardware when the system is reset, see section 5.7.

Selecting the Boot Source

Normally the CR8 will decide for itself on which disk drive it will find the operating system (in order to boot it). However, system programmers and the like may prefer to select a particular drive if they have, for example, just generated a new system.

In order to manually select the source disk for the boot, hold down the space bar on the system console immediately after switching on or pressing 'Reset'. A menu will the appear on the screen. The required drive may then be selected by typing its drive letter (name) at the console.

4. OPERATING SYSTEMS

The operating system(s) provided with the CR8 will depend upon the processor type(s) and whether single or multi-user cabability is required. Full details were given at the beginning of section 1.3. Normally the user will be dealing with CP/M or MP/M, the latter being the multi-user version of the former operating system.

N.B. The information given in the following subsections applies to all four operating systems (i.e. CP/M2.2, CP/M-86, MP/MII and MP/M-86) unless stated otherwise.

Note also that the information given does not generally apply to SCRIMP/M. This is covered in section 5.9. The remainder of this section (section 4) deals with the user's interface with the operating system; it is a brief overview only, because full details may be obtained from Digital Research's own documentation or from other sources.

The CP/M and MP/M operating systems all present similar user and file handling interfaces. The user often need not know whether he is under CP/M control, having the entire machine, or under MP/M control, sharing the machine with other users. Many utility programs, such as PIP, the Peripheral Interchange Program, and ED the text editor program, operate almost identically in all systems.

4.1 Disk Files

The operating system organizes information on disk into named files. The operating system maintains a record (held on each disk) of which files are currently on the disk. This record, called the file DIRECTORY, contains at least one entry for each file. The directory entry contains the file name, file type, file attributes, user number and sector map. File type is designated by the user and has no significance to the operating system. File attributes are such factors as whether the file can be written to (or merely read), or whether it is a system file. Disk space for a file is not necessarily contiguous (i.e. a file may be divided into different parts dotted here and there about the disk). The purpose of the sector map is to record the positions of the various disk sectors that comprise the file.

The operating system does not need to know how big a file will become. When more storage space is needed, CP/M or MP/M will add more sectors to the file. If the file is deleted, the sectors are 'freed', and may be reallocated to another file.

The system can support up to 16 disk drives, referenced by the first 16 letters of the alphabet, A-P.

Files on each disk may also belong to one of 16 'user numbers'. Each console on the system (or the console) has a 'current user number', set arbitrarily on system start-up to be the same as the console number. The user number for a console may be changed by the operator at the console to any value between 0 and 15. The operator at a particular console may only access files belonging to the current user number for that console.

The user number facility is of greatest value in a multi-user system, but it is still useful when there is only one system console, to provide a logical division of files into different applications. For example, files used within a word-processing system may belong to one user number, while files used within a financial planning system may have a different user number. This would protect against accidental misuse of files.

4.2 The System Prompt

The system prompt is a short string of characters that appears on the console to denote that the operating system is ready to receive a command. Under MP/MII or MP/M-86 the system prompt consists of the user number, the default drive, and the > character. For CP/M systems, the user number does not appear. For example,

```
1B>
```

indicates that the current user number at the console is 1, the default drive at the console is B, an MP/M system is running, and that it is ready to receive a command. Each console on the system receives a system prompt with the user number set at start-up to the console number. The user number may be changed by the operator at each console to a value between 0 and 15, for example:

```
1B> USER 3
3B>
```

The default drive, 'B' in the above example, will be used if no drive is specified in the command.

Here is another example of a user issuing a command. Note that throughout this document, text typed by the user is shown in bold face and text issued by the CR8 is in normal type script. On the VDU itself, however, they will appear the same. The system will look for the file PIP.CMD on drive B. The default drive may be changed by entering the required drive letter followed by a colon.

```
3B> E:
3E>
```

4.3 The Command Line

The user initiates programs and controls system facilities by entering commands at a console. Only one command may be entered on a line. A command line consists of a keyword, a command tail (optional), and a carriage return character (the RETURN key).

The keyword identifies the command to be executed: either a built-in command or the name of a program file on disk. Note that built-in commands are those that reside permanently in memory; they are faster to use than other commands. The optional command tail contains additional information required by the command.

```
3E> DIR *.BAK
```

In this example "DIR" is the command name, "*.BAK" is the command tail.

An unrecognised command will be echoed on the display, followed by a "?".

```
3E> DTR
DTR?
3E>
```

Commands may be entered in either upper or lower case, or in a mixture of both.

The command line processor (part of CP/M or MP/M) converts all letters internally into upper case, before performing any file searches, etc.
Thus:

```
    3E> dir testprog.bak
```

is equivalent to

```
    3E> DIR TESTPROG.BAK
```

4.4 Editing the Command Line

Typing errors in the command line may be corrected before the RETURN key is pressed by using the control characters given in the table below. Control characters are entered by holding down the CTRL key and pressing the required letter. For example, CTRL Z is entered by holding down the CTRL key and pressing Z. The most useful editing control character is available on a single key, the BACKSPACE key. Pressing this key deletes the last character entered. (The sequence "cursor left, space, cursor left" is sent to the display.) The most useful keys are outlined below:

KEYSTROKE	ACTION
BACKSPACE	deletes previous character, moves cursor back one space.
RETURN	command terminator, sends command to system
CTRL X	cancel line, cursor moves to the start of current line

COMMAND LINE EDITING
Note also that on some terminals (including the Integrated Workstation), holding down a key for a second or two causes the key to repeat. Thus if one had typed in a long command line then noticed an error half way along the line, one could hold down 'BACKSPACE' until the offending character(s) had disappeared. Then only the latter half of the line would need to be re-typed.

4.5 File Specification

A "file Specification" is that part of a command line which specifies the file or group of files to be used in the command. The file specification has three parts in CP/M, 4 parts in MP/M. Some fields are optional.

<d:> <filename> <.filetype> <;password>	MP/M-86 MP/M II
<d:> <filename> <.filetype>	CP/M-86 CP/M 2.2

<d:> is the drive specification, a single letter from A - P followed by a colon. If this field is omitted the default drive will be used instead. The current default drive is shown in the system prompt.

<filename> is a 1 to 8 character file name. The following characters are used by the command processor as separators, so must not be used in a file name (or file type):

< > . , ; : = ? * & ! ( ) / $ Æ Ø Å æ ø å [ ] \ { }
tab space carriage return

Any of the printable characters, other than the separators above, may be used in a file name; however, it is recommended that only letters and numbers are used.

<.filetype> is an optional 1 to 3 character file type. The file type (if used) is separated from the file name by a dot.

Example: NEWFILE1. TEM

The file type is useful for dividing files into groups, depending on their nature or use. In fact, in some cases the file type must not be specified, but is added to the file name by the command processor or excecuting program. E.g.:

3E> **PIP**

MP/M-86 looks for the file PIP. CMD on drive E

Some utility programs create files with the input file name but a different file type, to indicate the nature or stage of processing of the file.

3E> **ASM86 TESTPROG**

In this example, the MP/M - 86 assembler ASM86.CMD looks for a source file TESTPROG.A86 and creates three files: - TESTPROG.H86 containing the assembled program in hexadecimal format, TESTPROG.LST containing the listing, and TESTPROG.SYM containing the program symbols.

The <; password> parameter is available under MP/M II or MP/M-86. The password can be up to 8 characters long, and is separated from the file name and file type by a semicolon.

```
3E> DIR TESTPROG.BAK; STOCK
```

Passwords are assigned to files by using the SET command, described in a subsequent section. Any file may have a password assigned to it, including command files. This means that a command line may require two or more passwords, one for the command file itself and passwords for file specifications in the command tail.

```
3E> DIR; APOLLO TESTPROG.BAK; STOCK
```

The utility program DIR.CMD has been assigned the password APOLLO, and the file TESTPROG.BAK has the password STOCK.

Ambiguous File Specificatons

Some utilities and programs accept ambiguous file specifications, with the "wildcard" characters (* and ?) being used in the <filename> and <filetype>. This allows a range of files to be specified to a program. For example,

```
3E> DIR *.BAK
```

lists the names of all files on drive E with file type .BAK (within user area 3).

The ? character is used to match any single character in that position; a * will match any number of characters from that position. A * is converted internally into a number of ?'s, as many as are required to fill the file name or file type. For example, AB*.* is treated as AB??????.??? . The * must be the last character in the file name or file type. For instance, A*B.BAK is not valid because it is treated as A???????B.BAK which has one too many characters (only 8 being allowed in the file name).

In this case, the user should not use a * but should type in the correct number of ?'s (six - A??????B.BAK).

The following examples, using the DIR command in MP/M - 86, will illustrate the use of the wildcard characters.

```
3E> DIR
Directory for User 3:
E:   A    CMD   :  AA  CMD  :   LETTER1   TXT  :     AAA    CMD
E:   A    A86   :  B   A86  :   B         CMD

3E> DIR * . CMD
Directory for User 3
E:   A    CMD   :  AA  CMD  :   AAA       CMD :      B      CMD

3E> DIR ? . CMD
Directory for User 3
E:   A    CMD   :  B   CMD

3E> DIR ? . *
Directory for User 3
E:   A    A86   :  B   A86  :   B         CMD

3E> DIR A? . CMD
Directory for User 3
E:   A    CMD   :  AA  CMD

3E> DIR A*. CMD
Directory for User 3
E:   A    CMD   :  AA  CMD  :   AAA       CMD
```

4.6 Some System Commands and Utilities

DIR. This displays a list of the files on a given drive.

STAT. Displays the amount of disk space available, disk status, etc. It can also be used to change disk status, for example setting a disk drive to Read Only.

SDIR. This is similar to DIR but the file names are sorted into alphabetical order. It can also give additional information such as the number of records in each file. Available with MP/M only.

USER. This changes the user number. The user number is a means of separating files belonging to different people or applications.

DSKRESET. This must be used before changing a floppy diskette (MP/M only). It checks that no files are open.

SHOW. Similar to STAT; provides additional information about disk directory labels (timestamping, password protection, etc.) Shows which users have files open. (MP/M only).

SET. Sets time stamping, password protection and file attributes. (MP/M only).

MPMSTAT. Provides a long display of many system parameters such as the current states of different processes, etc. This is useful for system programmers and the like. (MP/M only).

ATTACH. This is used to connect a process to the console that was previously running independently (as one of several programs in the MP/M multiprogramming environment.)

ABORT. Stops a program. Can be used to stop a program running on another console. (MP/M only)

TYPE. Displays a text file on the terminal.

ERA. Deletes a file or files in order to release the disk space used.

REN. Renames a file (without copying it). Can be used to set up password protection in MP/M systems.

PRINTER. Displays or sets the printer to be used by a particular console. (MP/M only)

SPOOL. Queues a file or files to be printed automatically when the printer is available. (MP/M only)

STOPSPLR. Stops printing from spool queue and deletes the queue (MP/M only)

CTRL-P. By holding down CONTROL while pressing P, from then on all the text appearing on the terminal will also be printed. By pressing it again, text will no longer be printed.

SUBMIT. This excecutes several command lines in a text file, in order to reduce typing when a sequence of commands is used often. It includes a parameter passing capability, so the commands need not be identical each time the SUBMIT file is used.

PIP. A copying utility. It will copy a file, a part of a file, or several files from one disk to another or to some other peripheral such as a printer.

ED. A simple editor for entering text into a file or altering existing text.

ASM and ASM-86 Assemblers for Z80 and 8088 processors respectively.

COPYDISK. Copies a complete disk, sector by sector. Quicker than PIP but does not reorganize records into sequential order as PIP does. (CP/M-86 only)

DDT and DDT-86. A program-debugging aid that enables a user's program to be stepped though, breakpoints or trace points to be read, memory to be read or written, and so on.

DUMP. This displays the contents of a disk file in hexadecimal format.

GENCMD. Converts object code produced by assemblers or compilers into a form that may be executed as a command. (MP/M-86 and CP/M-86 only).

HELP Provides information at the console about the use of these utilities (CPM-86 only).

4.7 SPECIAL CR8 COMMANDS

The CR8 is furnished with certain utilities in addition to those provided by CP/M or MP/M. These are described in detail below. (The disk formatting program and FCOPY floppy disk copying programs also come into this category but were described in section 3.)

Setting Baud Rates

Three programs are supplied for setting the Baud rate to be used on the various line interfaces. LPBAUD is used for the line printer interface provided by the MP^2 card. CONBAUD sets the other interface provided by the MP^2 - the interface to the console. In multi-user systems, use SCIBAUD to set the Baud rate on the extra interfaces supplied by the SCI board.

The default Baud rate for all interfaces is normally 9600.

Having decided which of these three programs to use, then (assuming the program is on your default disk) type its name. All 3 are menu-driven, that is they will tell you what to type.

Switching between Danish & English Characters

Pressing CTRL-N will effect this, CTRL-O will return the terminal to English mode. Further details appear in section 3.2.2.

System Generation - Floppy-only System

The SYSGENF program is available for writing to the system tracks of the system diskette. The reason that a special program is needed is that PIP only references files within the normal directory system. The file containing the operating system resides on two reserved tracks of the disk and does not fall within the normal file management system.

SYSGENF will read or write the entire system at a time. The sequence of events would be to read in the system from one drive to starting address 3480 (hex) in memory. Then the system would be patched, if required, in accordance with Digital Research's documentation. For example, the default drive could be changed or an initial command to be executed could be put into the command-buffer. Finally the system (in original or patched form) would be written back to disk - possibly to a different drive.

A typical application of the program would be to put an operating system onto a diskette that had just been formatted.

The command syntax for SYSGENF is given below:

The program displays a menu offering:

<u>G</u>etting information from system tracks
<u>P</u>utting information on system tracks
<u>M</u>emory patching the operating system
<u>^C</u> - press CTRL-C to exit.

System Generation - Winchester/Floppy Systems

Any CR8 with Winchester and floppy disk drives is supllied with a diskette known as the **Installation Floppy.** This can be used to re-create the operating system on the Winchester if it is lost after some accident such as switching off the system while the Winchester's LED is illuminated.

To use the installation Floppy, boot from it in the normal way. A menu of options will be displayed. One of these options (which on some Installation Floppies does not appear on the menu yet is nevertheless available) is the **Toolbox** utility. Press **T** to enter this program. The Toolbox itself has its own menu of system utilities. One of these is a system generation program for patching or copying the system tracks on the Winchester.

N.B. The Toolbox should not be used by people who are not familiar with the operating system - the operating system could easily be rendered useless by misuse of some of the Toolbox's powerful utilities.

Further details on the use of the Installation Floppy are included in the text file SYSDISK:DOC which is included on that diskette.

4.8 Disk Access Error Codes

Floppy Disk

CODE:	DESCRIPTION:
08	**Checksum error (data)** One or more data bytes have not been read correctly from disk - Press <return> twice to ignore (the user may want to correct the error himself). Press CTRL-C to abort reading.
10	**Checksum error (sector heading)** The sector cannot be found due to reading error in sector header. Press <return> twice to ignore (the entire sector will be lost - 128 characters). Press CTRL-C to abort reading.

CODE:	DESCRIPTION:
40	**Write protected disk** Remove the cover from the write-permit notch on the diskette (see pg. 3-8). Then press CTRL-C, before re-trying the disk write.
FF	**Controller timeout error** This can be caused by the lid on the floppy drive being left open.

Winchester Disk

CODE:	DESCRIPTION:
02	**Drive Not Ready** Indicates that the requested drive is not ready or non-existent.
03	**Seek Timeout** The drive has failed to complete a seek operation within 2 seconds.
04	**Invalid Track 00 Indication** The drive has given the controller an unexpected track 00 indication. Generally indicates a drive problem.
05	**All ID Fields Bad** The controller is unable to read any ID field on the requested track. (Probably caused by an unformatted drive).
06	**Record Not Found** The requested record cannot be located on the specified track.
07	**Record Not Found and ID ECC Error** The requested record cannot be located and an ID Field ECC error exists.
08	**Position Error** Indicates that the seek operation completed and the drive is not on the desired cylinder. Recommended recovery is to issue a RESTORE command and retry the operation.

CODE:	DESCRIPTION:
09	**Data Transfer Start Error** The controller is unable to transfer data to or from memory. Generally indicates a hardware controller fault.
0A	**Write Fault Error** The requested drive has a write fault condition. This condition can be cleared only by powering down the drive.
0B	**Index/Sector Timeout** The controller did not receive an index or sector pulse from the requested drive in a reasonable period of time.
0C	**Command Parameter Error** The control module has received an illegal command parameter. This generally indicates a hardware problem in the controller.
0D	**Uncorrectable ECC Error** The specified record contains more than 11 erroneous bits, so it could not be corrected by the error correction circuit. The recommended recovery is to retry the command three times. If the error persists, the data must be considered non-recoverable.
24	**Defective Processor** A diagnostic error code which indicates a problem in the CPU of the controller board.
25	**Defective Buffer Memory** A diagnostic error code which indicates defective memory in the controller board.
26	**Defective EDD Circuitry** A diagnostic error code indicating a defect in the controller's error correction circuits.

CODE:	DESCRIPTION:
27	**Defective Program Memory** A diagnostic error code which indicates a defect in the controller's microprogram memory.
29	**Illegal Interleave Table Parameter** Indicates a problem in the user-defined interleave table. This can only occur during the execution of a 'set interleave' command.
30	**Illegal Cylinder Address** Indicates that either the controller has received a command containing a cylinder number larger than the last cylinder on the disk, or, that during a multi-record transfer, the record count causes the cylinder address to increment past the last cylinder address on the disk.
31	**Illegal Head Address** The controller has received a command which specifies a head which does not exist on the disk.
32	**Illegal Record Address** The controller has received a command which specifies a record address greater than the amximum record number (Record 17 for 512 byte format, record 32 for 256 byte format).

5. TECHNICAL OVERVIEW

This section provides the operational parameters for every component module of the CR8 microcomputer. It is intended to provide sufficient detail for users who wish to add their own systems software or incorporate additional hardware into the CR8.

5.1 THE CABINET

This section deals with the cabinet, Multibus card cage, fans, power supply, etc. for the cabinet model CR8. The user must supply (or CR can supply) a separate terminal to use as a console with this CR8 configuration. The cabinet is available as desk-top model or in a form ready for mounting in a standard 19" rack. Furthermore, the cabinet is available in three sizes which are 2, 4, or 6 standard Units high (One Unit = 4.425 cm). The differences are described in the following table:

CR8 Model No.	No. of Units	No. of Multibus Slots	Space for Peripherals
CR8/007/X	2	5	No
	4	5	Yes
	6	10	Yes

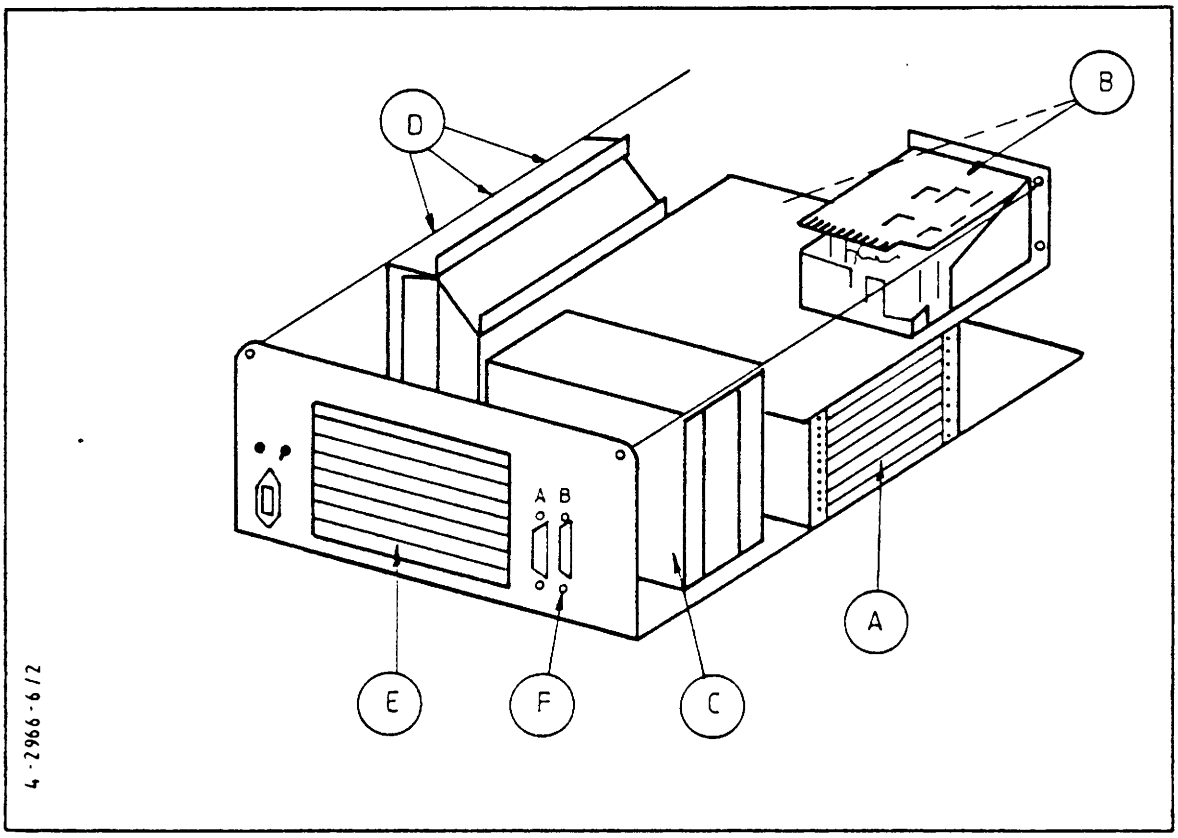
Note that the number of unused Multibus slots will be lower than the figures given, depending on the options chosen. A Winchester disk requires a controller board which uses one slot (because it is very thick - however it only uses one slot in the Workstation models since there is space behind the last slot for a thick board). An SCI board uses one slot (Multi-user options). All models include an MP^2 board which occupies one slot. The illustrations in this section refer to the 4-Unit model.

The "Space for Peripherals" part of the table above indicates whether or not disk drives are supplied with that model - there is no space for user-supplied disk drives, etc.

5.1.1 PHYSICAL LAYOUT

Access to the interior is obtained by loosening two screws and removing the front cover. The top cover can also be removed if necessary.

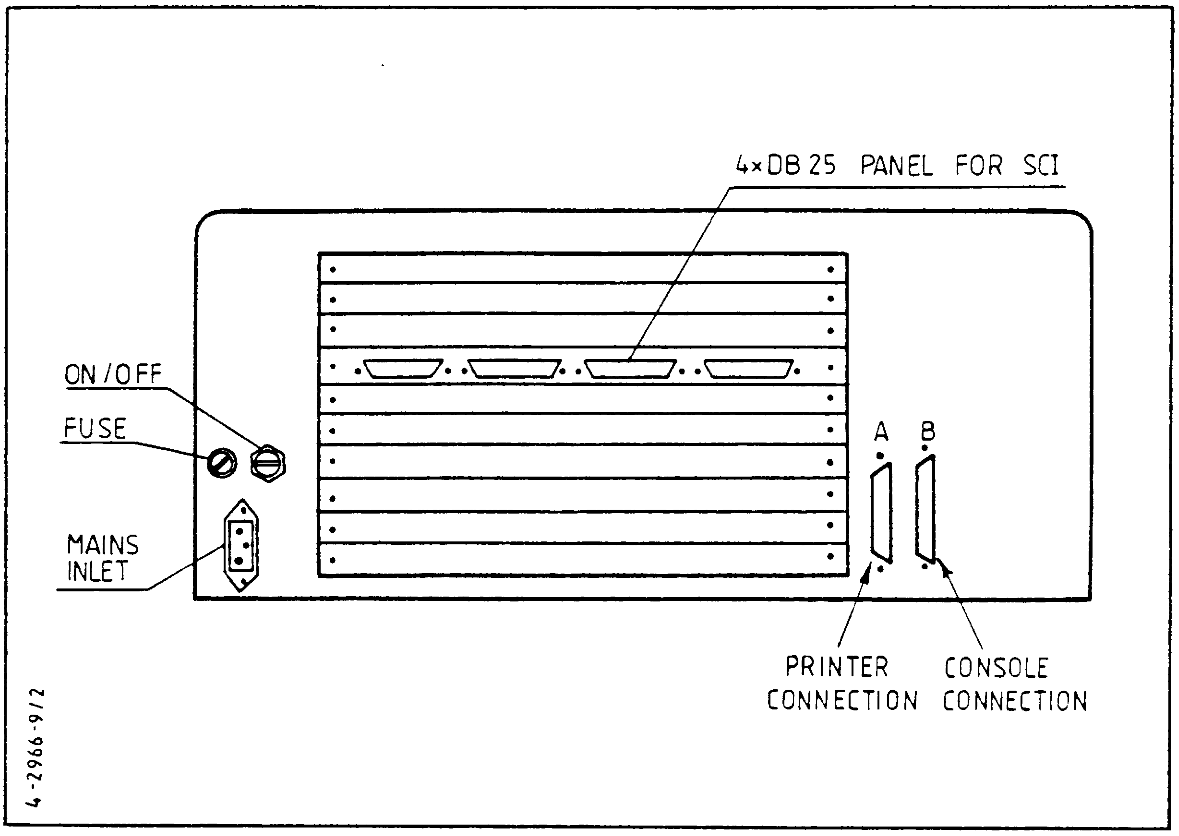
CAUTION:
Before opening be sure that the power is off, and that the power cord is detached.



CR8 CABINET INTERNAL VIEW

Key to diagram above:

A:	Multibus card cage
B:	Positions for disk drives
C:	Power supply
D:	Fan Units
E:	Connector panels (I/O)
F:	Operators console



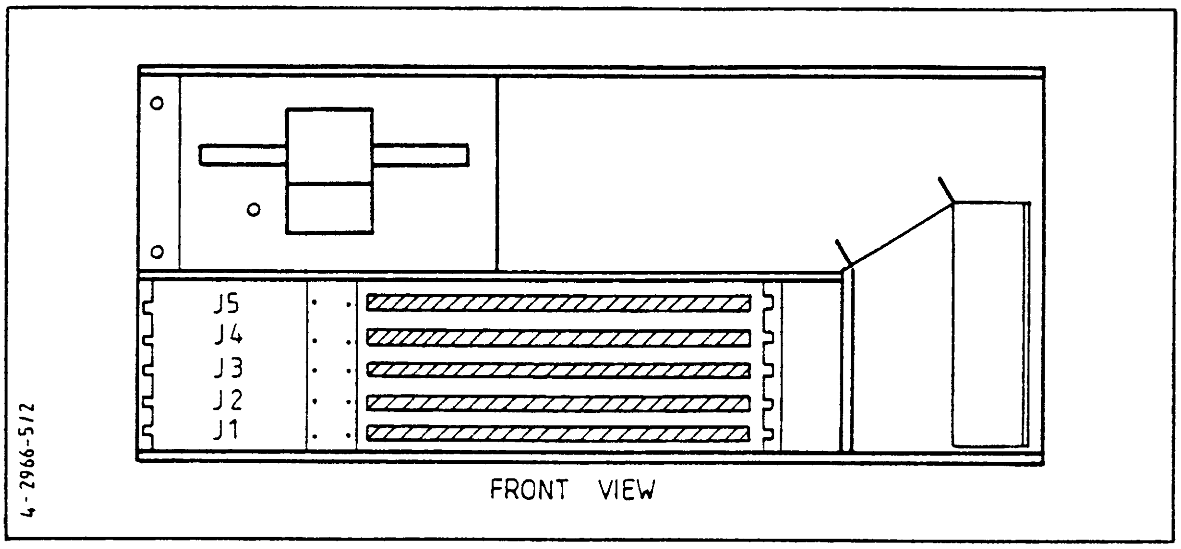
CR8 REAR CONNECTIONS

5.1.2 INSTALLATION OF MULTIBUS MODULES

Before the Multibus modules are slid into the card cage, a number of jumpers on both motherboard and modules have to be positioned.

The 5 slots are numbered as indicated below, and the standard module positions are as follows:

J5:	Cannot be used in CR8s with Winchester (Dec.82) due to thickness of Winchester controller board.
J4:	Winchester controller
J3:	MP^2
J2:	RAM Board
J1:	SCI



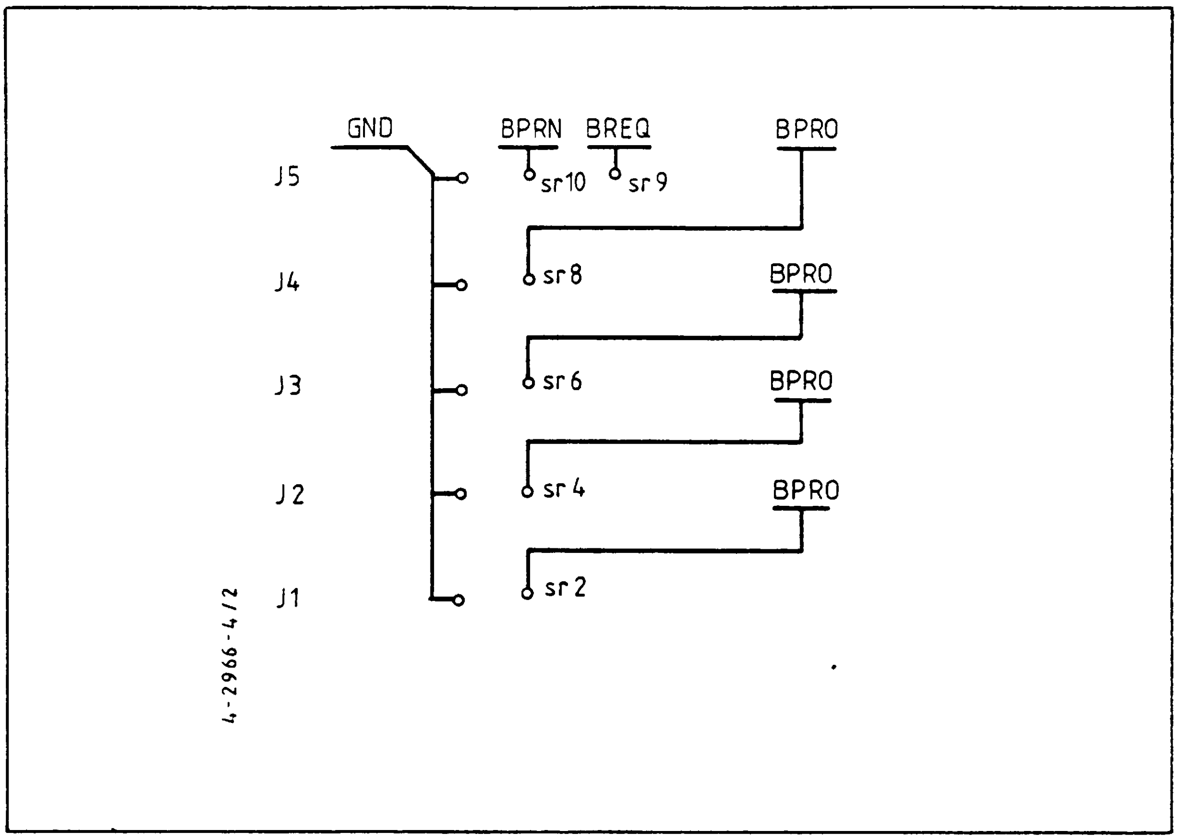
SLOT NUMBERING IN CR8 CABINET MODEL

5.1.3 THE CABINET MODEL MOTHER BOARD

This board is situated vertically in the cabinet. It houses the connectors for the Multibus boards to plug into. It has no logic circuitry; only the connectors and some jumpers which are described below.

JUMPER NO.	MOUNTED	NOT MOUNTED
1		X
2		X
3		X
4		X
5		X
6		X
7		X
8	X	
9		X
10		X

JUMPER LIST - MOTHERBOARD



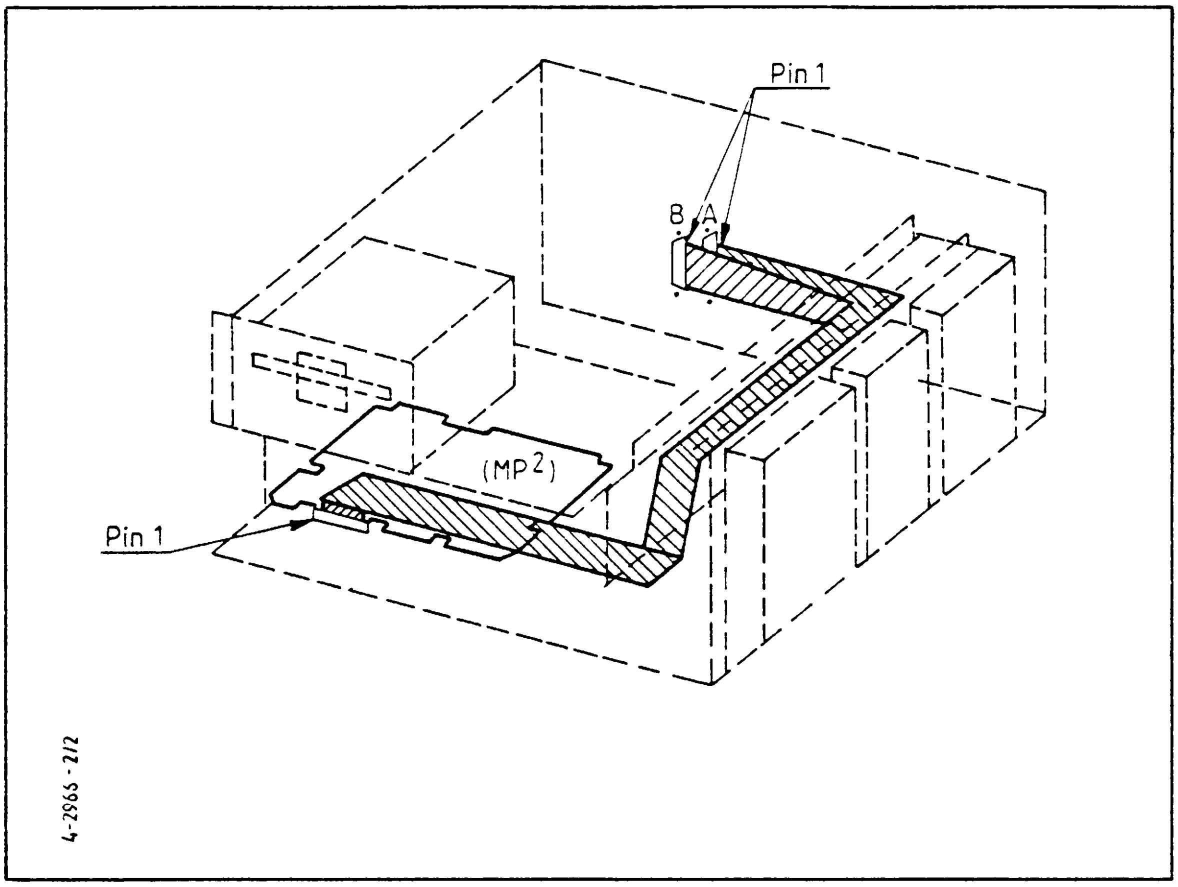
JUMPER MAP - MOTHERBOARD

The pins sr1 to sr6 in the above diagram serve the following purposes:

sr1:	Not used by CR
sr2:	Not used by CR
sr3:	Not used by CR
sr4:	Not used by CR
sr5:	Not used by CR.
sr6:	Not used by CR.
sr7:	Not used by CR.
sr8:	Bus priority out (Winchester) and Bus priority in (MP^2).
sr9:	Not used by CR.
sr10:	Not used by CR.

5.1.4 CABLING (CABINET TO MP^2)

When the MP^2 board is in position, a dual V.24 cable is installed as shown below.



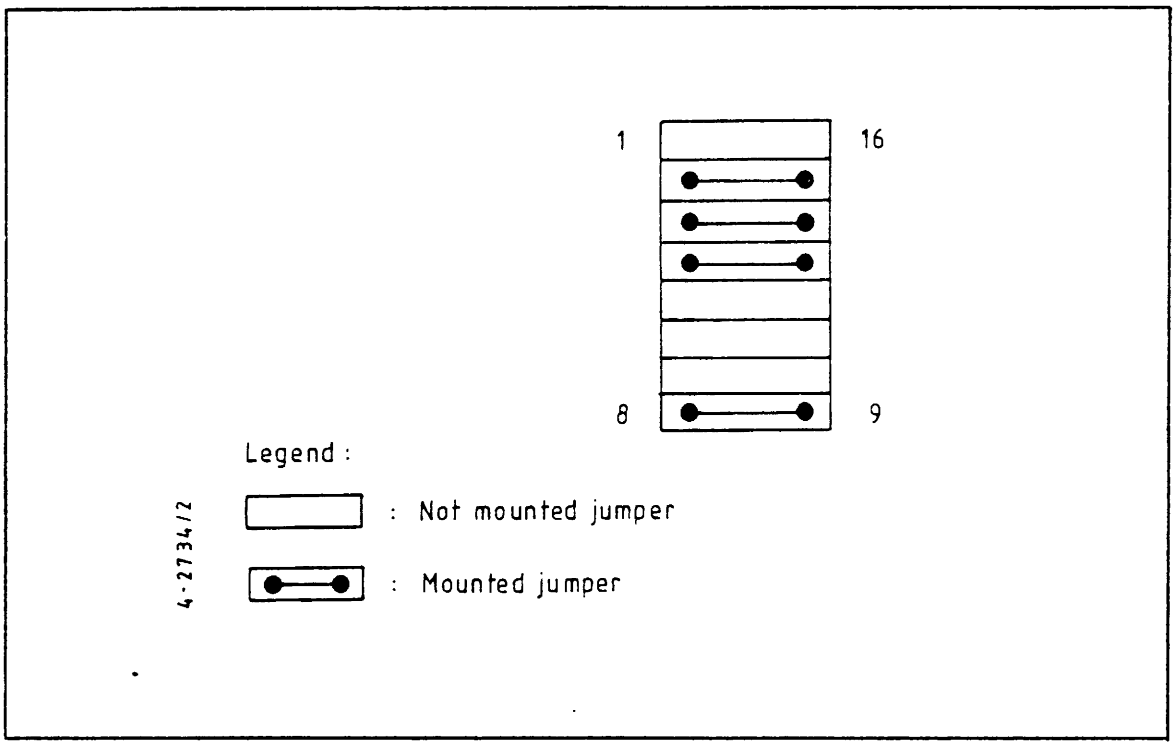
V.24 CABLE

5.1.5 INSTALLATION OF DISK DRIVES

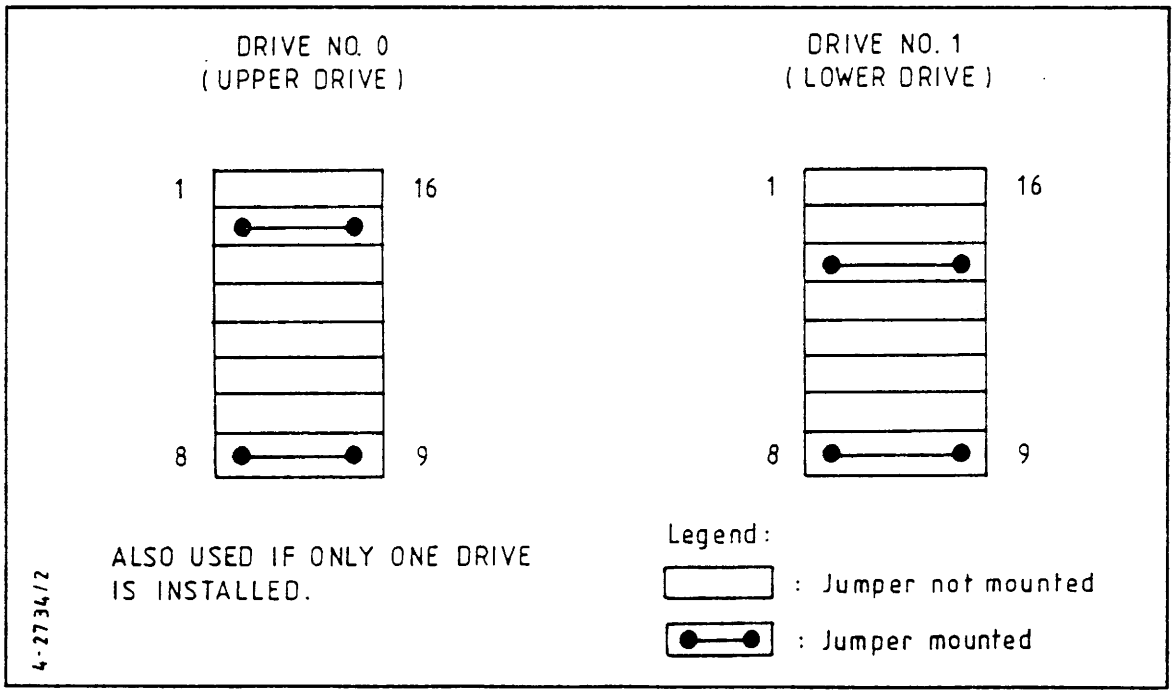
When the top cover is removed, the one or two disk drives can be installed directly, without further disassembly. Each drive is secured by 3 (leave one hole empty) UNC 6-32 x 0.25" screws, which fit in the side of the drive.

The left position is used when only one drive is installed.

5.1.5.1 STANDARD JUMPER SETTING (DISK DRIVES)

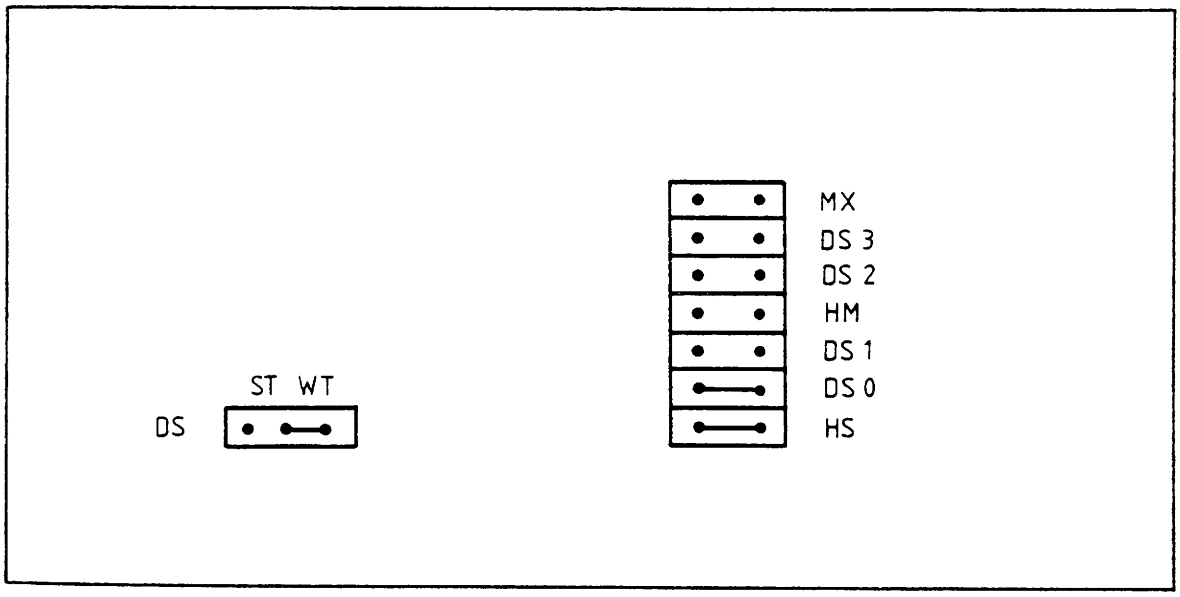


JUMPER ARRAY - WINCHESTER DRIVE



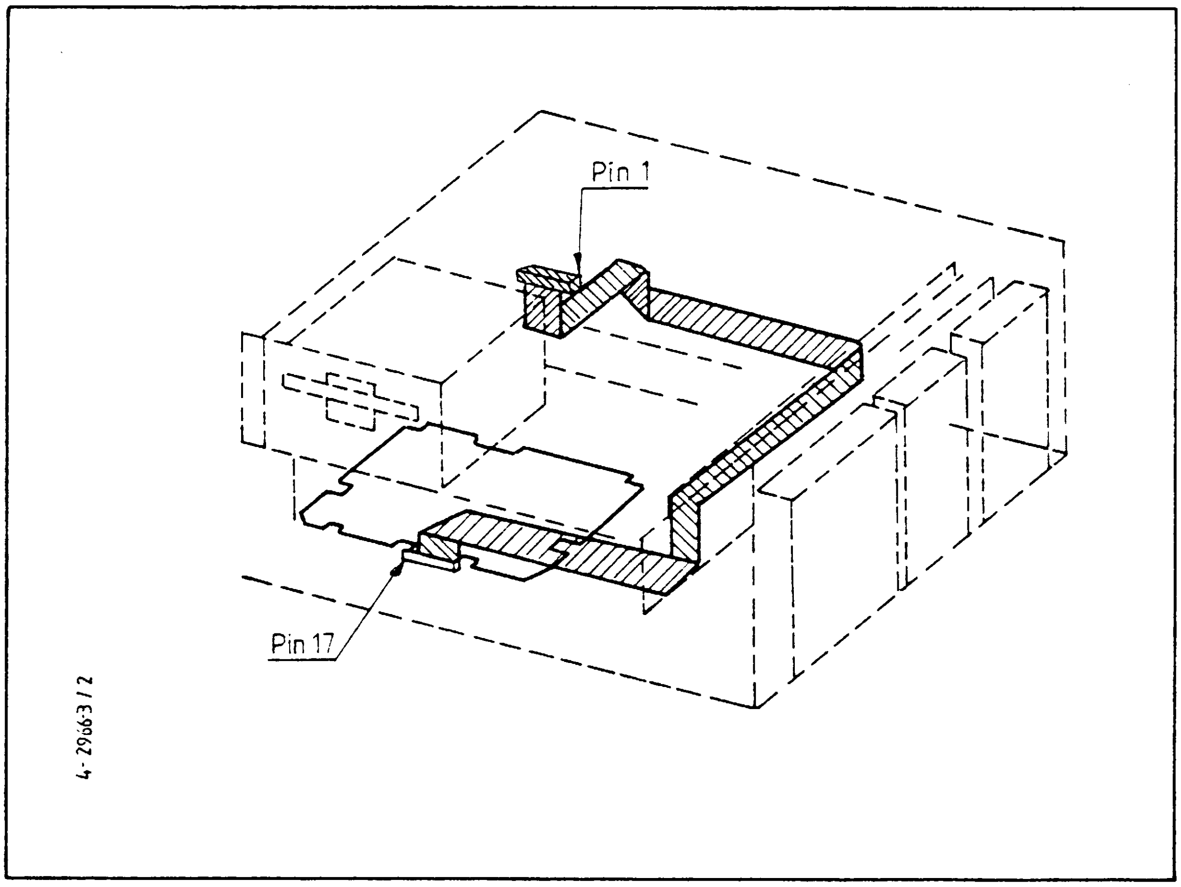
JUMPER ARRAY - FLOPPY DRIVE

The DIP resistor array 2F is removed from floppy drive no. 0, when two drives are installed.

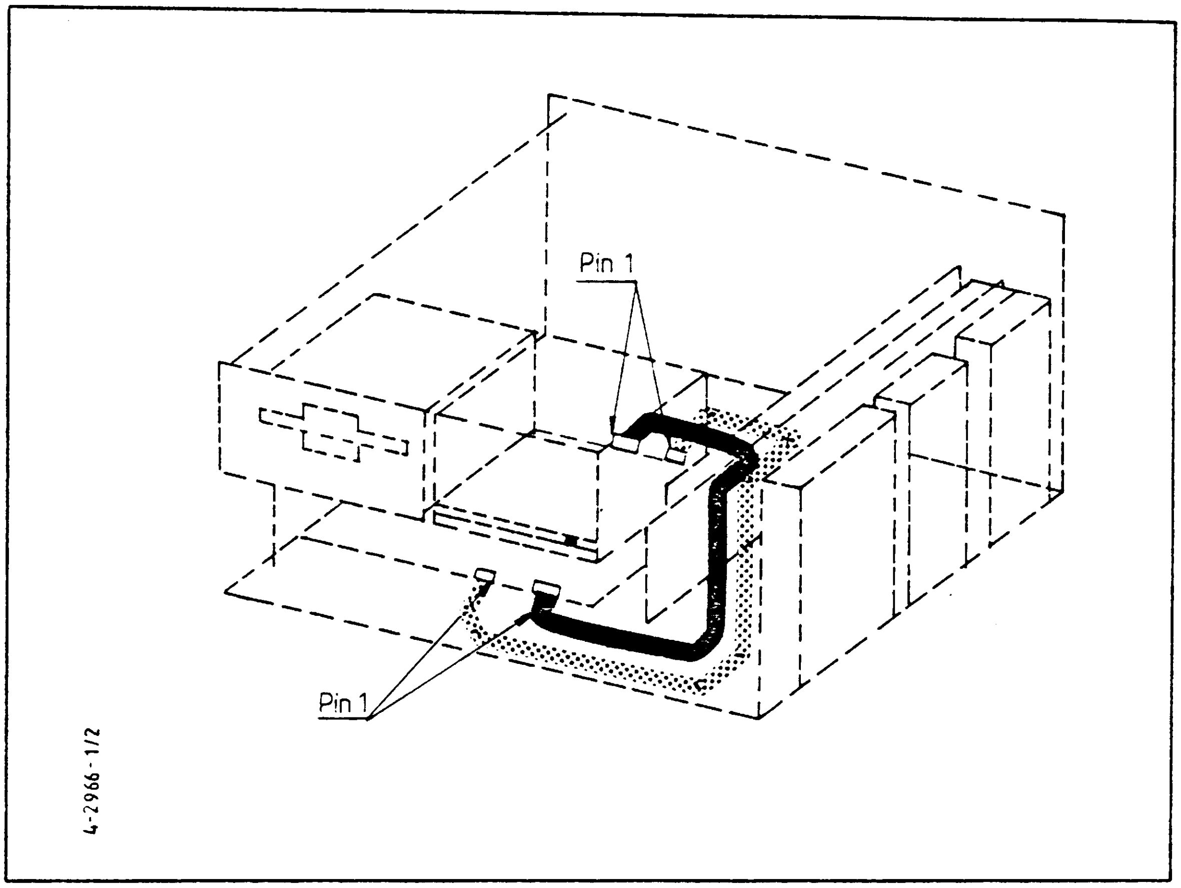


JUMPER ARRAY - FLOPPY DRIVE (TEAC FD-55F)

5.1.5.2 CABLING (DISK DRIVES)



FLOPPY CABLE ROUTING



WINCHESTER CABLE ROUTING

The drives should furthermore be connected to the yellow/green ground wire and the power wire assembly (yellow, red, black, black).

5.1.6 POWER SUPPLY

The power supply provided with the early CR8 Cabinet is of the Philips PE 1203 series type. Newer cabinets are fitted with a power supply of CR's own manufacture, which provides an output of 150 watts.

The CR8 equipment can be used in both 220VAC/50Hz and 110 VAC/60Hz power installations, provided the following instructions are observed:

- o The power supply should be adjusted according to the voltage
- o The two fans should match the voltage
- o The mains fuse should match the voltage.

The specifications of the CR power supply are given below:

Input Specification

Mains Voltage	: 220V
	: 110V - selectable
Mains Frequency	: 45 - 440 Hz
Inrush Current	: Max. 40A at Cold turn-on

Output Specification

Output Voltage	: +5.1V 2% at 10% - 100% load
	: +12V 5% at 10% - 100% load
	: -12V 5% at 0% - 100% load
Output Current (continuous)	: +5.1V / 20A
	: +12V / 6A
	: -12V / 1.5A

Output Current (peak)	: +5.1V / 30A
	: +12V / 12A
	: -12V / 1.5A
Output Power	: Max. 150W
Load Regulation	: Typical 0.5%
Line Regulation	: Better than 0.1%
Transient Response	: Within regulation in less than 0.5 msec. for a 50% - 100% - 50% load change mV peak to peak over 0 to 30 MHz..
Noise	: +5.1V: <150 mV peak-to-peak over 0 to 30 MHz
	: +12V : <300 mV peak-to-peak over 0 to 30 MHz
	: -12V : <300 mV peak-to-peak over 0 to 30 MHz
Temperature Range	: 0^{o}C - 55^{o}C
Temperature stability	: 250 ppm/^{o}C
Energy Storage	: Typically 30 msec from last peak of line voltage. (Nominal input voltage and full load).
Overvoltage protection	: 6.5V $\pm$8% on 5.1V.
Current limit	: All output are current limited and will automatically restart.

5.2 THE WORKSTATION

This Section deals with those parts of the CR8 Integrated Workstation which one would expect to find in a terminal, i.e. everything but the processor board and the disk drives.

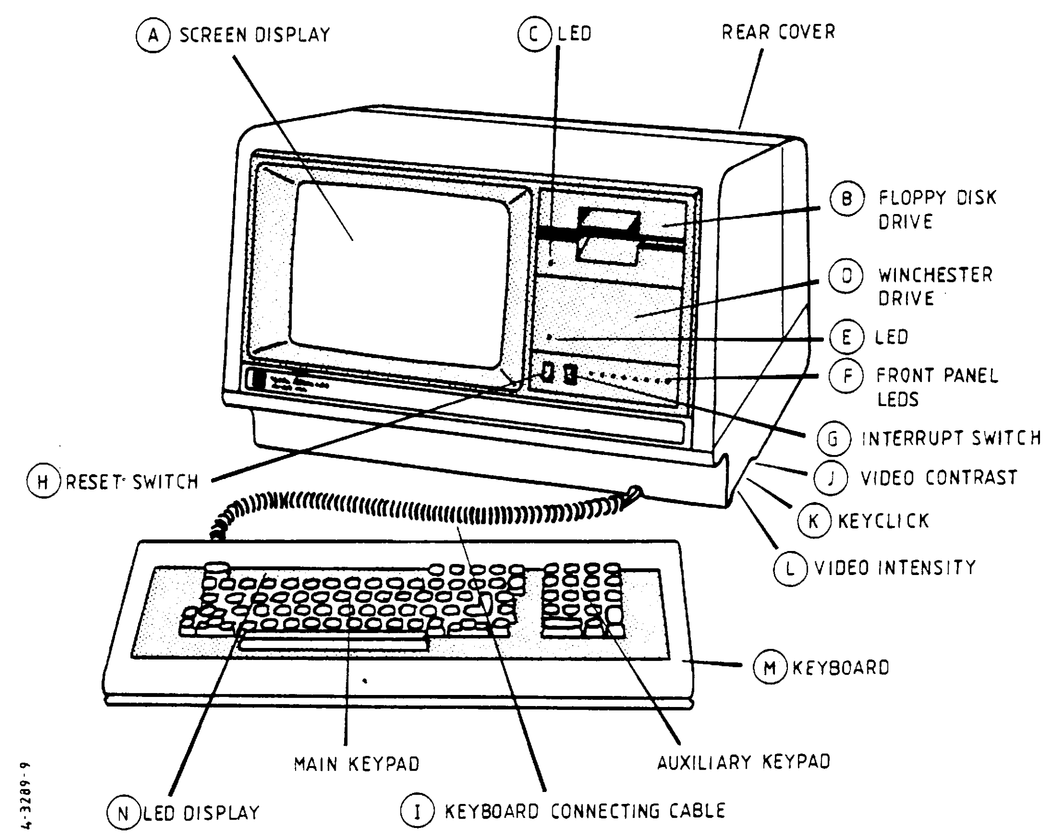
5.2.1 FEATURES

The terminal forms the basis of the CR8 integrated workstation models. It includes a 12" CRT display, a Multibus-compatible card cage and a high-capacity switching power supply. The unit has many advanced features, including the following:

- Split screen capability with independent scrolling of a part or the whole each screen section. The user can also define the transition point that separates the screen sections.
- Smooth scrolling up or down
- Cursor addressing - cursor can be moved anywhere and the character it refers to may be read or written.
- Tabulation commands
- Random line erasure
- Line and character edit
- Software control of LEDs
- Alternate character set supplied by user (in ROM)
- Character attributes to apply to specific character positions: underline, blink, overstrike, reserve video, dim
- 106 displayable characters including 11 form-drawing graphics characters.
- Mode to display control characters instead of obeying them
- Two keyboard modes - numeric keypad keys are special function keys in one mode or just numeric keys in the other mode.

5.2.2 TERMINAL COMPONENTS AND CONTROLS

In this section the various components of the terminal are identified and brief details given.



THE CR8 WORKSTATION - FRONT VIEW

Features highlighted in this drawing are described below.

(A) **CRT display**

The 12" CRT display is connected internally to the processor board using a 9600 baud V24/V28 link. The unit displays 80 characters on 25 lines in upper and lower case. Video attributes include normal and faint intensity, blink, underline, insert/delete and smooth scrolling.

(B) **Floppy disk drive**

This is a 5 1/4" dual side, dual density, floppy disk drive, capacity 1 megabyte unformatted, 608 Kbytes formatted. (Specification may be improved from time to time.)

(C) **LED**

This LED is lit when the floppy disk drive is being used by the system.

(D) **Winchester disk drive**

A mini-Winchester rigid disk drive, with a capacity of 5 or 10 megabytes (formatted). (Specification may be improved from time to time.)

(E) **LED**

This LED is lit when the Winchester disk drive is being used by the system.

(F) **Front panel LED's**

A row of 8 LED's which may be used as general purpose displays by various systems on the CR8.

(G) **"Interrupt" switch**

Not used

(H) **"Reset" switch**

A momentary-action switch which causes the Multibus to be reset and initiates the start-up procedure.
Note: The video display circuitry is not reset by this switch. This can only be achieved by switching the unit OFF, then ON.

(I) **Keyboard cable**

A telephone-style cable connects the keyboard to the main section of the terminal. The cable may be unplugged at both ends.

(J) **Video contrast**

Controls the overall intensity of the display ("white level").

(K) **Keyclick enable**

This turns keyclick on or off. When rotated towards the operator, a short click or squeak is produced whenever a key is depressed on the keyboard.

(L) **Video intensity**

Controls the "black level" of the display. This should be set so that the dark parts of the screen only just fail to glow; afterwards control (J) is set as desired.

(M) **Keyboard unit**

The detachable keyboard has 82 typewriter-like keys, including cursor control keys, functions keys, and a separate auxiliary keypad.
The keypad includes the digits 0-9, period, comma, minus and enter. When in numeric mode (default or power-up mode) these keys transmit the usual ASCII characters. However, when the terminal is in APP mode (application mode), these keys transmit unique escape sequences. They can then be used as an additional set of 14 function keys. Note that the cursor control keys and the HOME key on the main keyboard also send unique escape sequences in APP mode.

The auxiliary keypad also contains four function keys, PF1-FP4. These keys are not affected by APP mode.

The main keyboard includes a display of eight LED's. Three, L0 to L2, are software controllable, the remaining five display the terminal status, as described below.

The smooth scrolling feature is not software controlled, but is enabled/-disabled by successive depressions of the SLOW SCROLL key on the main keyboard. If smooth scrolling is disabled, data received at the terminal will cause the display to scroll upwards discountinously. If smooth scrolling is enabled, lines of data received at the terminal will move slowly and continuously up the screen, and will therefore be easier to read.

An additional feature of the keyboard is that the auxiliary keypad may be put into APP mode (program application mode). In this mode, the numeric keypad and a number of other keys generate unique character sequences instead of their usual ASCII codes. It is then possible for an application program to use the keypad as an extra set of functions keys.

The keyboard is available in Danish and English versions (see section 3.7). The English one is illustrated overleaf.

(N) **LED Display (keyboard)**

On line LED - This light is on when the Local/On-line switch on the rear panel is in the ONLINE position. In this mode, characters typed on the keyboard are transmitted to the processor. Note that the video display will only be affected when the processor transmits the characters back to the terminal.

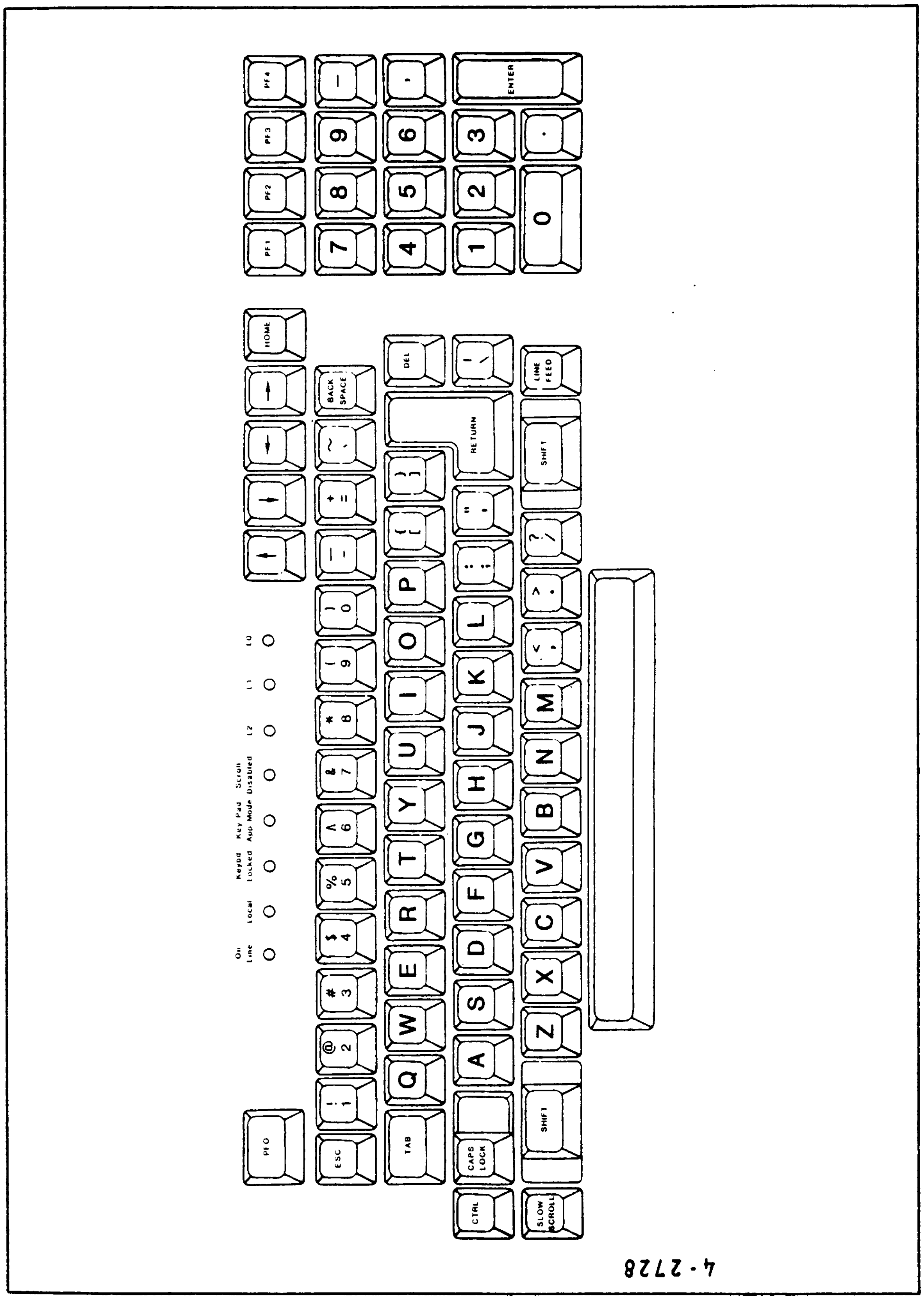
Local LED - This light is on when the Local/On-line switch on the rear panel is in the LOCAL position. In this mode, characters typed at the keyboard are "looped back" to the terminal, as though they were received from the processor.

Keyboard locked LED - The keyboard Locked light indicates that the keyboard has been deactivated, either by the processor sending the appropriate escape sequence, or by the escape sequence being entered at the keyboard while in the LOCAL mode.

In on-line mode the processor can unlock the keyboard by sending an appropriate escape sequence. In local mode, the terminal cannot be unlocked unless it is repowered.

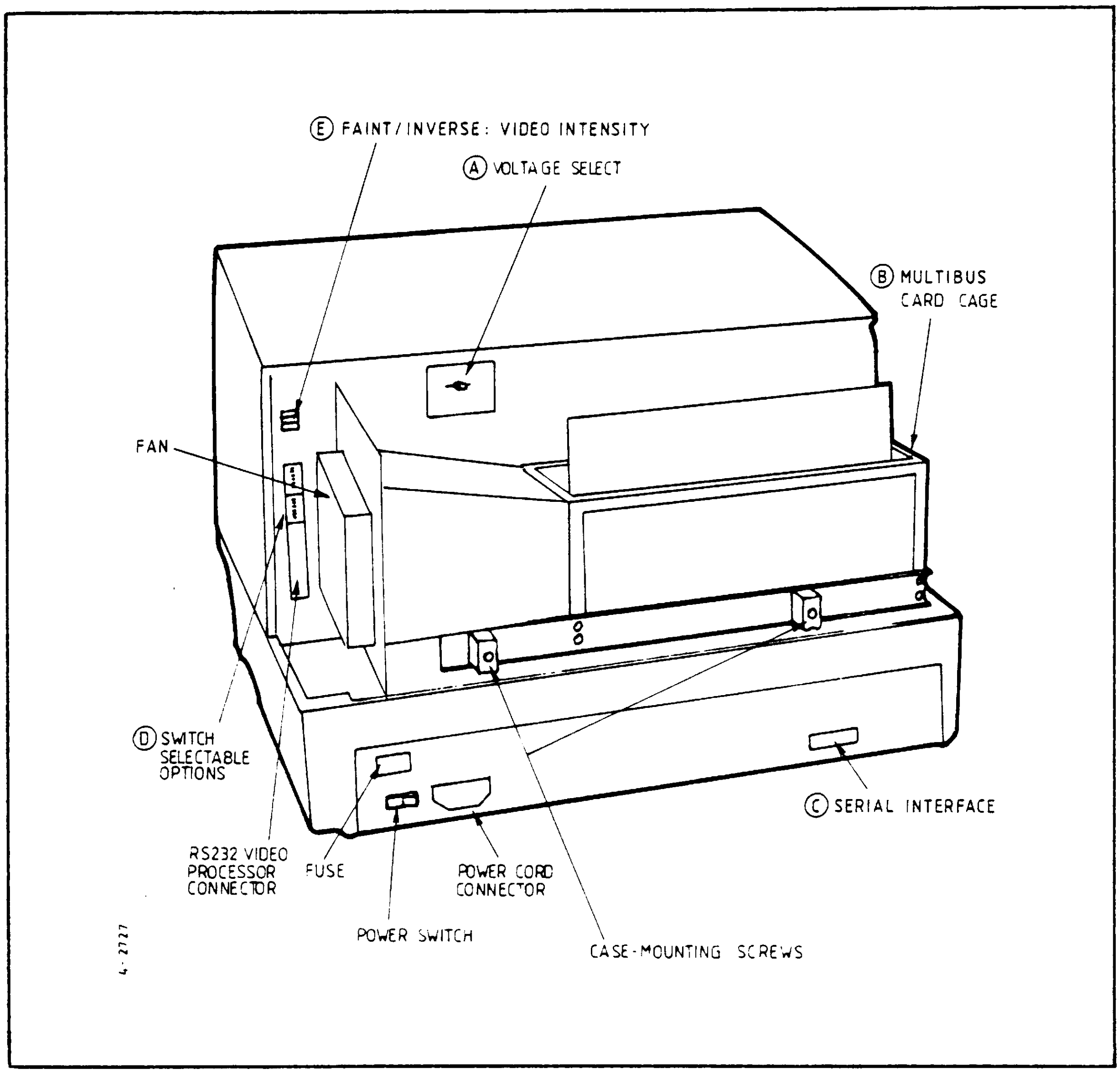
APP mode LED - This light is on when the keypad is in Application Mode. This means that the auxiliary keypad and the cursor control keys generate different characters from normal (see section 5.2.3.2).

Scroll disabled LED - this LED indicates that scrolling has been suspended by the user entering Control S (XOFF) at the keyboard. Scrolling will be resumed when the user enters Control Q (XON).



THE KEYBOARD

For the remaining components and controls, please refer to the illustration below of the rear view of the terminal.



DISPLAY UNIT - REAR VIEW (COVER REMOVED)

The rear cover of the display unit may be removed by removing the two case mounting screws at the rear then tilting the cover as though it were hinged where it joins the rest of the cabinet at the top. This gives access to the terminal configuration switches and the Multibus card-cage.

(A) **Voltage Select**

A 2-position switch, switching between 230V, 50Hz, 2A and 115V, 50Hz, 4A. Note: the fuse should be changed if the selected voltage is changed (fuse type 3AG, 4A at 115V, 2A at 230V)

(B) **Multibus card cage**
The card cage holds the MP-square dual-processor card, and has room for additional Multibus-compatible cards, such as SCI (Serial Communications Interface), memory cards, hard disk interface cards.

(C) **Serial interface**
This is an extended communication interface, fully compatible with the CCITT V24/V28 recommendations for DTE devices.

(D) **Switch selectable options**
These switches, arranged in 2 banks of 8 each, modify some characteristics of the video display and processor link. Details are given below (section 5.2.2.1).

(E) **Faint/inverse video intensity**
These two trimmer pots are adjusted at the factory and should not normally be changed. The upper control adjusts the intensity of the faint field attribute, the lower control adjusts the intensity of the inverse field attribute.

5.2.2.1 SWITCH SELECTABLE OPTIONS

The locations of these switches are shown in the illustration in the previous subsection.

SW1 - UPPER SWITCH BANK

		Switch Position	
Number	Name	Left (OFF)	Right (ON)
1-8	Auto Wrap	Enable Auto Wrap	Disable Auto Wrap
1-7	XON/XOFF	Enable XON/XOFF	Disable XON/XOFF
1-6	Auto Linefeed	Enable Auto Linefeed	Disable Auto Linefeed
1-5	Monitor Mode	Enable Monitor Mode	Disable Monitor Mode
1-4	Not used		
1-3	Cursor Blink	Non-Blinking Cursor	Blinking Cursor
1-2	Cursor Shape	Dash Cursor	Block Cursor
1-1	Inverse Video	Inverse Video	Normal Video

Switch 1-8 Auto Wrap

The AUTO WRAP feature defines the terminal's behaviour when a displayable character is received while the cursor is at column 80. If AUTO WRAP is enabled the character will be displayed in column 80 and the terminal will execute an automatic carriage-return/linefeed.

Switch 1-7 XON/ XOFF

The XON/XOFF feature provides a technique for coordinating the incoming data rate with the scroll rate of the display during smooth scroll mode. Because the smooth scroll rate of 3.9 lines/second (50Hz) may require a slower character rate than the incoming character rate, the terminal regulates the host computer's data rate by transmitting the XON/XOFF control characters.

If the XON/XOFF feature is enabled the terminal will transmit the XOFF control character (DC3) when the input buffer is almost full. When the buffer has almost emptied the terminal will transmit the XON control character (DC1).

If the XON/XOFF feature is disabled the terminal will use a different technique to regulate the data rate. The Request-To-Send (RTS) signal on the RS232 interface will be deactivated when the buffer has almost emptied.

Switch 1-6 Auto Linefeed

If the AUTO LINEFEED feature is enabled the terminal will execute an automatic linefeed when a carriage return is received.

Switch 1-5 Monitor Mode

MONITOR MODE determines the terminal's response to the 32 control characters (00H to 1FH) and the delete character (7FH). If MONITOR MODE is disabled (this is normally the case), the control characters implement their control function but are not displayed. If MONITOR MODE is enabled the control function is not implemented but an associated graphic character is displayed (see table below)

HEX	ASCII	MONITOR	HEX	ASCII	MONITOR
00	NUL	.	10	DLE	D_L
01	SOH	S_H	11	DC1	D_1
02	STX	S_X	12	DC2	D_2
03	ETX	E_X	13	DC3	D_3
04	EOT	E_T	14	DC4	D_4
05	ENQ	E_Q	15	NAK	N_K
06	ACK	A_K	16	SYN	S_Y
07	BEL	E_T	17	ETB	E_B
08	BS	B_E	18	CAN	C_N
09	HT	H_T	19	EM	E_M
0A	LF	L_F	1A	SUB	S_B
0B	VT	V_T	1B	ESC	E_C
0C	FF	F_F	1C	FS	F_S
0D	CR	C_R	1D	GS	G_S
0E	SO	S_O	1E	RS	R_S
0F	SI	S_I	1F	US	U_S
			7F	DEL	...

GRAPHIC DISPLAY OF CONTROL CHARACTERS

Switch 1-3 Cursor Blink

The terminal will generate a blinking cursor if this switch is in the right (ON) position.

Switch 1-2 Cursor Shape

If this switch is in the right (ON) position the terminal displays a 'block' cursor; in the left (OFF) position, the terminal displays a 'dash' cursor.

Switch 1-1 Inverse Video

If this switch is in the right (ON) position the display will be of light characters on a dark background. In the left (OFF) position the display will be dark characters on a light background.

SW2 LOWER SWITCH BANK

Number	Name	Switch Position Left (OFF)	Switch Position Right (ON)
2-8	Local/Online	Local	Online Mode
2-7	Data length	7 Data Bits	8 Data Bits
2-6	Parity	Even Parity	Old Parity
2-5	Parity Enable	No Parity	Parity
2-4	Baud rate select code	see table below	
2-3	Baud rate select code		
2-2	Baud rate select code		
2-1	Baud rate select code		

SW2-	4	3	2	1	Baud rate	
	N	N	N	N	9600	
	N	N	N	F	7200	
	N	N	F	N	4800	
	N	N	F	F	3600	
	N	F	N	N	2400	
	N	F	N	F	2000	
	N	F	F	N	1800	
	N	F	F	F	1200	
	F	N	N	N	600	
	F	N	N	F	300	
	F	N	F	N	150	(2 stop bits)
	F	N	F	F	134.5	(2 stop bits)
	F	F	N	N	110	(2 stop bits)
	F	F	N	F	75	(2 stop bits)
	F	F	F	N	50	(2 stop bits)

Select Codes
N = ON (right)
F = OFF (left)

BAUD RATE SELECT CODES

Switch 2-8 Local/Online

This switch determines if the keyboard communicates with the processor board (through the RS232 interface) or internally with the terminal itself.

In ONLINE mode (right (ON) position) characters entered at the keyboard (or generated by the function keybs or cursor control keys) do not have a direct effect on the display. The data generated is transmitted through the RS232 connection to the CPU, where it is processed and sent back to the display.

In LOCAL mode (left (OFF) position) characters generated by the keyboard are sent internally to the display. This mode is useful for diagnostic purposes. Also, if the terminal is in LOCAL mode when power is first turned on, the self-test disagnostic will be initiated.

Switch 2-7 Data Length

If this switch is in the right (ON) position, the terminal will transmit, and expect to receive, 8 data bits. In the left (OFF) position, 7 data bits are used. The data bits are exclusive of the start, stop or parity bits.

Switch 2-6 Parity

If parity is enabled (switch 2-5 PARITY ENABLE) then the PARITY switch will select ODD parity in the right (ON) position, EVEN parity in the left (OFF) position.

Switch 2-5 Parity Enable

If parity is enabled (right (ON) position) then parity will be generated on transmission and checked on reception. The sense of the parity is determined by switch 2-6. If a parity error is detected during reception, the ASCII graphic for a null is displayed.

If parity is disabled (left(OFF) position) then parity is neither generated nor checked by the terminal.

Switches 2-4 to 2-1 Baud Rate

These switches determine the baud rate for communication on the RS232 interface between the terminal and the processor board. Refer to table on pg. 5-22 for details of the switch settings. The transmission rate may be varied from 50 to 9600 baud.

If the selected baud rate is greater than 150 baud then 1 stop bit is generated and checked for by the terminal on transmission and reception.

If the baud rate is less than or equal to 150 baud, then 2 stop bits are generated and checked.

5.2.3 PROGRAMMERS' INFORMATION

This section describes the effects of the various control and escape command sequences that are implemented in the terminal. Note that they are all compatible with the ANSI X3.64 standard for terminals.

5.2.3.1 CONTROL CODES

The control codes which have an effect on the terminal are listed below. Control codes not discussed are ignored by the terminal. The mnemonic and ASCII code (in hexadecimal) for each are given.

BELL (BELL,07H) Sounds the alarm at the terminal

BACKSPACE (BS,08H) Moves the cursor left one position, if possible.

HORIZONTAL TAB (HT,09H) Moves the cursor to the next defined tab position. If the cursor position at or past the last tab, the cursor does not move.

LINEFEED (LF,0AH) Moves the cursor down one line, if possible. If LNM (New Line) mode is set, the terminal automatically executes a carriage return. If the cursor is already on the last line of the scrolling region, the action taken depends on the settings of the terminal modes AUTOSCRL and AUTOCLR (see pg. 5-30 Terminal Modes).

FORM FEED (FF, 0CH) If AUTOSCRL is reset, form feed clears the scrolling region and moves the cursor to the HOME position. If AUTOSCRL is set, the form feed is treated as line feed.

CARRIAGE RETURN (CR, 0DH) Moves the cursor to column 1. Then, if the Auto Linefeed rear panel switch is enabled, the terminal automatically executes a linefeed.

SHIFT OUT (SO, 0EH) Activates the alternate character set.

SHIFT IN (SI, 0FH) Reverts to the standard character set.

ESCAPE (ESC, 1BH) Initiates an escape sequence.

CANCEL (CAN, 18H) Cancels an incomplete escape sequence.

5.2.3.2 KEYPAD MODES

The auxiliary keypad and the cursor control keys can operate in two modes, numeric mode and application mode. The difference between the two modes is that the affected keys generate different characters or escape sequences in each mode. Selection of numeric or application mode is by a short escape sequence.

	ESCAPE SEQUENCE
Set Keypad Numeric Mode	ESC >
Set Keypad Application Mode	ESC =

The KEY PAD APP MODE light on the keyboard display panel will be ON if APP mode is selected, OFF if numeric mode is selected.

The keys affected by APP/numeric mode are:

(a) The cursor control keys (including HOME)
(b) The digits 0 to 9 on the auxiliary keypad
(c) The keys " - "
" , "
" ENTER"
" . " on the auxiliary keypad.

Character sequences generated for these keys in each mode are shown in the following table (the function keys PF0 to PF4 are included for reference).

CAPTION ON KEY	LOCATON (MAIN OR AUXILIARY)	NUMERIC MODE SEQUENCE	APP MODE SEQUENCE
↑	M	ESC[A	ESC OA
↓	M	ESC[B	ESC OB
→	M	ESC[C	ESC OC
←	M	ESC[D	ESC OD
HOME	M	ESC[H	ESC OH
ENTER	A	CR	ESC OM
,	A	,	ESC Ol
-	A	-	ESC Om
.	A	.	ESC On
0	A	0	ESC Op
1	A	1	ESC Oq
2	A	2	ESC Or
3	A	3	ESC Os
4	A	4	ESC Ot
5	A	5	ESC Ou
6	A	6	ESC Ov
7	A	7	ESC Ow
8	A	8	ESC Ox
9	A	9	ESC Oy
PF0	M	ESC OT	ESC OT
PF1	A	ESC OP	ESC OP
PF2	A	ESC OQ	ESC OQ
PF3	A	ESC OR	ESC OR
PF4	A	ESC OS	ESC OS

KEYBOARD GENERATED KEY SEQUENCES

5.2.3.3 PARTITIONS AND SCROLLING REGIONS

The split-screen capability of the terminal is controlled by the partition screen command (PSR) and the select partition command (SSPR). The screen may be divided into two; the top partition (partition 0) may have from 0 to 25 lines, and the bottom partition (partition 1) will have the remaining number of lines, if any. Each partition may have a scrolling region defined within it by using the Set Scroll Area command (SSCRL). The scrolling region may be fully embedded in the partition, i.e. there may be fixed lines both above and below it.

(1) PSR - PARTITION SCREEN

The format of the Partition Screen command is:

	ESC [P1 p	Partition screen from line P1
or	ESC [p	no partition (i.e.partition 1 null)

The parameter P1, which should have a value from 0 to 25, is the line number of the beginning of the bottom partition. So partition 0 will occupy lines 1 to P1 - 1, partition 1 will occupy lines P1 to 25. If P1 is zero or null then partition 0 will occupy all 25 lines of the screen and partition 1 will be null.

The state of the terminal after the PSR command is executed is as follows:

(a) the screen is cleared
(b) partition 0 is selected
(c) the cursor is moved to the home position
(d) the scroll regions are set to the partition boundaries
(e) MARGIN mode is reset
(f) the graphic rendition is set to normal for each partition
(g) the tab settings are cleared

Examples:

ESC[12p	partition 0 from lines 1 to 11
	partition 1 from lines 12 to 25
ESC[1p	partition 0 null, partition 1 whole screen

	ESC[0p	
or	ESC[p	partition 0 whole screen
		partition 1 null

(2) SSPR - SELECT PARTITION

The format of the Select Partition command is:

	ESC [1 s	Select partition 1
or	ESC [0 s	Select partition 0
or	ESC [s	Select partition 0

The cursor position and graphic rendition are restored to the values held when the partition was deselected. If a null partition is selected the cursor is suppressed and all characters sent to the display are ignored.

(3) SSCRL - SET SCROLL AREA

This command defines the limits of the scrolling region within the selected partition.

The format of the Set Scroll Area command is:

	ESC [P1; P2 r	Scroll lines P1 to P2
or	ESC [r	Default to partition boundaries

The parameters P1 and P2 are the line numbers (relative to the partition boundaries) of the beginning and end of the scrolling region. If both parameters are null the scrolling region will be set to the partition boundaries.

Example:

ESC [11p	Partition screen,	10 lines for partition 0
		15 lines for partition 1
ESC [1 s	Select partition 1	
ESC [3; 15 r	Define a 13 - line scrolling region	
	with 2 fixed lines above, none below	

The cursor is moved to the home position.

5.2.3.4 TERMINAL MODES

A number of terminal software modes may be set or reset using escape sequences terminating with 'h' for set, 'l' for reset. More than one mode may be set or reset with one command.

Example:

```
ESC[ P1; P2; P3 h
set terminal modes P1, P2, P3
ESC[ P1; P2; P3 l
reset terminal modes P1, P2, P3
```

Modes are specified by decimal numbers, preceded by a question mark for certain modes.

MODE	PARAMETER	DESCRIPTION
KAM	2	Keyboard Action Mode
VEM	7	Vertical Editing Mode
LNM	20	Linefeed New Line Mode
MAR	?0	MARGIN Mode
AUTOSCRL	?1	Autoscroll Mode
AUTOSCRL	?4	Autoclear Mode

(1) KAM (2) KEYBOARD ACTION MODE

This mode is used to lock or unlock the keyboard. If KAM is set, the KEYBD LOCK LED will be on, and the terminal will not respond to any charcters typed at the keyboard.

```
ESC [ 2 h          set KAM, lock keyboard
ESC [ 2 l          reset KAM, enable the keyboard
```

(2) VEM (7) VERTICAL EDITING MODE

This mode affects the insert and delete line commands.

If VEM is reset, lines are deleted from the active line down by the Delete Line command, and the active line is pushed down by the Insert Line command.

If VEM is set, lines are deleted from the active line up by the Delete Line command, and the active line is pushed up by the Insert Line command.

ESC [7 l	reset VEM

(3) LNM (20) LINEFEED NEW LINE MODE

If LNM is set, the terminal will automatically execute a linefeed after a carriage return is received.

ESC [20h	set LNM

(4) MARGIN (?0) MARGIN MODE

This mode determines whether cursor movement and data transfer will be restricted to the scrolling region within the partition or to the entire partition. The restriction will apply to all operations involving cursor movement, line editing, data transfer and scrolling.

If MARGIN mode is reset, these operations will be confined to the scrolling region within the partition.

If MARGIN mode is set, the operations will be restricted only by the partition boundaries.

ESC [?0l	reset MARGIN

(5) AUTOSCRL (?1) AUTOSCROLL MODE, AUTOCLR (?4) AUTOCLEAR MODE

These two modes define the action of the terminal after receiving a linefeed when the cursor is at the last line of the scrolling region.

If AUTOSCRL is set the lines in the scrolling region shift up one row. The original top line is lost, and the bottom line is cleared. The setting of AUTOCLR has no effect in this case.

If AUTOSCRL is reset and AUTOCLR is set, the scrolling region is cleared and the cursor is moved to the home position.

If AUTOSCRL and AUTOCLR are both reset the display is not altered.

AUTOSCRL	AUTOCLR	ACTION
SET	SET	ALL LINES SCROLL UP
SET	RESET	ALL LINES SCROLL UP
RESET	SET	CLEAR REGION, CURSOR HOME
RESET	RESET	NO ACTION

LINEFEED RECEIVED, CURSOR ON LAST LINE

If AUTOSCRL is set, a form feed is treated like a linefeed; otherwise, the scrolling region is cleared and the cursor is moved to the home position.

ESC [?1l	reset AUTOSCRL
ESC [?4h	set AUTOCLR

This sequence ensures operation of the Autoclear feature.

5.2.3.5 CURSOR CONTROL

Cursor control facilities on the terminal include commands for cursor movement in any direction, cursor positioning by line and character number, and cursor position reporting.

Note that cursor movement is restricted to the currently selected partition and, within that partition, to the active scrolling region, dependent on the setting of MARGIN mode.

If MARGIN mode is reset the cursor may only be positioned within the scrolling area defined by the Set Scroll Area command.

If MARGIN mode is set the cursor may be positioned anywhere within the partition.

(1) CUU - CURSOR UP

The format of the Cursor Up command is:

```
     ESC  [  P1 A      move up P1 lines
or   ESC     LA        move up 1 line
```

The cursor will move up P1 lines (or 1 line if P1 is zero or null). If the movement specified is above the active scrolling region the cursor will move to the top line of the scrolling region.

(2) CUD - CURSOR DOWN

The format of the Cursor Down command is:

```
      ESC  [P1 B       move down P1 lines
or    ESC  [B          move down 1 line
```

The cursor will move down P1 lines (or 1 line if P1 is zero or null). If the movement specified is below the active scrolling region the cursor will move to the bottom line of the scrolling region.

(3) CUR - CURSOR RIGHT

The format of the Cursor Right command is:

```
      ESC  [P1C        move right P1 positions
or    ESC [C           move right 1 position
```

The cursor moves right P1 character positions (1 position if P1 is zero or null). If the movement specified is beyond the rightmost character position on the line the cursor will move to the last character position (column 80).

(4) CUL - CURSOR LEFT

The format of the Cursor Left command is:

	ESC [P1D	move left P1 positions
or	ESC [D	move left 1 position

The cursor moved left P1 character positions (1 position if P1 is zero or null). If the movement specified is beyond the leftmost character position the cursor will move to the first character postion on the line (column 1).

(5) CUP - CURSOR POSITION

This command may be implemented in two different formats, one in the standard ANSI control sequence and the other more convenient for programming.

Format 1 - ANSI standard

	ESC [P1; P2H	move to row P1, column P2
or	ESC [; H	move to row 1, column 1 (HOME)
or	ESC [H	move to row 1, column 1 (HOME)

The parameters P1 and P2 are the row and column numbers. If the specified position is outside the current scrolling region the command will be ignored. A null or zero parameter is treated as a 1.

Format 2 - Program convenient

ESC R line column

This format is more convenient for program implementation. The two ASCII characters following the R correspond to the line and column numbers. The hexadecimal value 1FH is added to the position number, and the corresponding ASCII character is used in the format.

Example:

```
     20H, ASCII 'space', corresponds to position 1
     21H, ASCII '!', corresponds to position 2
     etc.
so   ESC R%* moves the cursor to line 6, column 11
     (% is 25H, * is 2AH)
```

(6) CPR - CURSOR POSITION REPORT

This command requests the terminal to report the current cursor position. The request may be in two formats, one as a special case of the Device Status Report command, the other as the Cursor Position Report command. The reply is always in the same format.

Request format:

```
     ESC [6 n
or   ESC [ R
```

Reply format:

```
     ESC [P1; P2 R
```

The reply parameters P1 and P2 are the line and column numbers, as ASCII strings.

Example:
If the cursor is in the HOME position the reply will be ESC [01; 01R

(7) RI - REVERSE INDEX

The Reverse Index command may be used to cause the terminal to scroll in the opposite direction, i.e. lines move downwards.

The format of the Reverse Index command is:

```
     ESC M
```

The cursor will move up one line unless it is already at the top line of the scrolling region. In this case the region will scroll down one line, and the top line will be cleared ready for incoming data. The downward scrolling is still subject to slow scrolling, controlled by the Slow Scroll Key.

5.2.3.6 LED CONTROL

The three LED's on the keyboard (L0 to L2) and the eight LED's on the front panel (D0 to D7) are controlled by the escape sequence terminating with 'q'. The format of the escape sequence is as follows:

ESC [P0; P1; P2.....q

The parameter P0 should be 0 to turn off LEDs or 1 to turn on LEDs. The other parameters are numbers indicating which LEDs to control.

LED NAME	LED NUMBER
L2	1
L1	2
L0	3
D7	4
D6	5
D5	6
D4	7
D3	8
D2	9
D1	10
D0	11

Examples

ESC [1; 1; 11q turn on L2 and D0

ESC [q turn off all LEDs

5.2.3.7 ATTRIBUTES

Characters may be assigned a number of attributes:

blink, faint intensity, reverse video, underline, overstrike, or any combination of these (or none, known as prime rendition).

The character rendition is defined using the Select Graphic Rendition escape sequence (SGR, terminates with 'm').

```
ESC [P1; P2.....m
```

PARAMETER	ATTRIBUTE
0 or Null	Prime Rendition
2	Faint
4	Underline
5	Blink
7	Reverse Video
99	Overstrike

Once a graphic rendition has been defined, all characters transferred to the partition have this rendition, until changed by another SGR command. Definition of character attributes does not occupy space on the screen

Example:

```
ESC   [5; 99m          select blink and overstrike
ESC   [m               select prime rendition
```

Restrictions

Each line is restricted to 15 rendition changes.

5.2.3.8 FORM-DRAWING CHARACTERS

A number of form-drawing characters are available. These are sent to the display one at a time using the GRAPH command.

The format of the GRAPH command is:

```
      ESC   [P1; P2 t
or    ESC   [P1 t
```

The parameter P1 specifies the form character and the parameter P2 specifies the intrinsic attribute of the form character, as in the following tables:

P1	GRAPHIC	DESCRIPTION
0	┌	Top left corner
1	┐	Top right corner
2	└	Bottom left corner
3	┘	Bottom right corner
4	┬	Top intersect
5	┤	Right intersect
6	├	Left intersect
7	┴	Bottom intersect
8	─	Horizontal line
9	│	Vertical line
10	┼	Crossed lines

4-3289-10

FORM CHARACTERS

P2	ATTRIBUTES
0 or null	Not faint, not blinking
1	Faint
2	Blinking
3	Faint and blinking

FORM CHARACTER ATTRIBUTES

As can be seen in the above table, the attributes of blinking and faint intensity may be specifically given to the form character. However, the form character will also take on the attributes underline and reverse video from the preceding field on the same line.

Example:

The current line is displayed in reverse video.

Then the sequence:

```
ESC [10; 2 t
```

will write blinking reverse video crossed lines in the active position.

Note the following restriction on the use of form characters. If a form character is preceded by a non-form character, then the two characters preceding the form character must have the same rendition.

This is illustrated in the following example where the character sequence ABF is to be placed on the screen, where A and B are non-form characters and F is a form character.

If A and B have different renditions it will not be permissable to write the form character F. If this is attempted the GRAPH command will be ignored.

If the sequence ABF is already on the screen and the rendition of A or B is changed, then the effect on the form character F will be unpredictable.

5.2.3.9 TABULATION

The TAB command may be used to set or reset the terminal's tab settings. A tab is set by first positioning the cursor in the required column (on any available line, as tab settings apply to all lines). The TAB command is then sent to the terminal. If the cursor is subsequently to the left of this position a Horizontal Tab control character will move the cursor to the tab setting.

A maximum of 16 tab settings are allowed, any additional TAB commands will be ignored.

The format of the TAB command is:

```
   ESC [P1 W
or ESC [W
```

P1	ACTION
0 or null	Set tab at current position
2	Clear tab from current position
5	Clear all tab settings

TABULATION CONTROL

Tab settings apply to both partitions.

5.2.3.10 EDIT COMMANDS

With the edit commands it is possible to clear part or all of the scrolling region, part or all of the current line, and insert or delete lines or characters.

The insert and delete line commands are affected by the terminals VEM mode (Vertical Editing Mode). This determines whether lines are inserted or deleted upwards (VEM set) or downwards (VEM reset) from the current line.

(1) ED - ERASE IN DISPLAY

The format of the ED command is:

	ESC	[P1 J	erase part or all of region
or	ESC	[J	erase to end of region

This command causes the specified portion of the display to be erased (set to prime rendition blanks).

P1	ACTION
0 or null	Erase from the cursor position to the end of the region. The cursor does not move.
1	Erase from the start of the region to the cursorposition. The cursor does not move.
2	Erase the entire scroll region. Move the cursor to the HOME position.

ED COMMAND OPTIONS

(2) EL - ERASE IN LINE

The format of the EL command is:

	ESC	[P1 K	erase part or all of line
or	ESC	[K	erase to end of line

The EL command causes part or all of the current line to be erased. The cursor position does not change for any of the following options.

P1	ACTION
0 or null	Erase from the cursor position to the end of the line.
1	Erase from the start of the line to the cursor position.
2	Erase the entire active line.

EL COMMAND OPTIONS

(3) DL - DELETE LINES

The format of the Delete Lines command is:

	ESC	[P1 M	delete P1 lines
or	ESC	[M	delete 1 line (the current line)

The DL command deletes P1 lines from the current scroll region. A value of 1 is assumed if P1 is 0 or null. If VEM is reset then P1 lines from and including the current line down are deleted. The remaining lines, if any, are moved up, replaced at the bottom of the scrolling region by blank lines.

If VEM is set then P1 lines from and including the current line are deleted. The remaining lines are moved down, replaced at the top of the scrolling region by blank lines.

If the number of lines to the region boundary is less than P1 then that number will replace P1.

Example:

The scroll region contains 10 lines, the cursor is on line 2. If VEM is reset then ESC [3M deletes lines 2, 3 and 4; lines 5 to 10 move up 3 lines; 8, 9 and 10 are erased. If VEM is set, then ESC [3M deletes lines 1 and 2.

(4) IL - INSERT LINES

The format of the Insert Lines command is:

```
     ESC   [P1 L      insert P1 lines
or   ESC   [L         insert 1 line
```

The IL command inserts P1 blank lines at the current line position. A value of 1 is assumed if P1 is 0 or null.

If VEM is reset the displaced lines are pushed downwards, lower lines being lost at the bottom of the scrolling region.

If VEM is set the displaced lines are pushed upwards, upper lines being lost at the top of the scrolling region.

If the number of lines to the region boundary is less than P1 then that number will replace P1.

Example:

The scrolling region contains 10 lines, the cursor is on line 2.

If VEM is reset then ESC [3 L moves lines 2 to 7 down to replace lines 5 to 10, lines 2,3 and 4 are erased, and the original lines 8,9 and 10 are lost.

(5) DCH - DELETE CHARACTERS IN LINE

The format of the DCH command is:

ESC [P1 P	delete P1 characters
or ESC [P	delete 1 character

The specified number of characters are deleted from the current line, starting from the cursor position, inclusive, and going to the right. If P1 is 0 or null one character is deleted.

The remaining characters on the line are shifted left to replace the deleted characters, with prime rendition spaces being shifted in at the right. If the specified number of characters is greater than the number remaining on the line then the remainder of the line is erased.

Example:

The cursor is at column 70, the current line is displayed in reverse video

ESC [5 P

moves the reverse video characters in columns 75 to 80 into columns 70 to 75, and moves prime rendition spaces into the five vacated positons 76 to 80.

(6) ICH - INSERT CHARACTERS IN LINE

The format of the ICH command is:

ESC [P1 @	insert P1 blanks
or ESC [@	insert 1 blank

This command inserts P1 prime rendition spaces in the current line at and to the right of the cursor position. Displaced characters are shifted P1 places right, those moving off the edge of the screen are lost. If P1 is 0 or null 1 space is inserted. The cursor position does not change.

Example:

The cursor is at column 70, the current line is displayed in reverse video

ESC [5 @

moves 5 prime rendition spaces into positions 70 to 74 and moves the reverse video characters in 70 to 75 into positions 75 to 80. The original 5 characters in positions 76 to 80 are lost.

5.2.3.11 STATUS AND CURSOR REPORTS

The Device Status Report command is used by the processor to determine the terminal status or cursor position. The command options and terminal responses are described below.

(1) REQUEST TERMINAL STATUS

Processor request:	ESC [5 n
Terminal response:	ESC [0 n

If no response is received the terminal is not available for command processing.

(2) REQUEST CURSOR POSITION

Processor request:	ESC [6 n or ESC [R
Terminal response:	ESC [P1; P2R

This command causes the terminal to reply with the Cursor Position Report. The reply parameters P1 and P2 are the line and column numbers of the cursor position. Line numbering is with respect to the current scrolling region.

5.2.3.12 ALTERNATE CHARACTER SET

Provision is made on the terminal controller board for installing a 2716 EPROM, or equivalent, containing a user-defined alternative character set. The characters are displayed using two mechanisms, dependent on the hexadecimal character code. The workstation is normally supplied with a PROM containing a Danish character set as the alternative. See section 3.2.2 for details. See also section 5.2.4.7 (pages 5-57..58).

(1) 00H TO 1FH

Characters in this range are output one at a time by using a short escape sequence

ESC N P1

The parameter P1 is an ASCII character generated by adding 40H to the character code.

Example:	ESC N B Would display the alternate character corresponding to 02H (B is 42H)

Note that although characters in this range (00H to 1FH) would normally not be displayable, when using the alternative character set they are displayable instead of having controlling functions.

(2) 20H TO 7FH

Characters in this range (corresponding to the ASCII displayable character set) are output by first activating the alternate character set using the SO control code (0EH).

Subsequently, codes in the range 20H to 7FH will display the character in the alternate set. The SI control code (0FH) will switch the terminal back to the primary character set.

SO :	Switch to alternate character set
SI :	Revert to primary character set

5.2.3.13 RESET

The Reset command is a short escape sequence which will re-initialize the terminal's software modes to the power-on conditions. The format of the Reset command is:

ESC c

The processor should delay approximately ½ second to allow the terminal to complete the initialization procedure.

The terminal is set to the following state:

(a) the RS232 input and output buffers are cleared
(b) the screen is erased
(c) the screen is partitioned with 24 lines in partition 0, 1 line is partition 1
(d) partition 0 is selected
(e) the cursor moves to the HOME position
(f) the primary character set is selected
(g) prime rendition is selected
(h) tab settings are cleared
(i) the terminal's software modes are defined as follows.

KAM	Keyboard action	reset
VEM	vertical editing	reset
LNM	new line	reset
MARGIN	margin	reset
AUTOSCRL	auto scroll	set
AUTOCLR	auto clear	reset

5.2.3.14 SCREEN ALIGNMENT DISPLAY

The Screen Alignment Display command has been included to provide a useful pattern for adjustments to the display.

The format of the command is:

ESC # 8

The entire screen area will be filled with upper-case E's.

The command is equivalent to the following sequence:

(a) partition screen with 25 lines in partition 0, 0 lines in partition 1.

(b) transmit 25 lines of E's

(c) send the cursor HOME.

5.2.3.15 ESCAPE SEQUENCE SUMMARY (PROCESSOR TO TERMINAL)

SEQUENCE	DESCRIPTION
ESC =	set keypad application mode
ESC >	set keypad numeric mode
ESC M	reverse index
ESC c	reset to power-on state
ESC # 8	screen alignment display
ESC N P1	display alternate control code character
ESC R l c	move cursor to line l, column c
ESC [P1 @	insert P1 blank characters in line
ESC [P1 A	move cursor up P1 lines
ESC [P1 B	move cursor down P1 lines
ESC [P1 C	move cursor right P1 lines
ESC [P1 D	move cursor left P1 lines
ESC [P1; P2 H	move cursor to line P1, column P2
ESC [P1 J	erase in display
ESC [P1 K	erase in line
ESC [P1 L	insert P1 lines
ESC [P1 M	delete P1 lines
ESC [P1 P	delete P1 characters in line
ESC [R	request cursor position
ESC [P1 W	set/reset tab at cursor position
ESC [P1; P2... h	set terminal modes P1, P2...
ESC [P1; P2... l	reset terminal modes P1, P2...
ESC [P1; P2... m	set character rendition
ESC [P1 n	request cursor position/terminal status
ESC [P1 p	position screen at line P1
ESC [P; P1; P2...q	turn off/on LED's P1, P2...
ESC [P1; P2 r	set scroll area between lines P1 and P2
ESC [P1 s	select partition P1
ESC [P1; P2 t	display FORM character P1 with attribute P2

ESCAPE SEQUENCE SUMMARY
(TERMINAL TO PROCESSOR)

SEQUENCE	DESCRIPTION
ESC O P1	Function key PF0 to PF4 or numeric keypad/cursor control key in APP mode
ESC [P1; P2 R	Cursor position report, row P1, column P2
ESC [0 n	Terminal ready
ESC [A	Cursor up key (numeric mode)
ESC [B	Cursor down key (numeric mode)
ESC [C	Cursor right key (numeric mode)
ESC [D	Cursor left key (numeric mode)
ESC [H	Cursor HOME key (numeric mode)

5.2.4 HARDWARE CONFIGURATION DETAILS

N.B. Except where stated otherwise, these details apply to the 6-slot version of the workstation.

Apart from the DIP switches described previously (on pgs. 5-19..24) for setting Baud rate etc., there are various strap-selectable options on the Multibus Backplane Motherboard. The options are concerned mainly with bus arbitration; also with the use of the interrupt and reset buttons on the front of the workstation.

The Multibus Motherboard is horizontally situated in the bottom of the CR8 Workstation.

Note that most of the processing required to implement the Multibus standard takes place within the Multibus modules (MP^2 card, Winchester controller, etc.) themselves. (In fact the cabinet model CR8 does not have any logic circuitry in its Multibus Motherboard.) Its main function is to implement parallel priority bus arbitration whereby all devices connected essentially have the same priority. This feature is selectable; the more usual serial priority method may be used instead if desired. (See next section).

5.2.4.1 BUS ARBITRATION

The CR8 workstation is supplied set up for parallel Multibus arbitration. For serial arbitration, connector and card position J6 (or J9 in some models) is the lowest priority slot (closest to the rear of the unit) while J1 is the highest. Normally the processor would be installed in the lowest priority slot. The Motherboard provides the chaining of the BPR0/ to BPRN/ signal between cards slots. Also, wirewrap posts are provided to connect parallel generated BPR0/ (grant) signals for parallel bus arbitration; the posts also allow the BPRN/ input of the last card slot used to be grounded thus making it the highest priority. Normally the cardcage would be populated by cards from the lowest priority slot J6 or J8 toward the highest (J1). Newer (grey) CR8 workstations have 8 slots while older (brown) ones have only 6.

Important:

The highest priority slot of the serial arbitration chain must have the shunt installed to ground its BPRN/Spare input in order for the arbitration signal chain to work properly. The factory installed shunt between posts A and B for slot J1 may be moved as appropriate. Refer to the illustration of the Motherboard on pg. 5-52 for the locations of the wirewrap posts. The wirewrap posts are defined as follows:

LABLED PAIR	FUNCTION
A-B	Grounds BPRN/ signal for J1
D-E	Grounds BPRN/ signal for J2
G-H	Grounds BPRN/ signal for J3
J-K	Grounds BPRN/ signal for J4
M-N	Grounds BPRN/ signal for J5
P-Q	Grounds BPRN/ signal for J6

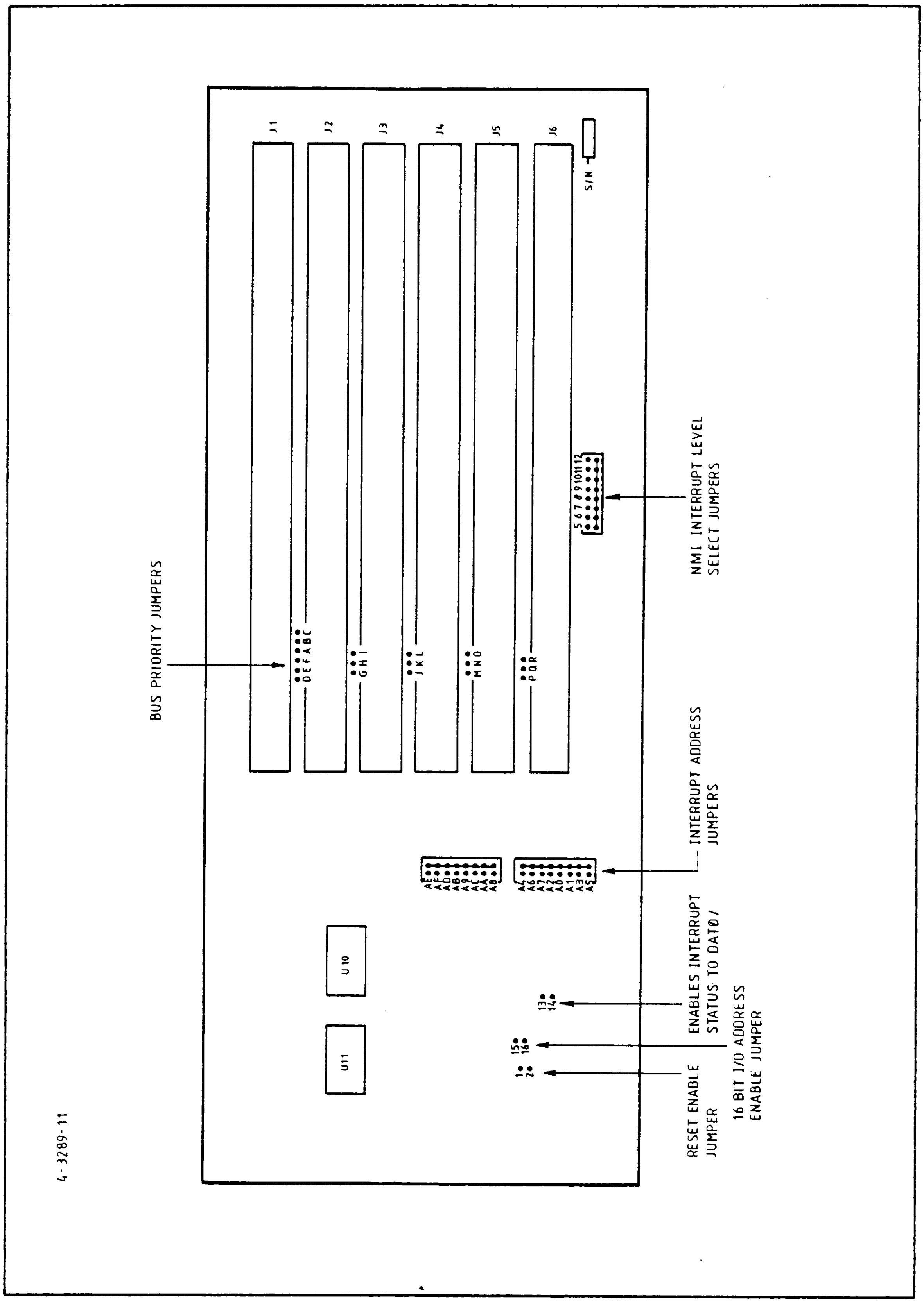
SERIAL PRIORITY CHAIN CONFIGURATION

Parallel bus arbitration may be implemented with the Backplane Motherboard. The user will need to determine whether this step is necessary by referring to the specifications for the Multibus boards that are being used. If the total on-board serial delays exceed the period of the bus arbitration clock minus input setup times and safety margins then parallel arbitration is required.

Parallel arbitration is implemented by installing U10 (74148) and U11 (74S138) in the sockets provided on the Motherboard. These are both special 16-pin units obtainable from CR. Refer to the Motherboard illustration on pg. 5-52 for their locations. Any grounding shunts initially installed for serial arbitration must be removed for proper operation.

Important:

When parallel arbitration is used, the Multibus boards installed must be configured to disconnect the BPR0/ Spare signal from reaching the P1 connector. For instance, Intel short E151-E152 would be removed on the 86/12A board. Boards installed in the lowest priority slot (J6 or J8) need not have this done since its BPR0/ signal is not used. Failure to do this will result in contention between TTL output drivers in the arbitration logic and the on-board PBR0/ driver. Additionally, for parallel arbitration, shunts must be installed to provide BPR0/signals from the arbitration logic on the Motherboard to each slot which contains a bus master. The wirewrap shorting posts to accomplish this are as follows:



MULTIBUS MOTHERBOARD JUMPER POSITIONS

LABLED PAIR	FUNCTION
B-C	Connects PBR0/ (parallel grant) to J1
E-F	Connects PBR0/ (parallel grant) to J2
H-I	Connects PBR0/ (parallel grant) to J3
K-L	Connects PBR0/ (parallel grant) to J4
N-O	Connects PBR0/ (parallel grant) to J5
Q-R	Connects PBR0/ (parallel grant) to J6

PARALLEL PRIORITY JUMPERS

5.2.4.2 RESET SWITCH LOGIC

The CR8 Workstation provides a front panel switch which may be used to reset boards installed in the backplane. Debounce logic and drivers are provided on the Motherboard. A factory installed shunt between posts labelled "1" and "2" connects the reset signal to the INIT/Spare signal on the backplane (initialization signal). The shunt can be removed to disable the reset switch.

5.2.4.3 INTERRUPT SWITCH LOGIC CONFIGURATION

The front panel Interrupt switch can be configured to provide a NON-BUS VECTORED type of interrupt to the Multibus system using any of the eight Multibus interrupt request lines. It can be further configured to respond to an 8 or 16 bit I/O address for clearing or reading the Interrupt request flip flop.

Option posts for the interrupt logic are defined as follows. Refer to pg. 5-52 for location on the Motherboard PCB.

Note: The Interrupt logic for the interrupt switch is not reset on power up. Therefore in applications which use this switch it is recommended that software clears the interrupt request as part of its initialization.

POST(S)	FUNCTION
13-14	Connects the Interrupt flip flop to data bus bit line DAT0/ to allow the CPU to read its value.
15-16	Connects the upper 8 address bit comparator to allow response to 16 bit I/O addresses. When not installed only 8 bit I/O addresses are used.
A0-A9 AA-AF	These positions configure the I/O address to which the Mother-board Interrupt Logic will respond. If an address bit is to be recognized as a logic one then the corresponding shunt must be installed in the position labeled with the address (i.e. ADR0=A0, etc).
5	Connects Front Panel Interrupt to INT6/ line.
6	Connects Front Panel Interrupt to INT7/ line.
7	Connects Front Panel Interrupt to INT4/ line.
8	Connects Front Panel Interrupt to INT5/ line.
9	Connects Front Panel Interrupt to INT2/ line.
10	Connects Front Panel Interrupt to INT3/ line.
11	Connects Front Panel Interrupt to INT0/ line
12	Connects Front Panel Interrupt to INT1/ line.

INTERRUPT CONFIGURATION JUMPERS

Performing an I/O write cycle to the configured interrupt I/O address port will clear the interrupt flip flop after it has been set by the leading edge of the interrupt switch signal. Reading the same address will allow the processor to read the status of the interrupt flip flop in the least significant data bit. Note: The I/O address is set at the factory to 0000H. In the event that address is used by any Multibus board it will have be reconfigured. Note that the Interrupt switch is not implemented in the CR8, but it can be by means of a jumper on the MP^2 card and software changes.

5.2.4.4 MULTIBUS IEEE 796 DIFFERENCES

The workstation's Multibus Motherboard was designed to the requirements of the Intel Multibus specifications as outlined in the Manual Order Number 98000683. However, the Intel P1 connector Pin 25 is used as the LOCK signal as required by the IEEE Standard. The workstation backplane also provides -5V which is not required by the IEEE specification but is by the Intel Multibus.

5.2.4.5 BACKPLANE POWER CAPACITY

The following table shows the maximum available power on the Multibus backplane.

VOLTAGE SPECIFICATION	MAXIMUM AVAILABLE CURRENT
+12.0 V ± 5.0%	2.4 Amps
+ 5.0 V ± 3.0%	22.0 Amps
-12.0 V ± 5.0%	1.5 Amps
-5.0 V ± 5.0%	0.5 Amps

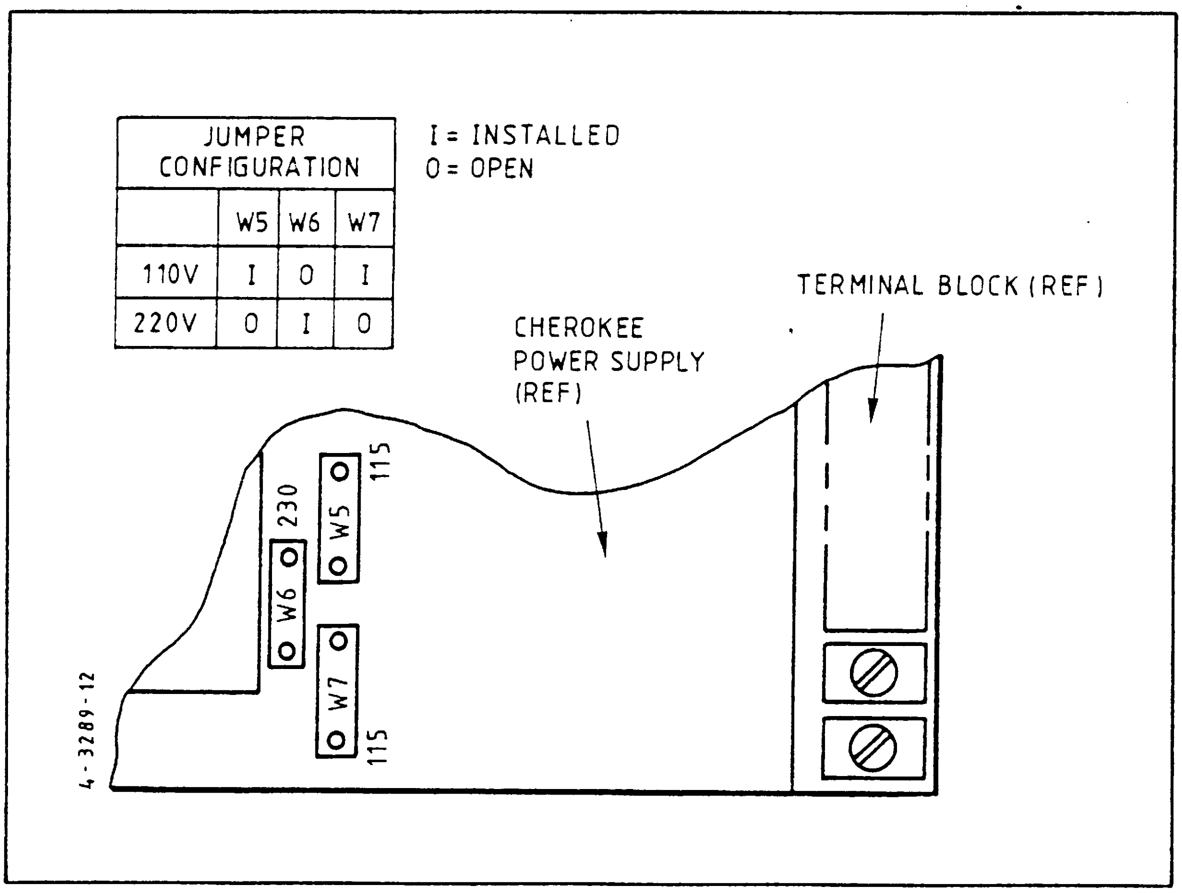
The combination of the above supplies are not to exceed the power limit, which is 165 Watts minus the power used by the disk drives.

CARDCAGE POWER SPECIFICATIONS

5.2.4.6 VOLTAGE SELECT

This switch selects the line voltage for fan operation and the CRT screen refresh frame rate. Place the switch in the left position for 110V/60Hz operation; place the switch in the right position for 220V/50Hz operation. To change the switch setting, remove the set screw, move the switch to the desired position then replace and secure the set screw. For models using the Cherokee switching power supply (serial numbers starting with 25014), voltage selection jumpers must be reconfigured at this time: Open the unit to access the base area and reconfigure jumpers W5, W6 and W7 as follows: For 110V operation, install W5 and W7 and remove W6. For 220V operation, install W6 and remove W5 and W7.

WARNING: Set this switch and configure power supply jumpers **PRIOR** to turning the unit on. Failure to do so damages the power supply and/or fans and voids the warranty.



JUMPER CONFIGURATION			
	W5	W6	W7
110V	I	O	I
220V	O	I	O

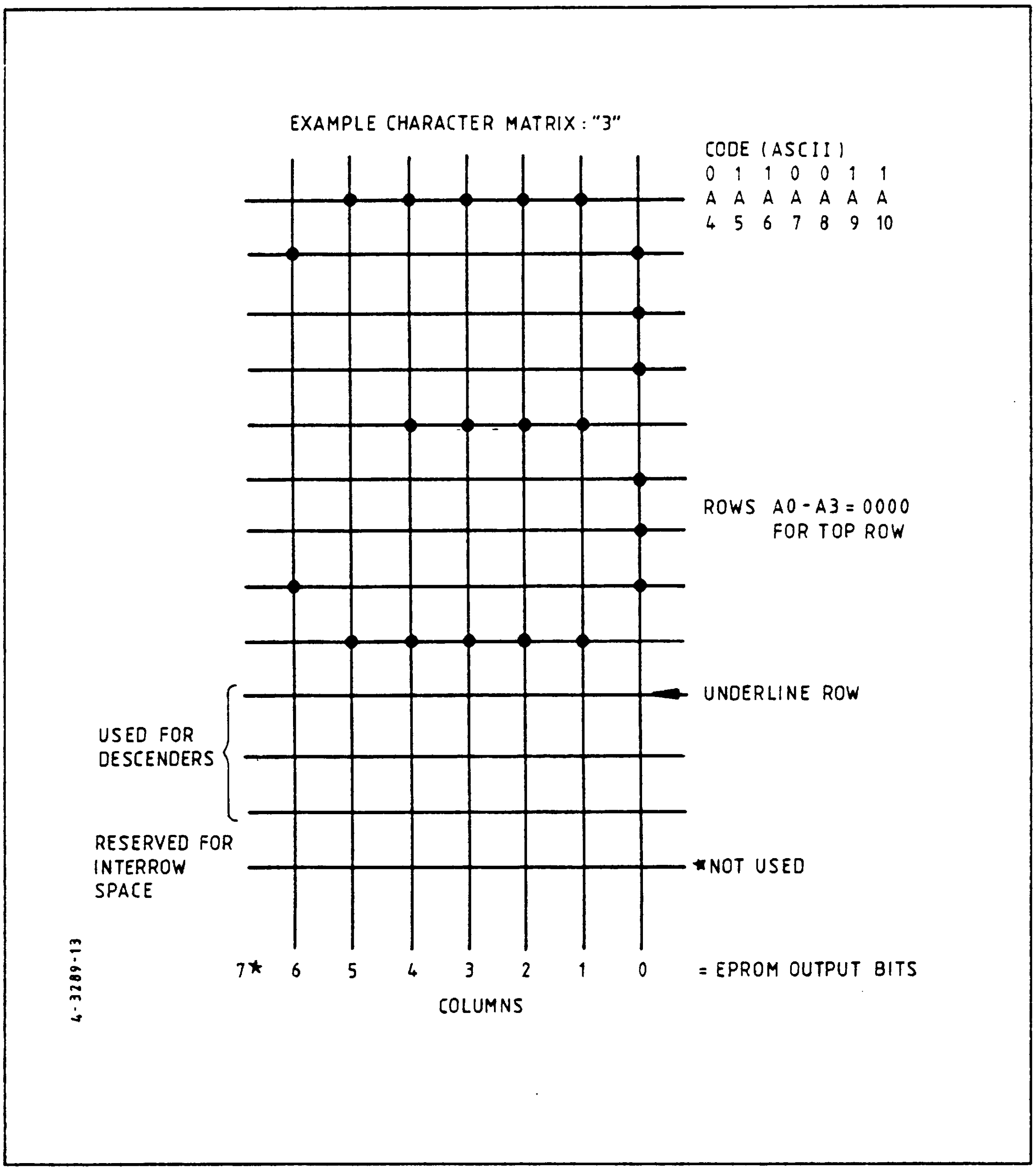
I = INSTALLED
O = OPEN

110V/220V JUMPER CONFIGURATION

5.2.4.7 ALTERNATE CHARACTER SET ROM GENERATION

An alternate character set can be incorporated in the workstation by installation of a 2716 EPROM or equivalent device in IC location 42 on the terminal controller board. Software driven commands are available for invoking the character set and have been described previously. Up to 128 characters can be included in the alternate character ROM. Sixteen bytes are stored in sequence for each character to define its character font (see diagram overleaf). A seven-bit binary code is used to address each character as it is displayed, in the same way that ASCII character codes are used to address characters in the standard set EPROM. Therefore, the seven bit character code is used on address lines A4-A10 of the 2716 where A4 is the least significant bit of the character code sent to the terminal. Address lines A0-A3 are used to select the rows of the character matrix as each character is displayed.

Within the data stored in the character generater EPROM, each byte stored represents a row of the character matrix. Bit 7 output is not used since the character matrix is 7 columns by 13 rows. An example of the character matrix format appears overleaf.



CHARACTER MATRIX FORMAT

5.2.4.8 TERMINAL RS232 PHYSICAL INTERFACE

The terminal controller board provides a standard 25 pin "D" type female communication connector and circuitry as per EIA Standard RS232C. Configuration interface type "D" is used for Duplex operation with Request to Send used to indicate a non-transmit mode to the local host (Multibus System Processor). The one exception to the type "D" interface is that the workstation is not microprogrammed (on the terminal controller) to respond to the Modem Ring Indicator signal (RS232C Circuit CF).

Data transmission protocol is industry standard asynchronous start-stop protocol. One stop bit is used at baud rates of 300 and above and two at 150 and below.

The terminal controller can be configured to disable use of various interchange circuits to accomodate simpler host interfaces which perhaps do not generate the required status signals. Refer to the table below for shunt (jumper) locations on the terminal controller board. The four last jumpers depicted in the table allow reconfiguration of the transmit and receive clocks of the RS232 terminal interface in order to use host supplied clocks.

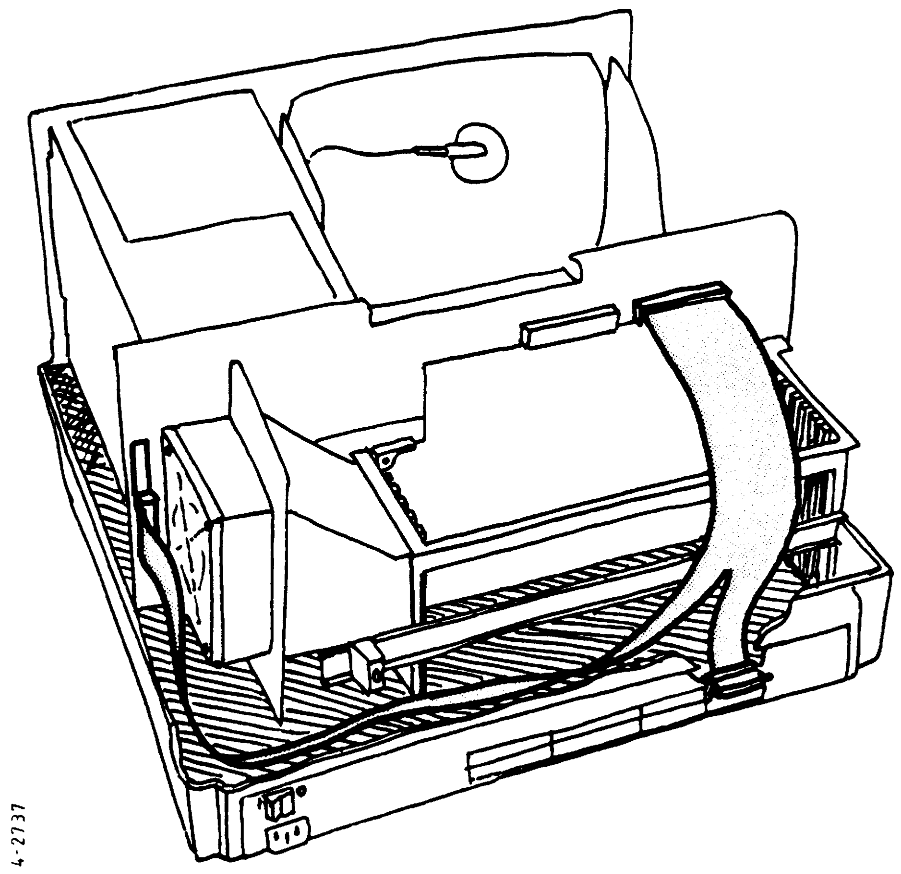
In order to gain access to the terminal controller board, remove the four access screws (two in the bottom front panel and the two under the rear cover at the top of the unit). The board is installed vertically at the far right hand side of the Workstation (seen from the front).

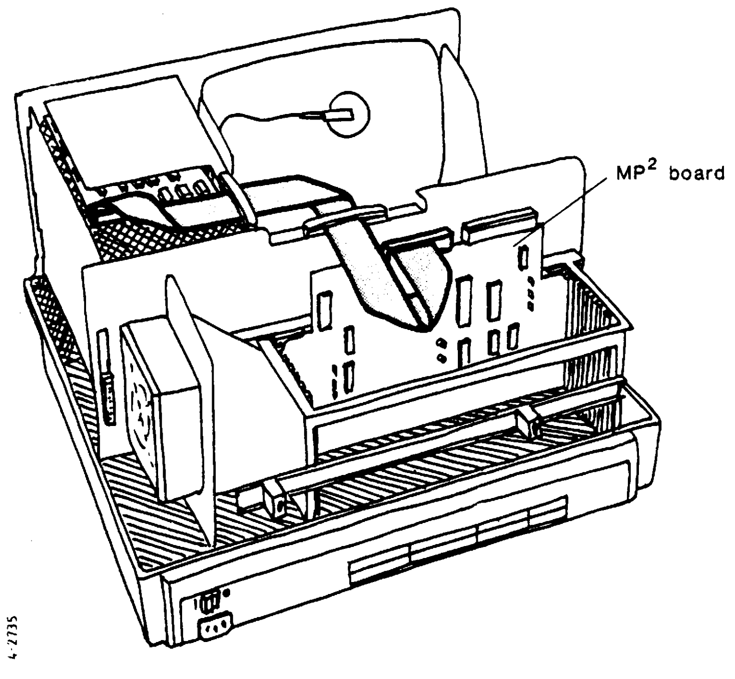
SHUNT		RS232 INTERCHANGE CIRCUIT
31-32.	Enable Clear To Send from Host	(CKT CB)
31-30	Disable Clear To Send from Host.	
34-33	Enable Data Set Ready from Host.	(CKT CC)
34-35	Disable Data Set Ready from Host.	
18-19	Enable Carrier Detect from Host.	
17-18	Disable Carrier Detect from Host.	
23-24	Use Internal Terminal Receive Clock	
23-25	Use Receive Signal Element Timing from Host for Receive Data Clock.	
20-21	Use Transmitter Signal Element Timing from Host for Transmit Clock.	(CKT DD)
21-22	Use Internal Terminal Baud Clock.	(CKT DB)

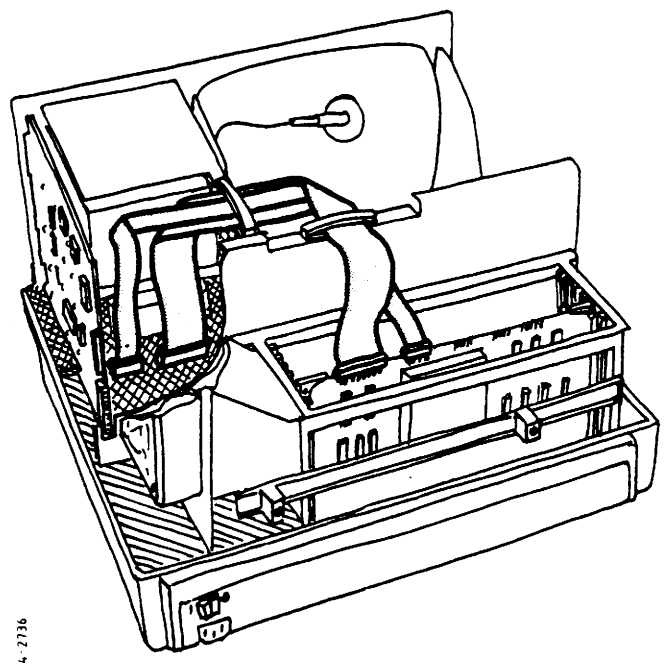
RS232 JUMPER SELECTION

5.2.4.9 CABLING

When the MP^2 - board is in position, a dual V.24 cable is installed as shown below. The cable itself is illustrated on pg. 5-98. Disk drive cabling is shown overleaf.







CABLE ROUTING - FLOPPY (above) AND WINCHESTER (below)

5.3 MP^2 PROCESSOR BOARD

The features of the MP^2 dual processor board (which forms the heart of the CR8) were described on pages 2-2 to 2-4. This section (5.3) describes in detail how these features are implemented.

5.3.1 INTRODUCTION

The Multi Purpose Multi Processor board (MP^2) specified in this document is a high performance dual microcomputer configured with a Multibus interface by which the module may communicate with slave modules as well as master modules.

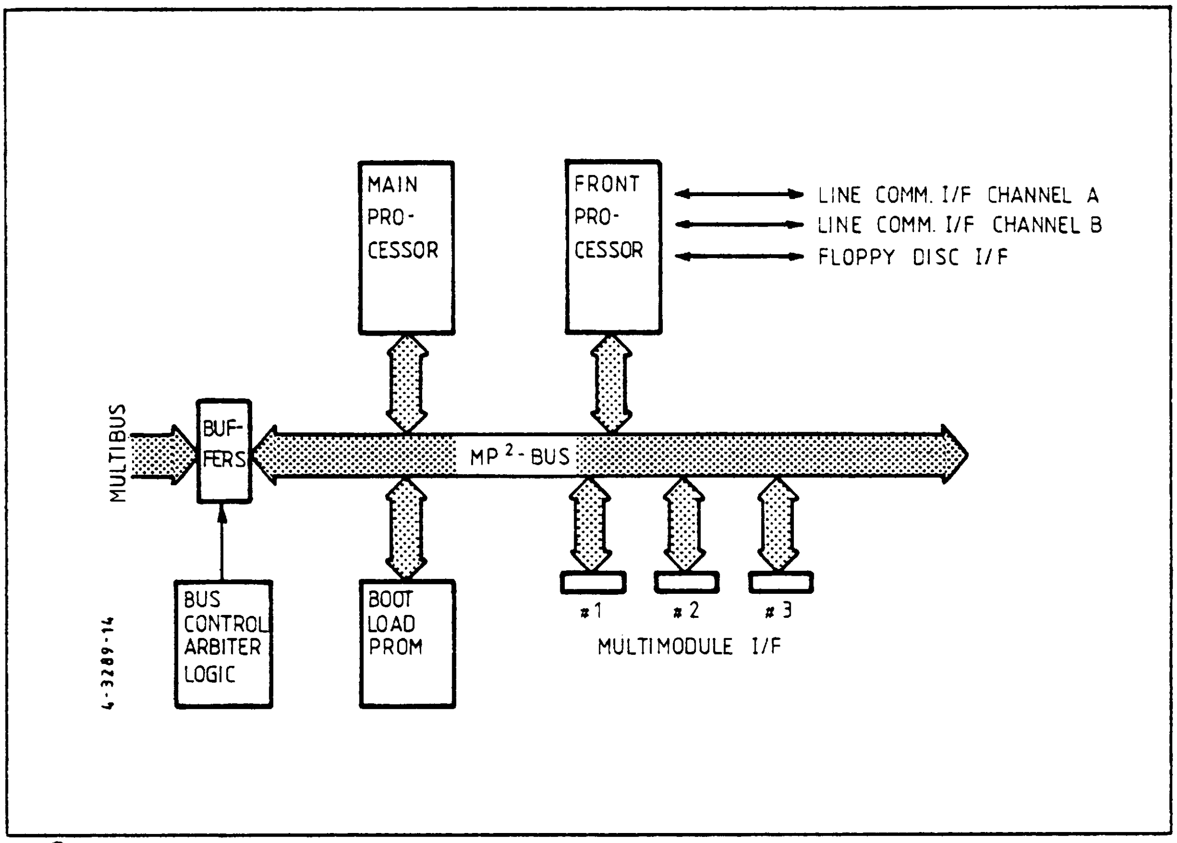
The two microcomputers are configured as general microcomputers with common access to all on-board I/O devices, to the Multibus and to all on-board memory.

Both computers may be equipped with up to 64 Kbytes of memory (Dynamic RAM) which can have battery backup to ensure no data loss during power failures.

To obtain the highest degree of expandability, the internal bus structure (MP^2 Bus) is connected to three 36 pin on-board connectors into which Multimodules and CR custom interface modules may be plugged.

A central Bus Control and Arbiter Logic is implemented in order to control the Multibus, MP^2 internal bus and MP^2 interfaces.

A PROM area is interfaced to the MP^2 Bus. This area is normally only accessed during a boot load procedure when the built-in test programs are run. This PROM area is 4K Byte but is extendable to 64K by use of a small on board extension board.



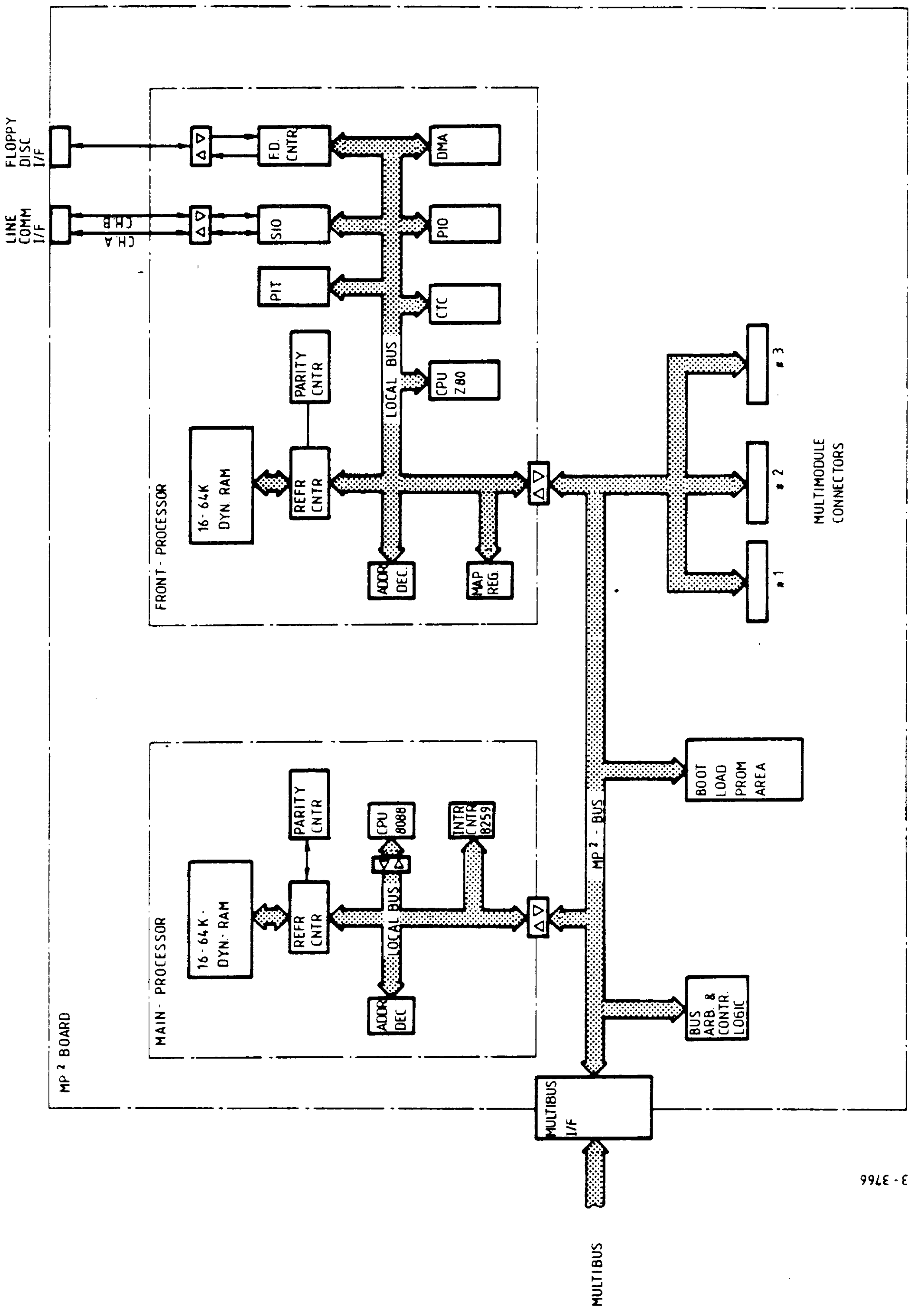
MP^2 BUS STRUCTURE

5.3.2 FUNCTIONAL SUMMARY

This brief functional description is based on the block diagram overleaf. It shows the common internal bus, called the MP^2 Bus, which interfaces to the following logic functions:

- o Main Processor
- o Front Processor
- o Multibus Interface
- o Multimodule Interface
- o Bus Arbiter & Control Logic
- o Boot Load Prom

The following sections describe briefly the functions and control of these logic blocks.



MP^2 BLOCK DIAGRAM

Key to Block Diagram (in alphabetical order):

ADDR DEC:	Address decoder
CH A:	Channel A
CH B:	Channel B
CTC:	Interrupt controller/timer (Control Timer Circuit)
FD CNTR:	Floppy disk controller
INTR CNTR:	Interrupt controller
PIO:	Parallel I/O
PIT:	Programmable interval timer
REF. CNTR:	Refresh controller (dynamic RAM)
SIO:	Serial I/O

5.3.2.1 MAIN PROCESSOR

The Main Processor is a microcomputer with its own local bus, 8 bit data bus and 20 bit address bus allowing control of 1 MByte memory.

The Main Processor consists of a CPU (i8088), and interrupt controller (i8259), 64kbytes of RAM with parity control and an MP^2-bus interface. Refresh and control of the memory is carried out automatically by the RAM interface circuit.

RESET: is activated by the Power Control Logic when the power supplies are below the defined levels, and is kept active for about 40 msec. after the levels have been re-established. Reset may be supplied to, or activated from the Multibus. See sect. 5.3.2.1.

NMI: is activated by a parity error, by time out during MP^2-bus access, by a watchdog alarm, or by Power Fail interrupt. NMI on the main processor is ored together with NMI on the Front Processor.

The **Interrupt Controller** (i8259A) controls the following interrupts, all supplied from the Front Processor.

IR0:	Interrupt from Multibus Interface
IR1:	Floppy disc controller

IR2:	Channel 0 from PIT (8253)
IR3:	Interrupt from front processor
IR4:	Not used
IR5:	Multimodule interface 0
IR6:	Multimodule interface 1
IR7:	Multimodule interface 2

5.3.2.2 FRONT PROCESSOR

The Front Processor is a microcomputer with its own local bus, an 8 bit data bus and a 16 bit address bus, extended during global access with a 4 bit MAP-register.

The Front Processor consists of a CPU, a 64K RAM-memory with parity control, a MP^2 Bus interface, a timer, a DMA controller and some I/O circuits.

The CPU is a Z80A on which following must be observed:

RESET: is activated by the Power Control Logic when the power supplies are below the defined levels, and is kepts active for about 40 msec. after the levels have been re-established. Reset may be supplied to, or activated from the Multibus.

NMI: is activated by a parity error, by time out during MP^2-bus access, by a watchdog alarm, or by Power Fail interrupt. NMI/ to front and main processor are ored together.

The Timer-circuit (Z80-CTC) is configured as the timer and interrupt controller as follows:

CH0:	Interrupt from DMA controller (End of Processing Signal)
CH1:	Floppy Disk interrupt
CH2:	Software Timer (programmable to provide interrupt when required).
CH3:	Interrupt from main processor or Multibus (selectable by a jumper). In addition interrupts on this channel may come from none, some or all of the 3 Multimodule interfaces (selectable by 3 jumpers).

The **Direct Memory Access Controller** used with the Z80A processor is the Am9517A chip (made by Advanced Micro Devices). It is configured with the following input:

CH0:	Floppy disk controller
CH1:	Not used as input
CH2:	Z80-SIO channel A.
CH3:	Z80-SIO channel B.

CH0 and CH1 are also used during memory to memory transfers.

For correct timing of DMA transfers, the following conditions must be specified when programming the DMA controller:

1. Normal timing
2. Late write
3. DREQ active high
4. DACK active low

The **line communication interface** (Z80-SIO chip) implements two serial interfaces with V24/V28 drivers and receivers.

Further specification of the interfaces is found in section 5.3.3.4. The two interfaces are known as Channel A and Channel B. Channel A has more features than Channel B.

The Z80-SIO may be serviced by one DMA channel per serial interface, or it could be serviced in an interrupt or polling scheme. Which scheme is selected depends on software.

The **Programmable Interval Timer** used in the front processor section is an Intel 8253. Its outputs are as follows:

Channel 0: Used by software to interrupt the main processor after a user-specified time interval (implemented via input line 2 of the 8259A Interrupt Controller).

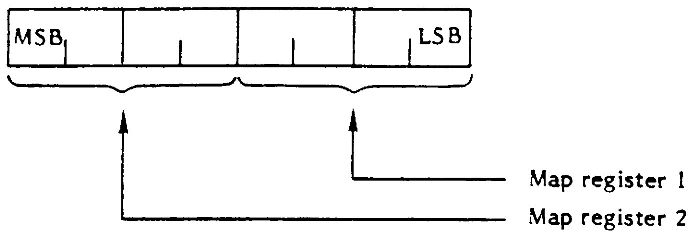
This channel is also used for hardware timeouts on the MP^2 bus (in the event of a transfer on the bus failing to complete correctly).

Channel 1:	Baudrate generator for Channel A. (The fancy RS232 interface)
Channel 2:	Baudrate generator for Channel B:. (The simpler RS232 interface)

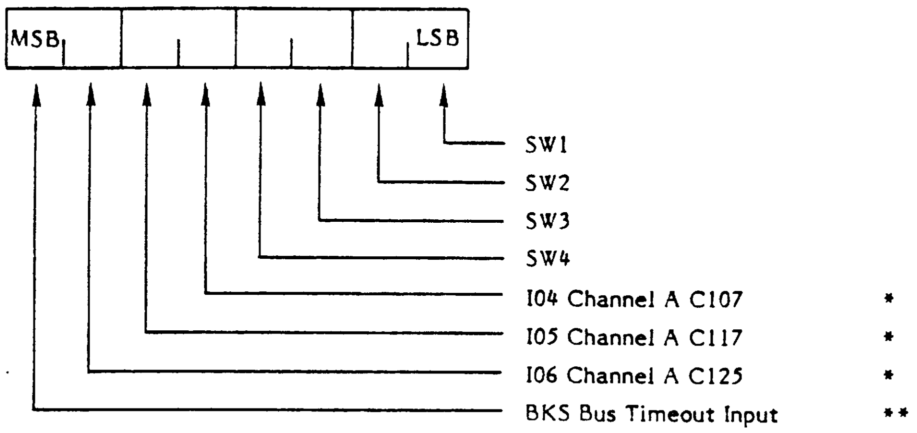
Input frequency to the PIT is 1.25 MHz.

Parallel Input Output: To extend the availability of control lines for the SIO, a PIO (i8255) and an output register have been implemented. The parallel in-outputs of the PIO are assigned as follows:

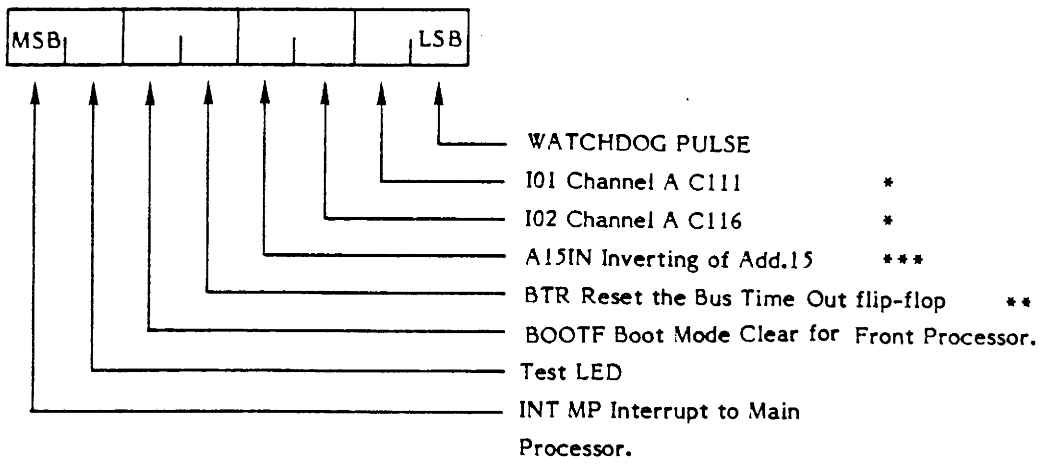
PORT A
(output)
(address 08 Hex)



PORT B
(Input)
(address 09 Hex)



PORT C
(bit 0-7: output)
(address 0A Hex)


MSB
LSB
WATCHDOG PULSE
I01 Channel A C111 *
I02 Channel A C116 *
A15IN Inverting of Add.15 ***
BTR Reset the Bus Time Out flip-flop **
BOOTF Boot Mode Clear for Front Processor.
Test LED
INT MP Interrupt to Main Processor.

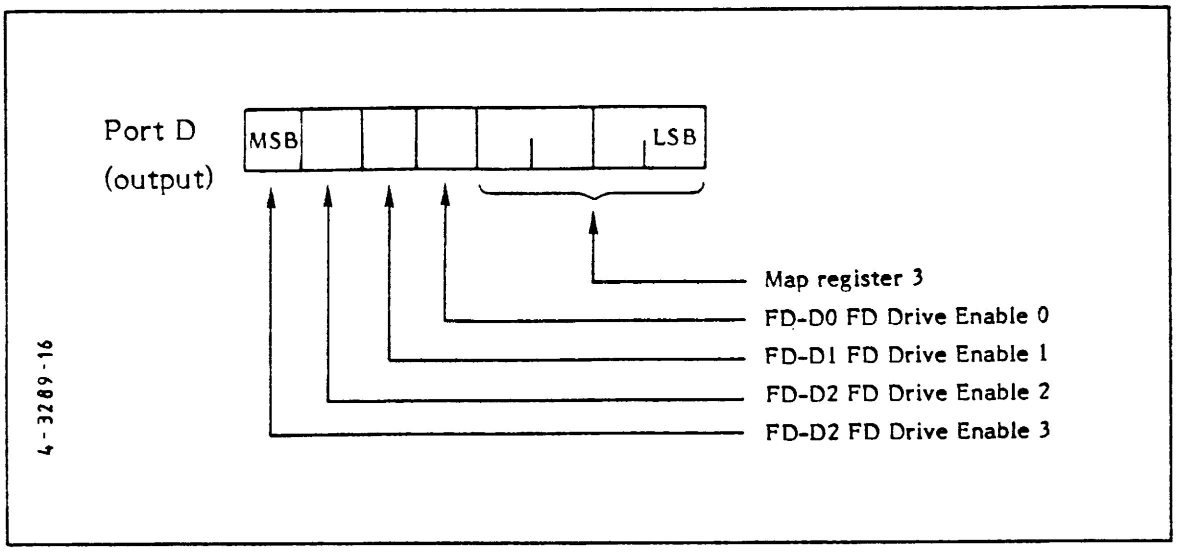
4-3289-15

* To extended line communication I/F

** The bus time out signal BKS (active high) indicates a bus time out and NMI. The BKS and NMI signals are reset and disabled by pulling BTR low and enabled when BTR are set high again.

*** This bit is used to invert address line 15 when the Front Processor accesses the MP^2 bus. When the bit is "0" then address 15 (hex) on the front processor is inverted; if "1" then address 15 is not inverted. See in section 5.3.3.

Port D of the PIO is assigned as follows:



To extend the area addressed directly by Front Processor and DMA, three MAP registers are implemented. The control of these registers is rather complicated and must be executed under semaphore protection by a general operating sytem. (See section 5.3.4.1).

FLOPPY DISK CONTROLLER

A general floppy disk controller (Western Digitals 1793 chip) has been implemented for accessing one or two double density, double sided, 5 1/4" floppy disk drives.

The controller is connected to a DMA channel and to an interrupt-input of the Front Processor which means that data transfers are normally controlled by this processor.

INTERRUPT PRIORITY

The Z80-CTC and Z80-SIO are connected using a daisy chain interrupt scheme whereby the Z80-CTC has highest priority.

5.3.2.3 MULTIBUS INTERFACE - MODUS OPERANDI

The Multibus provides a channel for the communication of data between the devices connected to it. There are 8 data lines, thus only one byte can be "in motion" at any instant in time. It follows that there must be some way of ensuring that several devices do not try to send data along the bus simultaneously. The way this is done in the CR8 is as follows:-

The MP^2 board acts as Master on the bus and all other devices (with one or two exceptions described later) act as Slaves. This means that the MP^2 initiates all data transfers (thus preventing simultaneous use of the bus by different devices).

The data transfer is effected as follws: The MP^2 sets the 20 address lines of the Multibus to the (predefined) address of the Slave device with whom it wants to communicate. Every device on the Multibus contains an address decoder - that is, circuitry which can detect whether the address on the address lines corresponds with its own address.(To be exact, each device has a range of addresses). No two devices have the same address, of course, so the required slave device will then respond by either sending or accepting the data on the 8 data lines.

Now for the one or two exceptions alluded to above. These are the Winchester Controller, any extra MP^2 cards, or possibly a user's own Multibus-compatible card(s). These devices can act as "Intelligent Slaves", in other words, as additional Masters on the bus. In this case, although at one time there is only ever one Master, at different times different cards may act as Master. Normally this is decided on a simple so-called Serial priority scheme. In this case the competing Masters have pre-assigned priorities; an aspiring Master of low priority can only be Master if all possible Masters of higher priorities do not require the bus at that time. The other way of resolving this conflict is by means of a Parallel priority scheme, whereby the bus is given to the first device that requests it after it becomes available. Additional hardware is required for this Parallel scheme: the Workstation CR8 includes this but the Cabinet Model does not.

5.3.2.4 MULTIMODULE INTERFACE

The MP^2 board includes three connectors for interfacing standard iSBX modules or special purpose modules. Each of the three connectors is addressed in a separate I/O area of 16 consecutive addresses.

The Interrupt request lines of the Multimodule sockets are connected to main processor but the DMA-request lines have no connection on the MP^2 board.

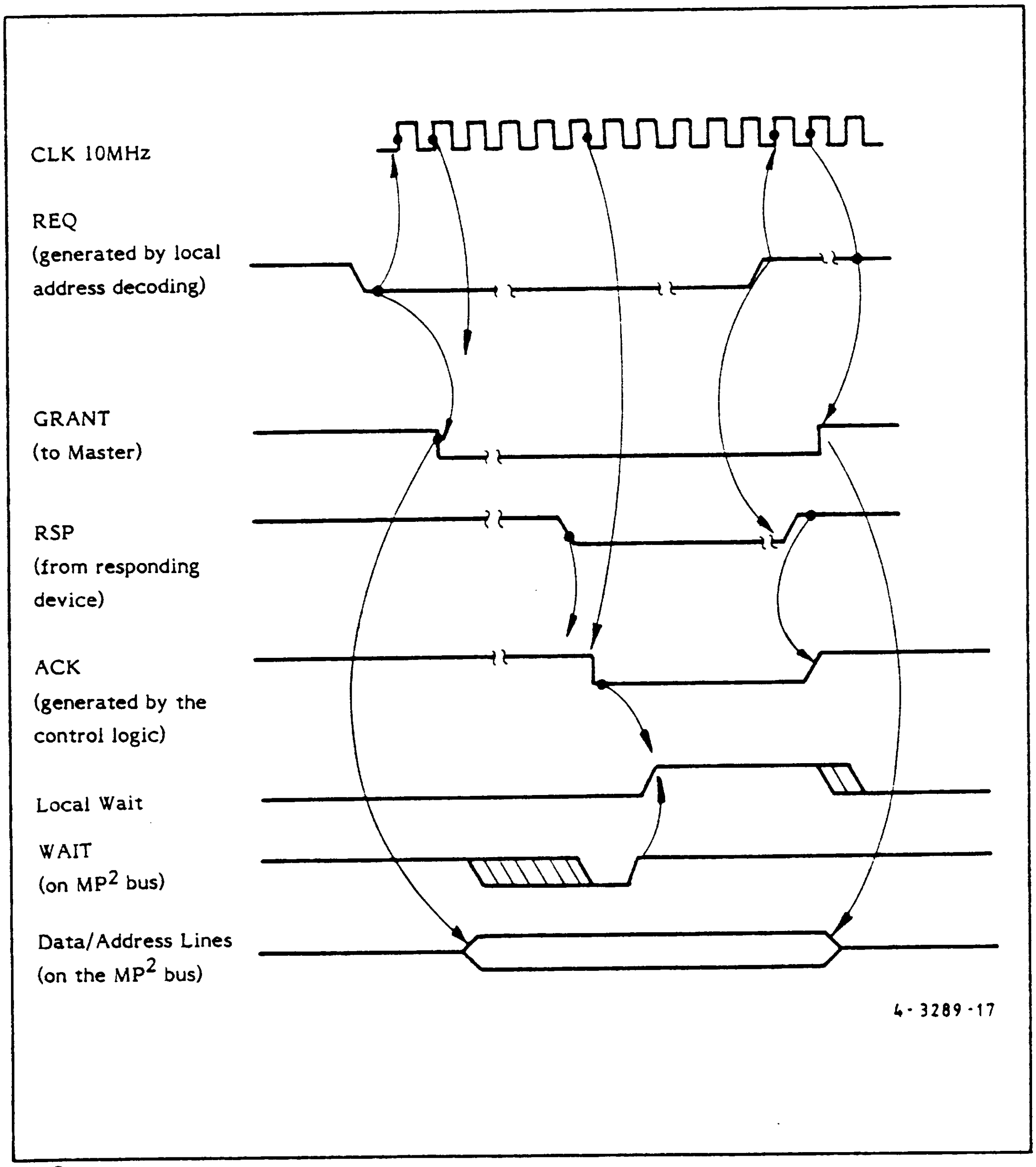
5.3.2.5 MP^2 BUS ARBITER & CONTROL LOGIC

The MP^2-bus includes an 8-bit databus, 20-bit address bus and a control bus.

The interfaces to the bus are of two types: slave interfaces (multimodule I/F), and the master/slave interfaces (Multibus I/F, Main Processor, Front Processor).

Seen from the Bus Arbiter and Control Logic there is no difference between the three master/slave interfaces. Arbitration is carried out by a state controller which performs a rotating access-priority scheme using the functions request/-grant/release.

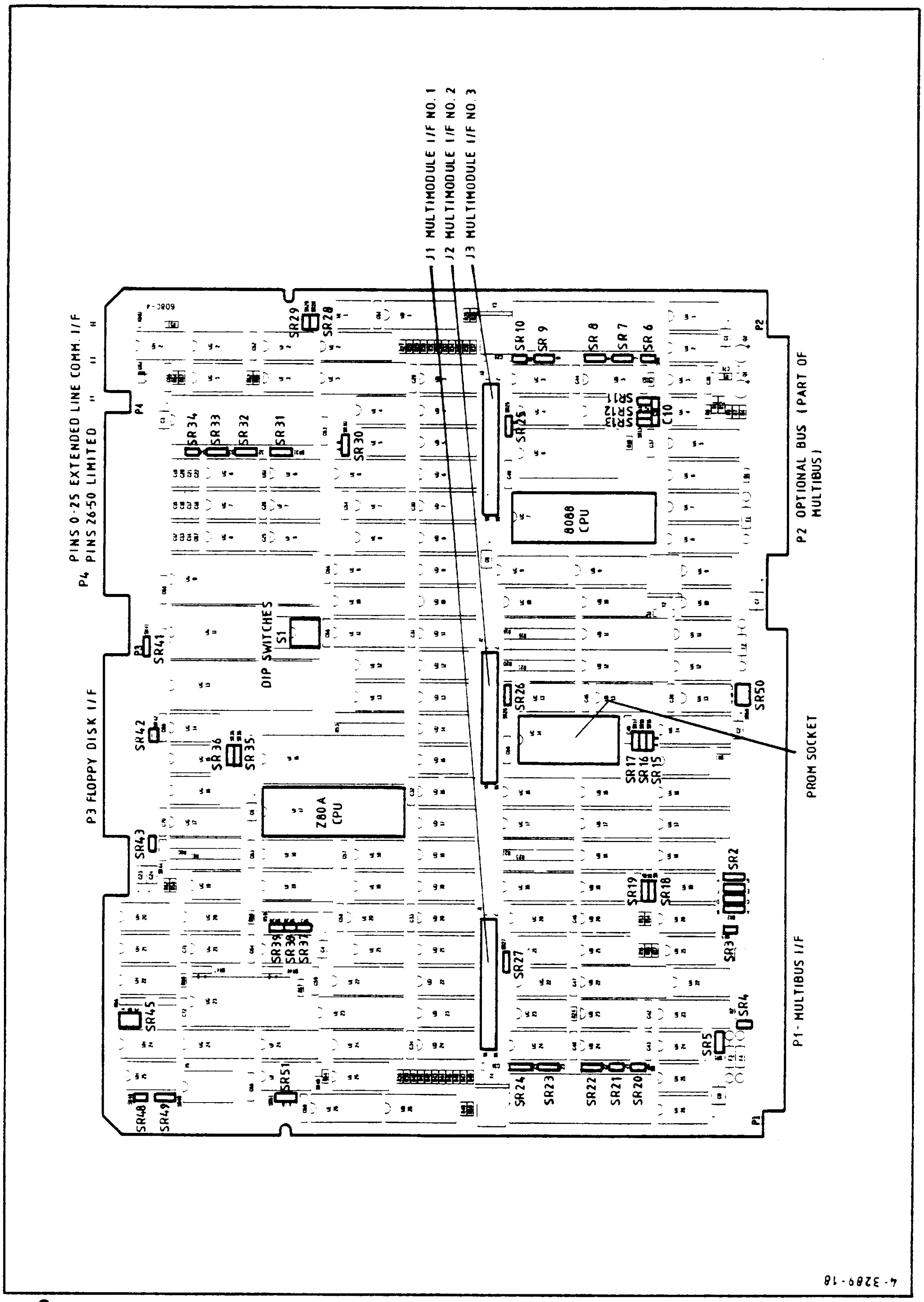
A timing diagram for MP^2 bus access is provided overleaf.


CLK 10MHz
REQ (generated by local address decoding)
GRANT (to Master)
RSP (from responding device)
ACK (generated by the control logic)
Local Wait
WAIT (on MP^2 bus)
Data/Address Lines (on the MP^2 bus)
4-3289-17

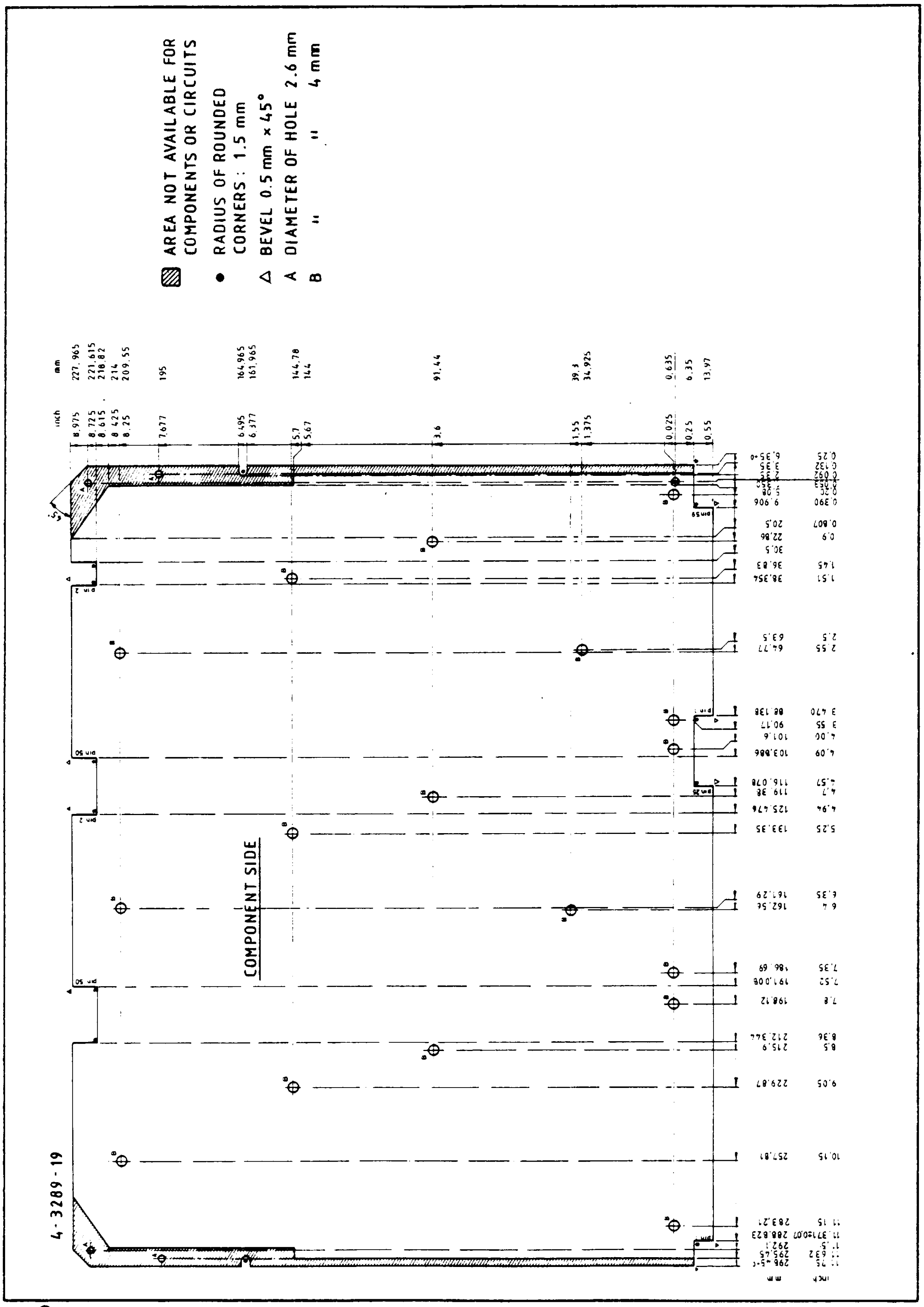
MP^2 BUS ACCESS TIMING

5.3.2.6 MECHANICAL LAYOUT

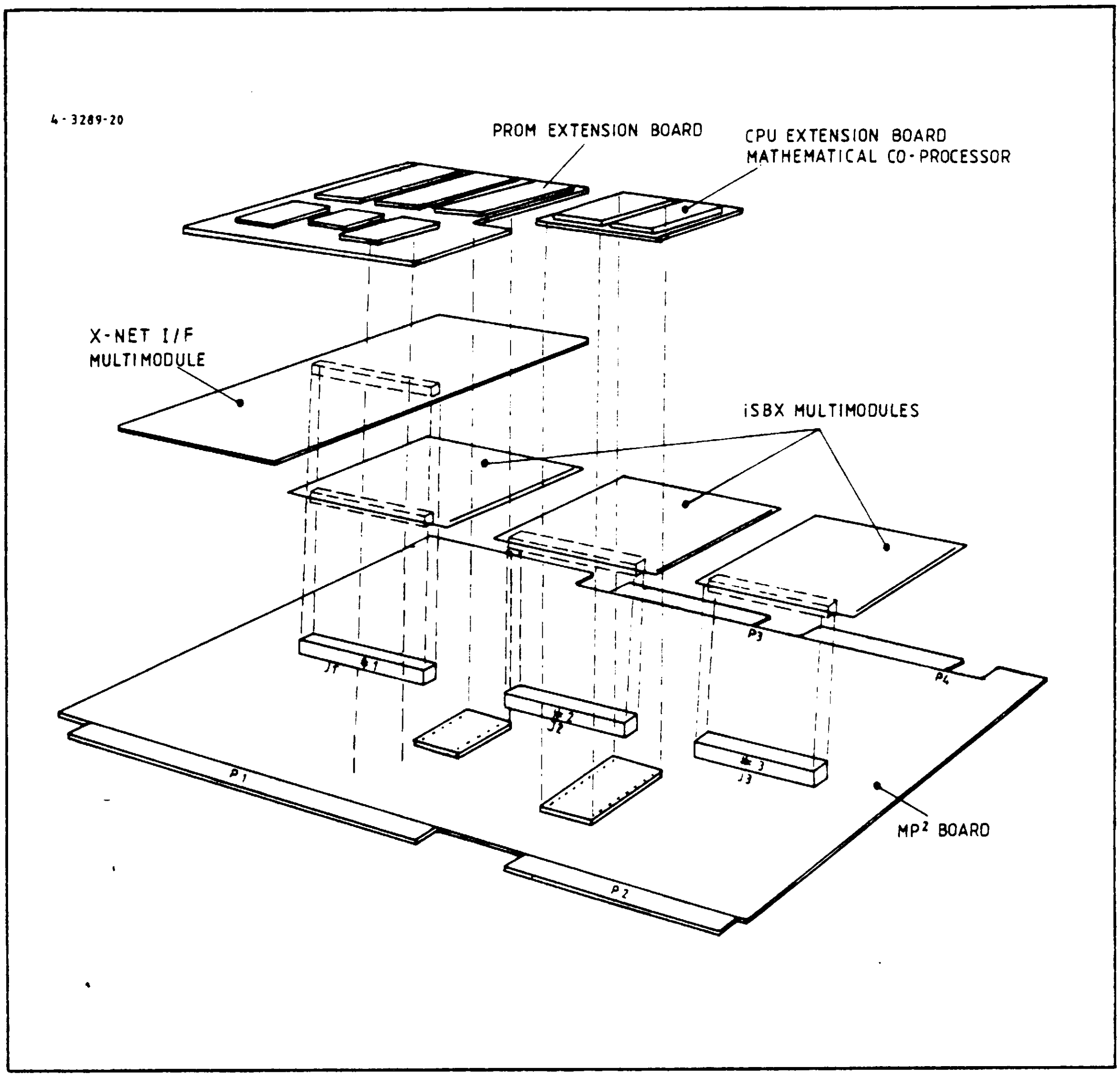
The MP^2 resides on a single card whose dimensions are as specified by the Multibus standard. The width of the board (between the tip of P1 and the tip of P3) is 24.2 cms. The positions of jumpers, connectors and some of the components are shown overleaf. Following that is a diagram illustrating expansion options on the card.



MP^2 BOARD LAYOUT



MP^2 PRINTED CIRCUIT BOARD DIMENSIONS



ON-BOARD EXPANSION OPTIONS

5.3.3 INTERFACE SPECIFICATIONS

The MP^2 interfaces to external devices via the edge connectors P1, P2, P3, P4, and the on-board connectors J1, J2, J3. In addition, for special purposes, access via the CPU socket and the Boot-load Prom-socket is possible.

NB. Please note that a slash (/) after a signal mnemonic or a bar over it (e.g. $\overline{\text{WPRT}}$) means that the signal is active when low, similarly the lack of such notation implies the signal is active when high.

5.3.3.1 MULTIBUS INTERFACE

The Multibus interface is the flexible bus structure used to interface the family of SBC boards, which include 8- and 16-bit single board computer, memory expansion boards, digital and analogue I/O boards and peripheral controllers. It supports direct addressability up to one megabyte through 20-bit addresses and 8-and 16-bit data transfers.

The bus structure is built upon the master-slave concept where the master device in the system takes control of the Multibus interface and the slave device, upon decoding its address, acts upon the command provided by the master. This handshake between master and slave devices allows modules of different speeds to use the Multibus interface and allows data rates of up to five million byte transfers per second.

Another important Multibus feature is the ability to connect multiple master modules for multiprocessing configurations. The Multibus interface provides control signals for connecting multiple masters either in a daisy-chain priority fashion or in parallel. With this latter arrangement, up to sixteen masters may share Multibus resources.

The Multibus Interface of the MP^2 is a limited Multibus Interface as defined in the following sections.

It may be configured as the master module or as an intelligent slave module.

See also section 5.3.2.3 (pg. 5-73).

SIGNAL SPECIFICATIONS

The two tables below contain definitions of the signals which appear on the limited Multibus connectors P1 and P2.

Timing specifications are according to those recommended for Multibus interfaces.

MULTIBUS CONNECTOR (P1) SIGNAL DEFINITIONS

SIGNAL	FUNCTIONAL DESCRIPTION
ADR0/ to ADR13/ (Hex.)	Address: These 20 lines transmit the address of the memory location or I/O port to be accessed. For memory access, ADR0/(when active) enables the even byte bank (DAT0/-DAT7/) on the Multibus; i.e., ADR0/ is active for all even addresses. ADR13/ is the most significant address bit.
BCLK/	Bus Clock. Used to synchronize the bus contention logic on all bus masters.
BPRN/	Bus Priority In. When low indicates to a particular bus master that no higher priority bus master is requesting use of the bus. BPRN/ is synchronized with BCLK/.
BPRO/	Bus Priority Out. In serial (daisy chain) priority resolution schemes, BPRO/ must be connected to the BPRN/ input of the bus master with the next lowest bus priority.
BREQ/	Bus Request. In parallel priority resolution schemes BREQ/ indicates that a particular bus master requires control of the bus for one or more data transfers. BREQ/ is synchronized with BCLK/.
BUSY/	Bus Busy. Indicates that the bus in in use and prevents all other bus masters from gaining control of the bus. BUSY/ is synchronized with BCLK/.

CBRQ/	Common Bus Request. Indicates that a bus master wants control of the bus but does not presently have control. As soon as control of the bus is obtained, the requesting bus controller raises the CBRQ/signal.
CCLK/	Constant Clock. Provides a clock signal of constant frequency for use by other system modules.
DAT0/ to DAT7/	Data. These 8 bi-directional data lines transmit and receive data to and from the addressed memory location or I/O port. DAT7/ is the most significant bit.
INIT/	Initialize. Reset the entire system to a known internal state.
INT0/ to INT7/	Interrupt Request. Non Bus vectored interrupt inputs, high to low edge triggered. Only one of the Interrupt Request lines is serviced by the MP^2. The input is selected by strap; normally INT0 is connected to the interrupt controller of the Main Processor.
IORC/	I/O Read Command. Indicates that the address of an I/O port is on the Multibus address lines and that the output of that port is to be read (placed) onto the Multibus data lines.
IOWC/	I/O Write Command. Indicates that the address of an I/O port is on the Multibus address lines and that the contents on the Multibus data lines are to be accepted by the addressed port.
MRDC/	Memory Read Command. Indicates that the address of a memory location is on the Multibus address lines and that the contents of that location are to be read (placed) on the Multibus Data Lines.
MWTC/	Memory Write Command. Indicates that the address of a memory location is on the Multibus address lines and that the contents of the Multibus data lines are to be written into that location.

XACK/	Transfer Acknowledge. Indicates that the address memory location has completed the specified read or write operation. That is, data has been placed onto or accepted from the Multibus data lines.

AUXILIARY CONNECTOR (P2) SIGNAL DEFINITIONS

SIGNAL	FUNCTIONAL DESCRIPTION
RESET/	Reset. This externally generated signal initiates a power-up sequence.
PFIN/	Power Failure Interrupt. This signal from the power supply interrupts both processors by NMI when a power failure occurs.

	PIN	(COMPONENT SIDE)		PIN	(CIRCUIT SIDE)	
		MNEMONIC	DESCRIPTION		MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal GND	2	GND	Sig GND
	3	+5V	+5Vdc	4	+5V	+5Vdc
	5	+5V	+5Vdc	6	+5V	+5Vdc
	7	+12V	+12Vdc	8	+12V	+12Vdc
	9			10		
	11	GND	Signal GND	12	GND	Signal GND
BUS CONTROLS	13	BCLK/	Bus Clock. 10MHZ	14	INIT/	Initialize
	15	BPRN/	Bus Pri.In	16	BPRO/	Bus Pri.Out
	17	BUSY/	Bus Busy	18	BREQ/	Bus Request
	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21	IORC/	I/O Read Cmd	22	IOWC/	I/O Write Cmd
	23	XACK/	XFER Acknowledge	24		
BUS CONTROLS AND ADDRESS	25			26		
	27			28	AD10/	
	29	CBRQ/	Common Bus Request	30	AD11/	Address
	31	CCLK/	Constant Clk.	32	AD12/	Bus
	33		10MHZ($\overline{\text{BCLK}}$/)	34	AD13/	
INTERRUPTS	35	INT6/		36	INT7/	
	37	INT4/	PARALLEL	38	INT5/	PARALLEL
	39	ILT2/	INTERRUPT	40	INT3/	INTERRUPT
	41	INT/0	REQUESTS	42	INT1/	REQUESTS
ADDRESS	43	ADRE/		44	ADRF/	
	45	ADRC/		46	ADRD/	
	47	ADRA/	Address	48	ADRB/	Address
	49	ADR8/	Bus	50	ADR9/	Bus
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADR0		58	ADR1/	
DATA	59			60		
	61			62		
	63			64		
	65			66		
	67	DAT6/	Data	68	DAT7/	Data
	69	DAT4/	Bus	70	DAT5/	Bus
	71	DAT2/		72	DAT3/	
	73	DAT0		74	DAT1/	
POWER SUPPLIES	75	GND	Signal GND	76	GND	Signal GND
	77			78		
	79	-12V	-12Vdc	80	-12V	-12Vdc
	81	+5V	+5Vdc	82	+5V	+5Vdc
	83	+5V	+5Vdc	84	+5V	+5Vdc
	85	GND	Signal GND	86	GND	Signal GND

PIN ASSIGNMENT OF BUS SIGNALS ON MULTIBUS BOARD CONNECTOR (P1)

PIN	(COMPONENT SIDE)		PIN	(CIRCUIT SIDE)	
	MNEMONIC	DESCRIPTION		MNEMONIC	DESCRIPTION
1	GND	Signal GND	2	GND	Signal GND
3	5VB	+5V Battery	4	5VB	+5V Battery
5			6		
7			8		
9			10		
11			12		
13			14		
15			16		
17	PFIN/	POWER FAIL Interrupt	18		
19			20		
21	GND	Signal GND	22	GND	Signal GND
23			24		
25			26		
27			28		
29			30		
31			32		
33			34		
35			36		
37			38	AUX RESET/	Reset switch
39			40		
41			42		
43			44		
45			46		
47			48		
49			50		
51			52		
53			54		
55			56		
57			58		
59			60	AUX PWR	AUX. POWER

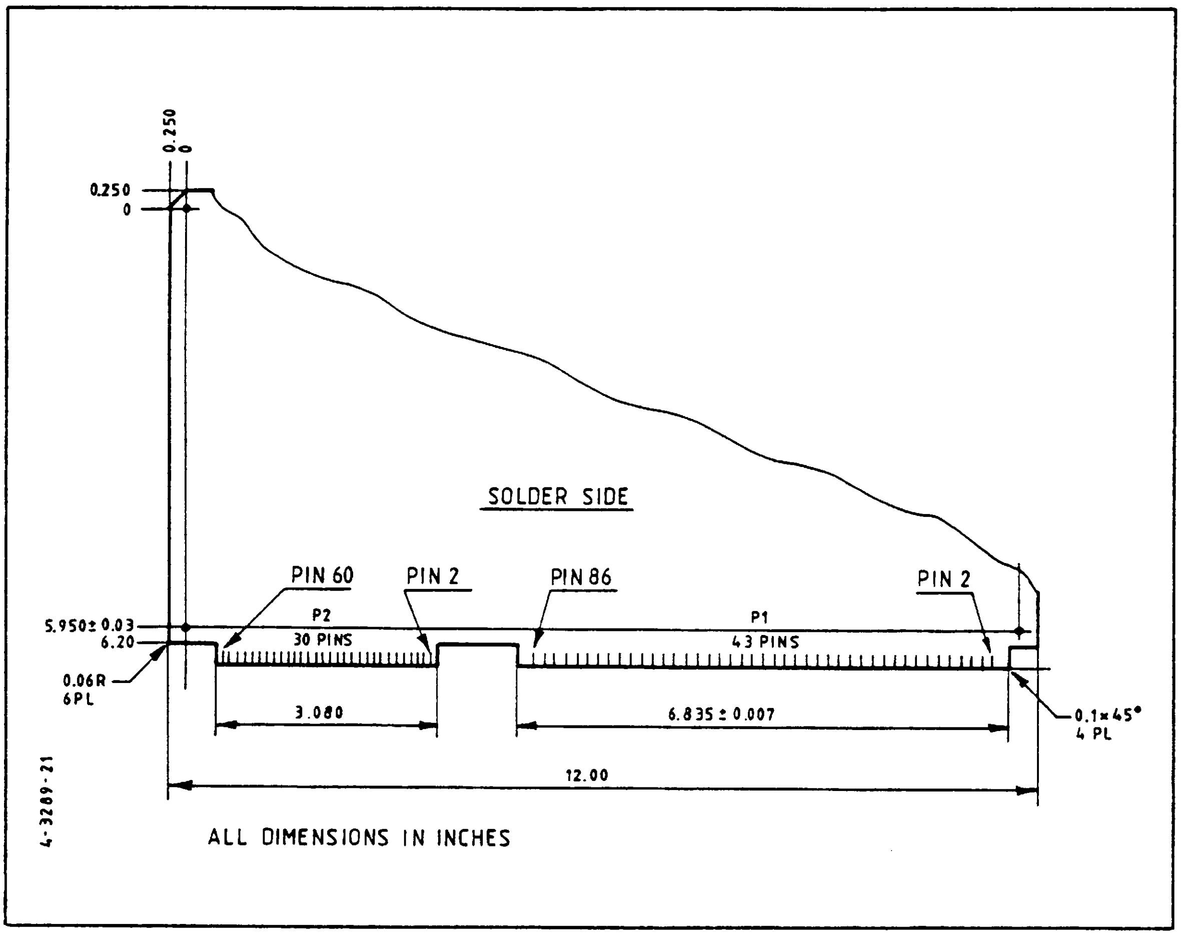
P2 CONNECTOR PIN ASSIGNMENT - OPTIONAL BUS SIGNALS

	Driver 1,3					Receiver 2,3				Termination			
Bus Signals	Location	Type	I_{OL} Min$_{ma}$	I_{OH} Min$_{\mu a}$	C_O Min$_{pf}$	Location	I_{IL} Max$_{ma}$	I_{IH} Max$_{\mu a}$	C_I Max$_{pf}$	Location	Type	R	Units
DATO-DATF/ (16 lines)	Masters and Slaves	TRI	16	-2000	300	Masters and Slaves	-0.8	125	18				
ADRO/-ADRB/ (20 lines)	Masters	TRI	16	-2000	300	Slaves	-0.8	125	18				
MRDC/.MWTC/	Masters	TRI	32	-2000	300	Slaves (Memory; memory-mapped I/O)	-2	125	18	MP ^{2}	Pullup	1	KΩ
ICRC/.IOWC/	Masters	TRI	32	-2000	300	Slaves (I/O)	-2	125	18	MP ^{2}	Pullup	1	KΩ
XACK/	Slaves	TRI	32	-2000	300	Masters	-2	125	18	MP ^{2}	Pullup	510	Ω
BCLK/	1 Place	TTL	48	-3000	300	Master	-2	125	18	Mother-board			
BREQ/	Each Master	TTL	5	-400	60	Central Priority Module	-2	50	18	Central Priority Module			
BPRO/	Each Master	TTL	5	-400	60	Next Master in Serial Priority Chain at its PRN/	-1.6	50	18				
BPRN/	Parallel: Central Priority Module Serial: Prev Masters BPRO/	TTL	5	-400	60	Master	-2	50					
BUSY/.CBRQ	All Masters	O.C.	32	-	300	All Masters	-2	50	18	MP ^{2}	Pullup	1	KΩ
INIT/	Master	O.C.	32	-	300	All	-2	50	18	MP ^{2}	Pullup	2.2	KΩ
CCLK/	1 place	TTL	48	-3000	300	Any	-2	125	18	Mother-board			

ELECTRICAL SPECIFICATIONS OF BUS DRIVERS, RECEIVERS, AND TERMINATION

MECHANICAL SPECIFICATION

This section shows the dimensions of the connectors P1 and P2.



5.3.3.2 MULTIMODULE INTERFACE

The Multimodule interface facilitates on-board expansion with Multimodule boards. The Multimodule bus is derived directly from the MP^2 bus and as such, a Multimodule board plugged into the MP^2 bus becomes an integral element of the MP^2. The physical interface between the MP^2 and the Multimodule board is a connector designed specifically for this interface. The Multimodule bus is brought out to a female connector on the MP^2 and mates with its male equivalent resident on the Multimodule board.

The MP^2 contains three Multimodule interfaces. These are limited Multimodule Interfaces and are numbered # 1, # 2 and # 3.

The user may plug in his own Multimodule devices that he has bought "off the shelf", or design his own. In addition, CR supply some, e.g. a Line Communication Interface, UPI Interface (process control), PAM Interface (process monitoring) and TDX Interface (X-Net local area network).

SIGNAL SPECIFICATIONS

The following two tables contain definitions of the signals which appear on the limited Multimodule interface connectors J1, J2 and J3.

Timing specifications are according to those recommended for Multimodules.

Signal	Functional Description
MD0-MD7	**Data** These active high bidirectional data lines transmit data between the Multimodule and MP^2.
IORD/ IOWRT/	**I/O Read** **I/O Write** The command lines are active low signals which provide the communication link between the MP^2 board and the Multimodule board. An active command line, conditioned by chip select, indicates to the Multimodule board that the address lines are valid and the Multimodule board should perform the specified operation.
MA0-MA2	**Address** These positive true input lines to the Multimodule boards are generally the least three significant bits of the I/O address. In conjunction with the command and chip select lines, they establish the I/O port address being accessed.
MCS0/ MCS1/	**Chip Select Lines** These input lines to the Multimodule board arethe result of the base board I/O decode logic. MCS/ is an active low signal and thus enables communication with the Multimodule boards.
RESET	**Reset** This input line to the Multimodule board is generated by the base board to put the Multimodule board into a known internal state.

MULTIMODULE CONNECTOR SIGNAL DEFINITIONS

Signal	Functional Description
MWAIT/ MPST/	**Wait** **Multimodule Present** These output signals form the Multimodule board control the state of the system. Active MWAIT/(Active low) will put the CPU on the board into a wait state providing additional time for the Multi-module board to perform the requested operation. MWAIT /must be generated from address (address plus chip select) information only. If MWAIT/is driven active due to a glitch on the CS line during address transitions, MWAIT/must be driven inactive in less than 75 ns. The Multimodule board present (MPST/) is an active low signal (tied to signal ground) that informs the base board I/O decode logic that an Multimodule board has been installed.
MINTR0-1	**Interrupt lines** These active high output lines from the Multimodule board are used to make interrupt requests to the base board. Only one is connected to interrupt controller of the Main Processor. Selectable by straps. The three selected interrupts from the Multimodules are gated together and connected to CTC channel 3.
MCLK	**Clock lines** This 10MHz input to the Multimodule board is a timing signal
AUX-PWR	**Aux power** This is a power input for special purposes

MULTIMODULE CONNECTOR SIGNAL DEFINITIONS (CONTINUED)

Pin	Mnemonic	Description	Pin	Mnemonic	Description
35	GND	Signal Ground	36	+5V	-+5 volt
33	MD0	MDATA Bit 0	34		
31	MD1	MDATA Bit 1	32		
29	MD2	MDATA Bit 2	30		
27	MD3	MDATA Bit 3	28		
25	MD4	MDATA Bit 4	26		
23	MD5	MDATA Bit 5	24		
21	MD6	MDATA Bit 6	22	MCS0/	M Chip Select 0
19	MD7	MDATA Bit 7	20	MCS1/	M Chip Select 1
17	GND	Signal Gnd	18	+5V	+5 Volts
15	IORD/	I/O Read Cmd	16	MWAIT/	M Wait
13	IOWRT/	I/O Write Cmd	14	MINTR0	M Interrupt 0
11	MA0	M Address 0	12	MINTR1	M Interrupt 1
9	MA1	M Address 1	10	AUX.PWR	AUX. POWER
7	MA2	M Address 2	8	MPST/	iSBX Multimodule Board Present
5	RESET	Reset	6	MCLK	M Clock
3	GND	Signal Gnd	4	+5V	+5 Volts
1	+12V	+12 Volts	2	−12V	−12 Volts

PIN ASSIGNMENT OF BUS SIGNALS ON MULTIMODULE CONNECTOR J1,J2,J3

ELECTRICAL SPECIFICATIONS

Multimodules interfacing to MP^2 must fulfill the following electrical specifications.

Output[1]

Bus Signal Name	Type[2] Drive	I_{OL} Max −Min (mA)	@ Volts (V_{OL} Max)	I_{OH} Max −Min (μA)	@ Volts (V_{OH} Min)	Co (Min) (pf)
MD0-MD7	TRI	1.6	0.5	−200	2.4	130
MINTR0-1	TTL	2.0	0.5	−100	2.4	40
MDRQT	TTL	1.6	0.5	− 50	2.4	40
MWAIT/	TTL	1.6	0.5	− 50	2.4	40
OPT1-2	TTL	1.6	0.5	− 50	2.4	40
MPST/	TTL	Note 3				

Input[1]

Bus Signal Name	Type[2] Receiver	I_{IL} Max (mA)	@ V_{IN} Max (volts)	I_{IH} Max (μA)	@ V_{IN} Max (volts)	Ci Max (pf)
MD0-MD7	TRI	−0.5	0.8	70	2.0	40
MA0-MA2	TTL	−0.5	0.8	70	2.0	40
MCS0/-MCS1/	TTL	−4.0	0.8	100	2.0	40
MRESET	TTL	−2.1	0.8	100	2.0	40
MDACK/	TTL	−1.0	0.8	100	2.0	40
IORD/ IOWRT/	TTL	−1.0	0.8	100	2.0	40
MCLK	TTL	−2.4	0.8	100	2.0	40
OPT1-OPT2	TTL	−2.0	0.8	100	2.0	40

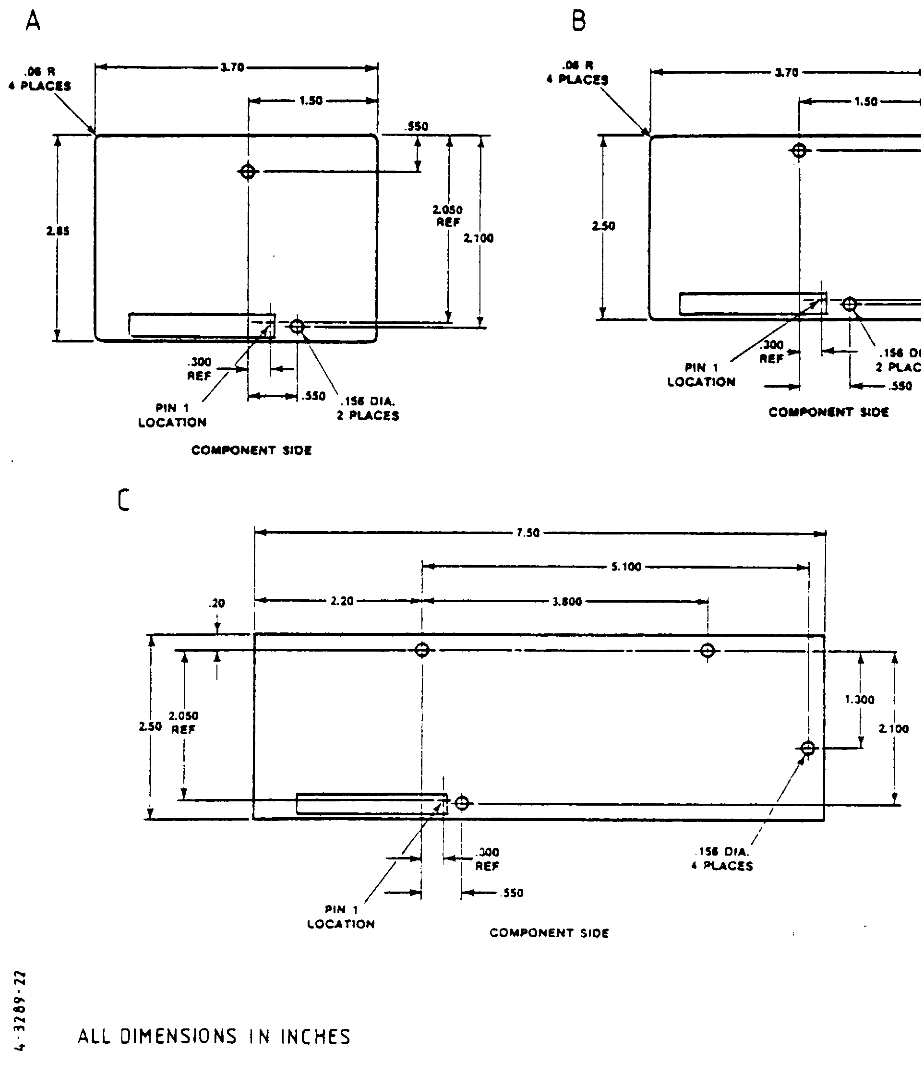
NOTES:
1. Per iSBX Multimodule I/O board.
2. TTL = standard totem pole output. TRI = Three-state.
3. iSBX Multimodule board must connect this signal to ground.

MECHANICAL SPECIFICATIONS

This sections specifies the mechanical layout for the different Multimodules.

MECHANICAL LAYOUT FOR STANDARD MULTIMODULES

The outlined connectors are placed on the solder side of the PCB.



5.3.3.3 FLOPPY DISK INTERFACE

This interface is wired to the 50-pin edge connector, P3 on the MP^2 board. It connects to one or two double-sided double-density 5 1/4" floppy diskette drives. Both can be connected to the same piece of 50-core ribbon cable by using 3 connectors in all on the cable (including the edge connector for the MP^2 board).

Note that one Select output is used for the Motor Drive On signal (which is common to both drives if there are two), another Select output is used for Head-select, and one or two are for drive(s) select.

The signals on the edge connector are described in the two tables below.

Signal	Functional Description
READY	**READY** This input indicates disk readiness and is sampled for a logic high before Read or Write commands are performed.
TR0/	**TRACK ZERO** This input informs that the Read-Write Head is positioned over Track 00 when a logic low.
INDEX/	**INDEX HOLE** Input, when low for a minimum of 10 microseconds informs the controller when an index mark is encountered on the diskette.
WPRT/	**WRITE PROTECTED** This interface signal is provided by the drive to give the system an indication when a Write Protected Diskette is installed.
READ DATA/	**READ DATA** The data input signal directly from the drive. This input shall be a pulse for each recorded flux transition.

FLOPPY DISK INTERFACE SIGNALS

Signal	Functional Description
WRITE DATA/	**WRITE DATA** These interface lines provide the data to be written on the diskette.
DIR/	**DIRECTION** This interface line is a control signal which defines direction of motion the R/W head will take when the step line is pulsed.
HLD/	**HEAD LOAD** The HDL output controls the loading of the Read-Write head against the media.
WG/	**WRITE GATE** This output is made valid when writing is to be performed on the diskette.
STEP/	**STEP** The step output contains a pulse for each step of the head's movement. The R/W head has to move with the direction of motion as defined by the Direction Select line (DIR).
DR0/-DR3/	**Drive Select Lines** Activates, when active (low), one of four Floppy Disc Drives.
HEAD SELECT	Side select when using double sided Floppy Disk. If using 1793 chip, DR2 can be strapped to Head Select by jumper. The 1797 chip has an output pin for Head Select.

FLOPPY DISK INTERFACE SIGNALS (CONTINUED)

PIN	DESCRIPTION	PIN	DESCRIPTION
		11	GND
22	READY DR3/	15	GND
42	TR0	17	GND
20	INDEX	19	GND
44	WPRT	21	GND
46	READ DATA	23	GND
38	WRITE DATA	25	GND
18	HLD	27	GND
40	WG	29	GND
34	DIR	31	GND
36	STEP	33	GND
26	DR0/	35	GND
28	DR1/	37	GND
30	DR2/	39	GND
32	DR3/	41	GND
24	INDEX	43	GND
		45	GND
48 }	Head Select DR2/	47	GND
14 }		49	GND

PIN ASSIGNMENT OF SIGNALS ON FLOPPY DISC INTERFACE

INPUTS				
Signal Name	Receivers	Pullup resistance	Iil (max)	Vil (max)
SEPC READY TRQ INDEX WRPT READ DATA	TTL (7414)	150 ohms	-1.2mA	0.6V

OUTPUTS			
Signal Name	Drivers	Iil (max)	Vil (max)
WRITE DATA DIR HLD WG STEP DR0-DR3	TTL (OC) (7438)	48mA	0.4V

ELECTRICAL SPECIFICATIONS OF FLOPPY DISC INTERFACE

MECHANICAL SPECIFICATION

Mechanical lay out and dimensions are shown in section 5.3.2.6.

5.3.3.4 LINE COMMUNICATION INTERFACES

These two interfaces are connected to the edge connector P4 in such a way that when a 50 wire flatcable and connector is fitted to P4, and that cable is split between the wires 25 and 26, it may be mounted with two 25 pin Cannon connectors. The connectors will then be configured in accordance with CCITT V24 recommendation for Data Terminal Equipment (DTE) devices. Further mechanical details appear on page 5-98.

These interfaces are implemented by the SIO chip in the front processor, with some additional signals from the PIO (see pgs 5-69 to 5-72).

V24 CIRCUIT NO.	DRIVER/ RECEIVER	CONNECTION TO/FROM	DESCRIPTION
103	DRIVER	SIO	TRANSMITTED DATA
104	RECEIVER	SIO	RECEIVED DATA
105	DRIVER	SIO	REQUEST TO SEND
106	RECEIVER	SIO	CLEAR TO SEND
107	RECEIVER	PIO	DATA SET READY
108	DRIVER	SIO	DATA TERMINAL READY
109	RECEIVER	SIO	DATA CARRIER DETECT
111	DRIVER	PIO	DATA SIGNALLING RATE SELECT
113	DRIVER	TIMER	TRANSMITTER CLOCK
114	RECEIVER	SIO	TRANSMITTER CLOCK
115	RECEIVER	SIO	RECEIVER CLOCK
116	DRIVER	PIO	SELECT STANDBY
117	RECEIVER	PIO	STANDBY INDICATOR
125	RECEIVER	PIO	CALL INDICATOR

CHANNEL A SIGNAL DEFINITION
(EXTENDED LINE COMMUNICATION INTERFACE)

V24 CIRCUIT NO.	DRIVER/ RECEIVER	CONNECTION TO/FROM	DESCRIPTION
103	DRIVER	SIO	TRANSMITTED DATA
104	RECEIVER	SIO	RECEIVED DATA
105	DRIVER	SIO	REQUEST TO SEND
106	RECEIVER	SIO	CLEAR TO SEND
108	DRIVER	SIO	DATA TERMINAL READY
109	RECEIVER	SIO	DATA CARRIER DETECT
113	DRIVER	TIMER	TRANSMITTER CLOCK
115	RECEIVER	SIO	RECEIVER CLOCK

CHANNEL B SIGNAL DEFINITION
(LIMITED LINE COMMUNICATION INTERFACE)

V24 Signal	Cannon Connector	P4 Connector		Cannon Connector	V24 Signal	Channel
	1	1	2	14		
C-103	2	3	4	15	C-114	
C-104	3	5	6	16		
C-105	4	7	8	17	C-115	
C-106	5	9	10	18		
C-107	6	11	12	19		
C-102	7	13	14	20	C-108	A
C-109	8	15	16	21		
	9	17	18	22	C-125	
	10	19	20	23		
C-111	11	21	22	24	C-116	
	12	23	24	25	C-117	
	13	25				
			26	1		
	14	27	28	2	C-103	
C-113	15	29	30	3	C-104	
	16	31	32	4	C-105	
C-115	17	33	34	5	C-106	
	18	35	36	6		B
	19	37	38	7	C-102	
C-108	20	39	40	8	C-109	
	21	41	42	9		
	22	43	44	10		
	23	45	46	11		
	24	47	48	12		
	25	49	50	13		

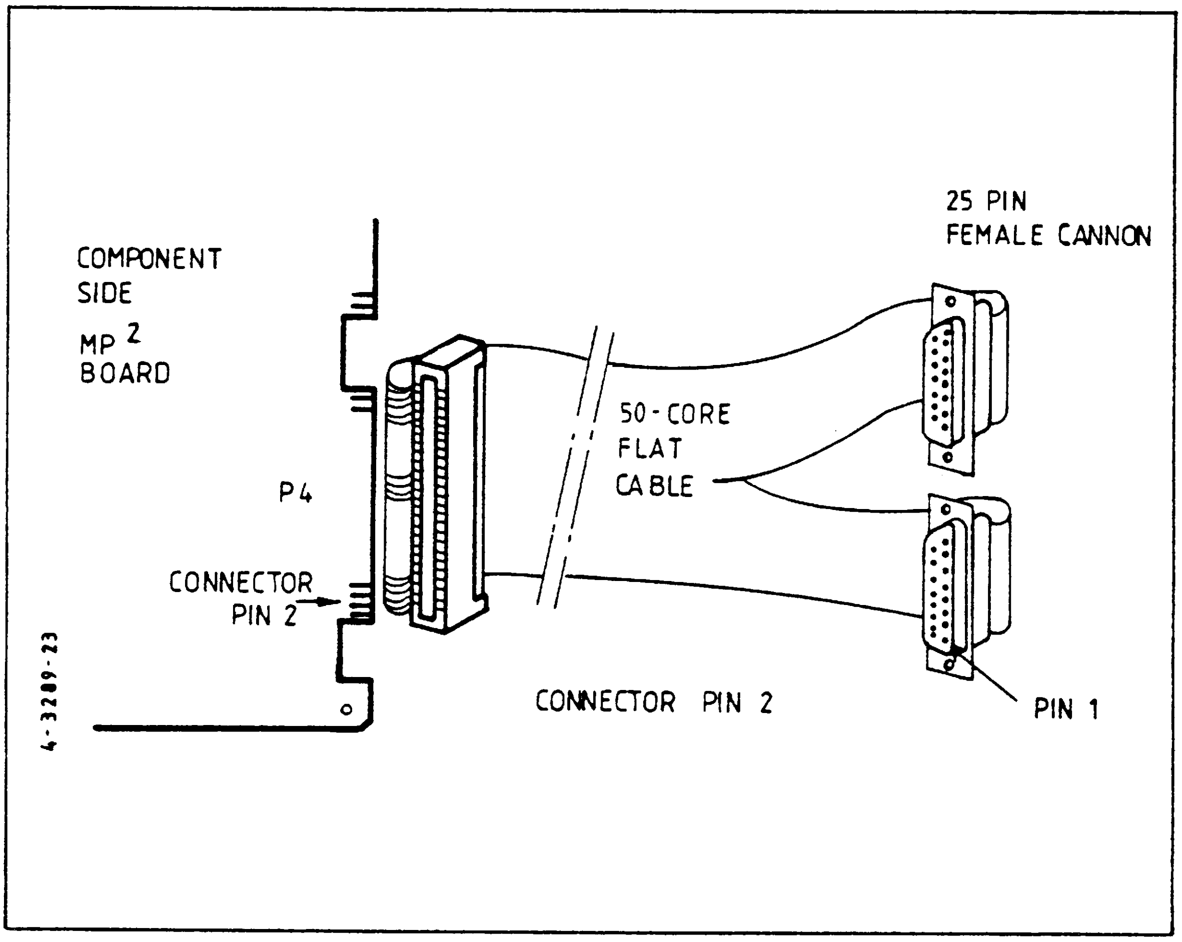
PIN ASSIGNMENT ON THE CANNON CONNECTORS AND MP^2 CARD FOR THE LINE COMMUNICATION INTERFACE

ELECTRICAL SPECIFICATIONS

The electrical specifications are according to CCITT-V28 recommendation.

The pin numbering for the flat cable connector and the edge connector P4 is shwn below. The cable should be split between wires 25 and 26.

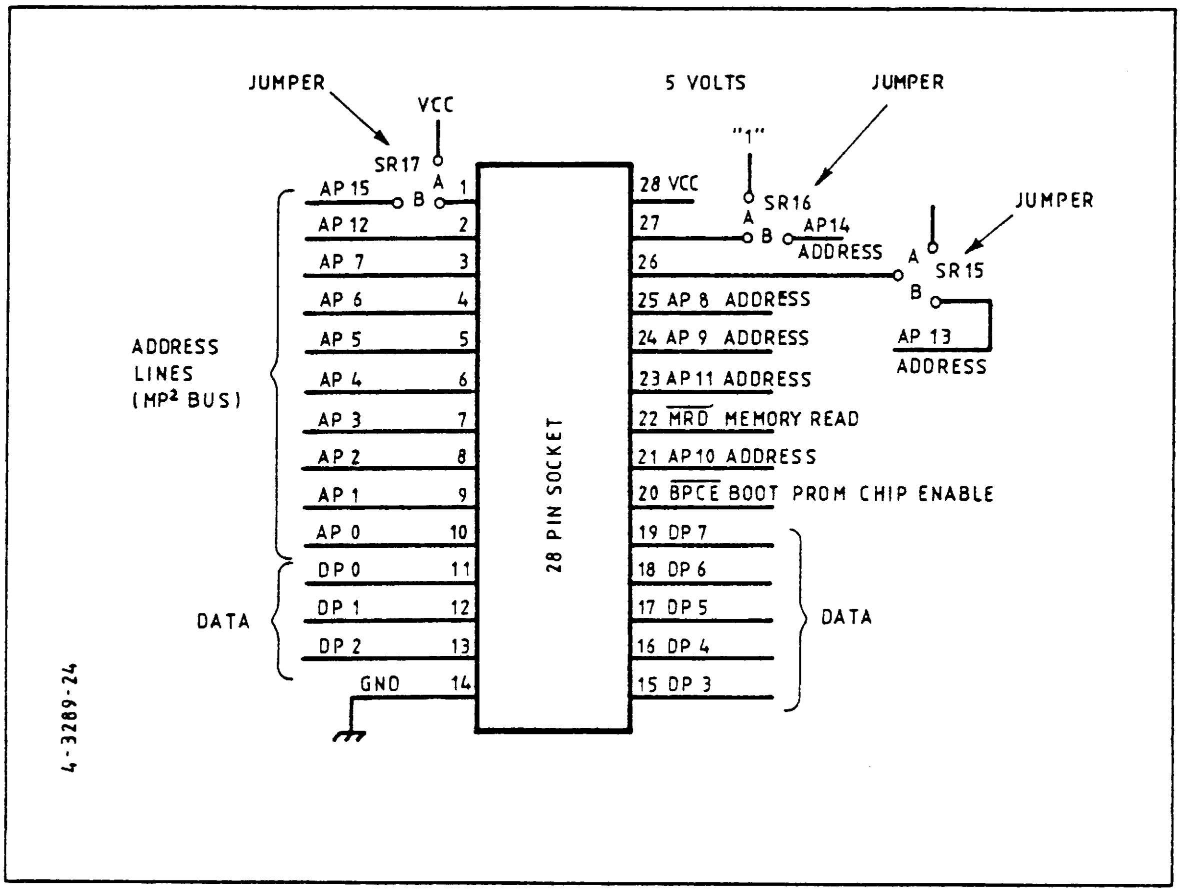
Cannon Connector	Cable wire	P4-pin
A, pin 1	1	1
B, pin 1	26	26



SERIAL INTERFACE PIN NUMBERING

5.3.3.5 BOOT LOAD PROM

The PROM area of the MP^2 may be extended by replacing the PROM with a small PROM-board with a socket fitting directly into the PROM socket. In this way the Boot Load PROM-area may be extended up to 64K bytes. All support functions for the PROM, such as address decoding and buffering, are included on the board.



BOOT PROM SIGNAL SPECIFICATION

5.3.3.6 MATHEMATICS CO-PROCESSOR

By replacing the Main processor CPU with a small PCB containing a CPU and a mathematics co-processor a very high performance is obtained in number-crunching applications.

All circuits needed for this extension are prepared on the MP^2 or are situated on the PCB which fits directly into the CPU socket.

Intel offers a standard Co-processor Multimodule (iSBC 337) containing an 8088 CPU and an 8087 Numeric Data Processor.
The Interrupt from the SBC 337 (referred to as pins P2-1 and P2-2) is not connected to the MP^2 board. Full details of the use of this processor can be obtained from Intel's iSBC 337 Multimodule Numeric Data Processor Hardware Reference Manual, order no. 142887-001.

5.3.4 PROGRAMMING AND OPERATIONAL CONSIDERATIONS FOR THE MP^2

Hardware wise, there are nearly no operational restrictions, but software wise, a well designed grant technique must be observed when accessing all I/O-devices, especially the DMA and the MAP Registers.

To ensure a safe hardware system, the MP^2 Bus interfaces are equipped with a time out function which means that no MP^2 Bus access can take more than 50 microseconds. Furthermore, a Watchdog function is implemented in order to ensure correct interaction between the processors. This means that the line called Watchdog Pulse, on Port C of the PIO (see pg. 5-71) has to be pulsed continously, i.e. set by Front Processor and reset by the Main Processor about once every 250 msec.

The Bus Timeout function comes into force in two situations - a bus lock that cannot be resolved, and the addressing of a missing, restricted or faulty module. If the MP^2 is running CP/M or MP/M, this will result in a system crash; a reboot will be necesary.

Certain bus lock situations can be resolved however. A bus lock is the situation where two CPU's address each other's RAM simultaneously and they both wait for each other to reply. In cases where one of the processors involved is the main processor on an MP^2 card, then the conflict can be resolved. (The other processor could be the front processor on the same MP^2 - the most likely case since normally CR8s have only one MP^2 card - or it could be either processor on another MP^2 or on some other connected Multibus card.)

In such cases, the conflict is resolved as follows. This situation is detected by the Bus Arbitration & Control circuitry (see block diagram on pg. 5-66) which then disables the driver logic which interfaces the 8088 CPU to its local bus (represented as a small box with two arrows in diagram). Once this has been done it is no longer necessary for the 8088 to "give permission" for access to its RAM. Thus the Z80 accesses the 8088's RAM. After the access is complete, the 8088 driver logic comes back to life, then access between 8088 and Z80's RAM takes place in the normal way.

In cases of Bus lock that cannot be resolved as just described, the timeout mechanism is brought into effect in the following way:

The Bus Timeout logic will give an NMI/ to both processors, thus disabling their wait states (possibly only one will be waiting). (Incidentally, NMI/ means non-maskable interrupt (active low signal), in other words an interrupt which will take place immediately in all situations.) Furthermore the NMI will cause the bus timeout function itself to be disabled.

Next, the processors must read the status bit BKS on the PIO, port B bit 7, in order to discover what sort of NMI had occured. (The NMI could have been caused by the watchdog process or a memory parity error.) So in this case, a bus timeout was the cause; this will be indicated by the BKS status bit being "high". Now the software knows what has happened and can send an appropriate message to the console in due course. First of all, within one CLK INT timeperiod from the PIT timer chip, the bus timeout function must be re-enabled.

This is done by pulling bit 4 on port C (BTR/) low and then high again. If BTR/ (Bus Time Reset) is kept low, the bus timeout function is continually disabled. Now there is time to inform the user what has happened. Processing can then proceed as normal - though of course if the original cause of the bus timeout still applies, the whole process will be repeated.

Note that the CLK INT is an output from the PIT programmable timer chip (see pg. 5-69...70). This output, as well as defining the bus timeout period, can be used by software for its own timing purposes. It should be remembered therefore, when programming it, not to set it to a shorter period than 50 microseconds since the hardware bus timeout mechanism will not work correctly with periods much less than this.

5.3.4.1 I/O AND MEMORY ADDRESSING

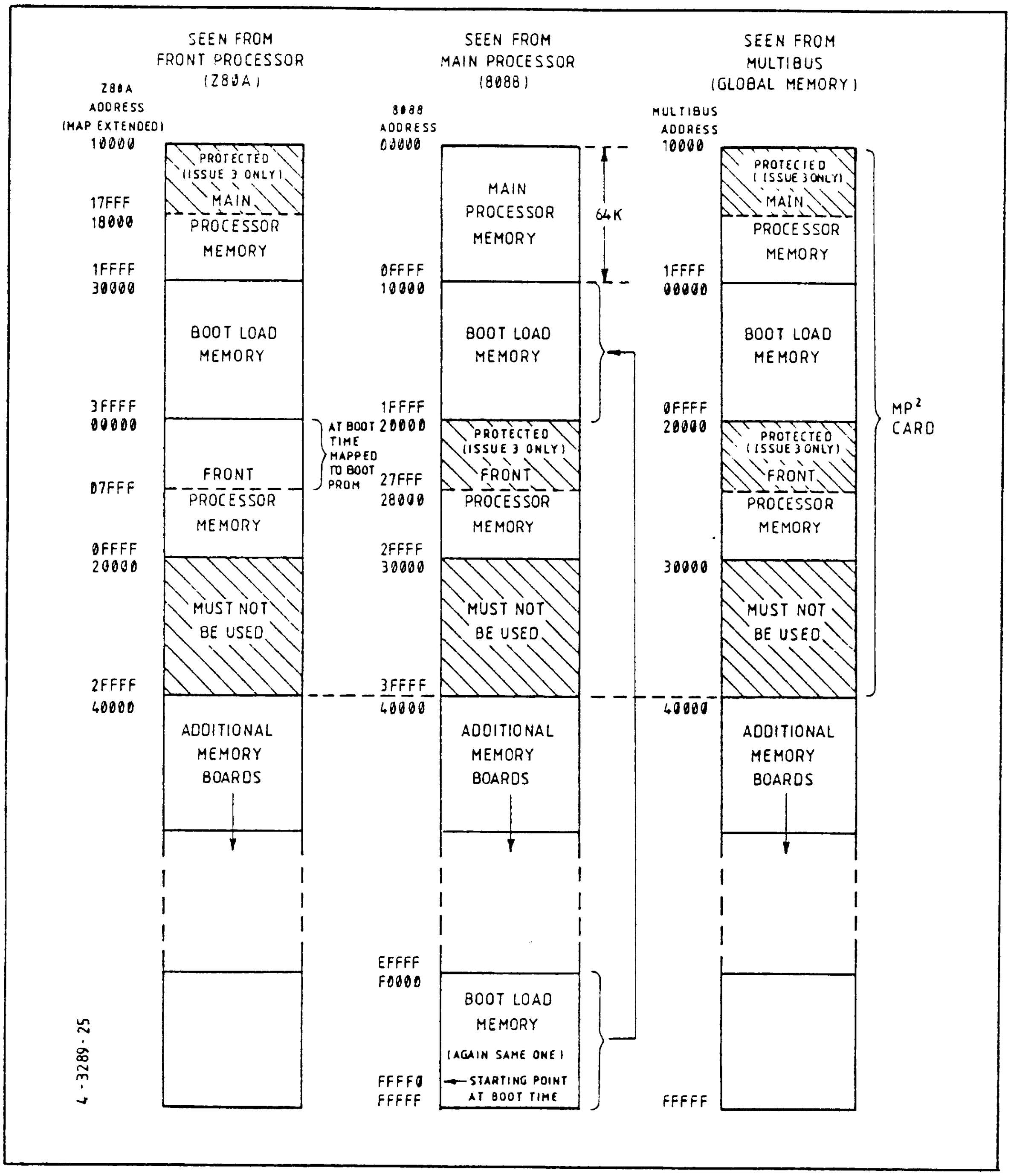
This section describes the local and global address map for each of the processors of the MP^2 and addresses as seen from the Multibus. Section 5.3.4.2 is a description of the Map function performed by the Front Processor.

The Boot Load PROM contains Boot Load programs and built in test programs for both processors and is normally only accessable after a reset or when a "Clear Boot Mode" output from the PIO is activated. Note that I/O addresses and memory addresses, and the use thereof, are totally independent.

The following diagram summarises how different memory areas are addressed from the two processors and from the Multibus.

Note that the three columns in the diagram all refer to the same physical memory locations. In other words, any 3 local-relative addresses read across the diagram horizontally refer to the same physical byte of memory.

The one exception is the last page for the main processor which is not a Multibus address as might be expected; it refers back to the MP^2 boot PROM.



CR8 MEMORY MAP

MP ^{2} BOARD BASE ADDRESS

In the CR8 Memory Map above, the MP^2 board has a base address of 00000 Hex on the Multibus (the Boot Load Memory in fact). By use of the jumpers SR18 and SR19 it is possible to change this base address in steps of 40000 Hex (4 pages) to any of four positions within the total 1 megabyte addressing range (16 pages).

The exact details are given below:

	SR19	SR18	MP^2 BASE ADDRESS
No.0	A	A	00000
No.1	A	B	40000
No.2	B	A	80000
No.3	B	B	C0000

Thus, for example, if the No.2 settings of the jumpers had been selected, then the number 80000 Hex would have to be added to every MP^2 address in the right hand "Multibus" column of the Memory Map diagram, in order for it to be correct. It should be stressed that these jumpers only affect the addresses **within** the MP^2 as seen from **outside** the MP^2.

Note that these jumper settings also affect the I/O address space (see pg. 5-107).

I/O Address	Item	From Front Processor	From Main Processor	From Multi-Bus *
00-07	CTC	V	X	X
08-OF	PIO	V	V	V
10-1F	SIO	V	V	V
20-2F	Map Register 3 (4 bits) Floppy Drive Select (4 bits)	V	V	V
30-3F	DMA	V	V	V
40-4F	Floppy Disk Ctrl	V	V	V
50-57	Multimodule Interface #1, Chip Select 0	V	V	V
58-5F	Multimodule Interface #1, Chip Select 1	V	V	V
60-67	#2, 2	V	V	V
68-6F	#2, 3	V	V	V
70-77	#3, 4	V	V	V
78-7F	#3, 5	V	V	V
80-8F	See here $\longrightarrow$	I/O on Multibus	Interrupt to Front Pr.	X
90-CF	I/O on Multibus	V	V	X
DO-DF	8353 PIT	V	V	V
EO-FF	I/O on Multibus	V	V	X
0100-03FF	See here $\longrightarrow$	M.S byte of address is ignored	X	X
0400-3FFF	See here $\longrightarrow$		X	Other devices on multibus - not MP^2
4000-FFFF	I/O on Multibus		V	

Key:

V Addressable
X THIS ADDRESS RANGE MUST NOT BE USED
* Note that the information given in the Multibus column applies when the jumpers SR19 and SR18 are set at A and A. If they are not, please refer to the Table overleaf/below.

CR8 I/O MAP

SR19	SR18	MP^2 BASE ADDRESS (I/O)
A	A	0000 Hex
A	B	4000 Hex
B	A	5000 Hex
B	B	6000 Hex

Note that these jumpers also affect memory addressing offsets (see pg. 5-105). Also, note that the MP^2 always occupies an I/O addressing range of 400 Hex.

ADDRESSING THE BOOT PROM

The 8088 CPU starts after power up or reset at address FFFFO. If only a 4K Prom (2732A) is used, address FFFFOH from the 8088 will be address FFOH in the PROM. If it is an 8K PROM (2764A), the address will be 1FFOH, and if it is a 64K PROM board, the address will be FFFO.

Note that the Z80, due to its small inherent addressing range, operates in two modes - boot mode where it is addesses only the boot PROM, and normal mode where the Map registers are enabled and it can address up to a megabyte of RAM (minus the boot PROM and one other page which is inaccessible).

5.3.4.2 EXTENDED ADDRESSING FOR THE Z80A

To increase the overall flexibility of the CR8, the addressing range of the Z80A has been increased from the normal 64K bytes up to a megabyte. Thus it matches that of the 8088 processor. This is implemented by means of 3 four-bit Map registers and Bit 3 in Port C of the PIO. The following table shows which of the 3 registers is used in different situations. Bit 3, Port C is known as "the inverter bit" and is used for all map registers and both addressing schemes (see pg 5-71).

Unfortunately there is a further complication - namely that two different schemes for memory mapping of the Z80A may be used. This is necessary for software compatibility with MP/M (scheme A) and pre-existing CR software (scheme B). The scheme to be used is selected by means of jumper R51 on the MP^2. Set to A for scheme A or B for scheme B.

Controlling Device	Type of Data Transfer	Memory Map Register 1	2	3
Front Processor	Between CPU and memory external to the front processor			X
Front or Main Processor	Memory to memory DMA transfer	Data Read	Data Write	
Front or Main Processor	DMA transfers between Z80A's SIO and memory (either direction)	X		
Front or Main Processor	DMA Transfers Between Floppy Disk Controller and Memory (either direction)		X	

ASSIGNMENT OF THE MAPPING REGISTERS

Ref. to text	Comments	INPUTS TO MAPPING PROCESS								OUTPUT ADDRESS THAT IS REFERENCED BY THE Z80A						
		Map Register (4 bits)				Inverter Bit	Z80A Address (16 bits)									
							Bit 15	Bit 14	Bits 13 to 0	19	18	17	16	15	14	13 to 0
1.	2 m.s. bits of Z80A address are both "1" : processor accesses its own memory (mapping not in operation).	p	q	r	s	t	1	1	a to n	0	0	0	0	1	1	a to n
2a.	Bits 15 and 14 are not both "1", so mapping operates. Inverter bit is "1", ans so no inversion of bit 15.	p	q	r	s	1	1	0	a to n	p	q	r	s	1	0	a to n
2a.	Again, bits 14 and 15 are not both "1", therefore mapping occurs.	p	q	r	s	1	0	1	a to n	p	q	r	s	0	1	a to n
2a.	Similar to example above	p	q	r	s	1	0	0	a to n	p	q	r	s	0	0	a to n
2b.	Bits 14 and 15 not both "1", therefore mapping. Inverter bit "0", therefore bit 15 inverted.	p	q	r	s	0	1	0	a to n	p	q	r	s	1	0	a to n
2b.	Similar to example above	p	q	r	s	0	0	1	a to n	p	q	r	s	1	1	a to n
2b.	Similar to examples above	p	q	r	s	0	0	0	a to n	p	q	r	s	1	0	a to n

SCHEME "A" MEMORY MAPPING

In the diagram on the previous page (scheme 'A'), the letters a to n and p to t represent particular bits which can of course be set to "0" or "1". To summarize what happens:

1. Logic checks bits 14 and 15 of the address and if they are both "1", then nothing further is done; the 16-bit address is used exactly as it is.

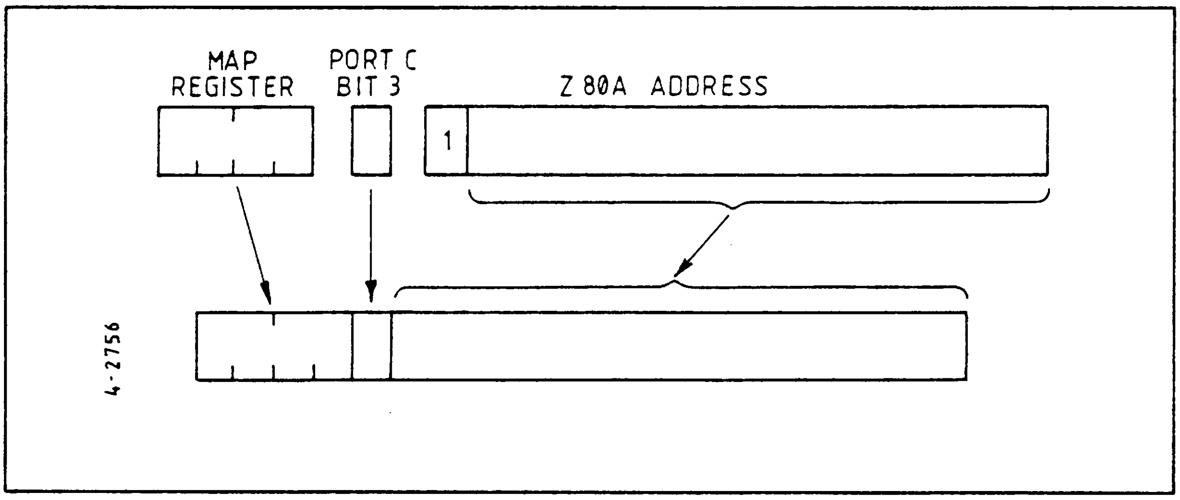
2a. The logic looks at the inverter bit (assuming it did not stop after stage 1), and if it is "1", the final address is the concatenation of the map register with the 16-bit address.

2b. If the inverter bit is "0", the final address is as in stage 2a except that bit 15 is inverted ("0" would become "1", "1" would become "0").

Scheme 'B'

This scheme is much simpler. It works as follows.

1. The logic tests bit 15. If it is "0", then no mapping takes place; the address is a local address (16 bits), the m.s. being "0".
2. If bit 15 was "1", then the address is a concatenation of the map register, the inverter bit itself, and bits 14 to 0 of the Z80A address. This is illustrated below.



SCHEME 'B' EXTENDED ADDRESSING

Remember of course, that this only happens when bit 15 is "1".

5.3.5 BATTERY BACK-UP

To protect against data loss during power failures or simple power-off, a battery back-up option has been implemented.

The circuits which may be supplied from the Multibus connection called "5V Battery" are the dynamic RAM memories of the Main Processor and the Front processor.

When the main power supply (5V) level falls below 4.75 volts, the RAM chips and the refresh and control circuits must be supplied from this optional connection, and external access to the memories while in this state is not possible.

There are two 5V power rails on the MP^2 card - VCC and VCC2. VCC2 powers the RAMs and VCC powers everything else on the MP^2 card. The floppy disk controller also requires +12V and the V24 interface also requires both +12V and -12V. Normally VCC and VCC2 are shorted together on the Multibus backplane. This connection must be broken if a separate battery backup is to be provided for VCC2 only.

5.3.6 POWER SPECIFICATIONS

Power must be supplied at the following three voltages:

VCC	5V	approx. 5.0A $\pm$ 0.5A	
VCC2	5V	approx. 0.5A	(Active RAM operation)
		approx. 0.35A	(Memory retention during power failure - no memory read/write).
+12V		approx. 0.1A	
-12V		approx. 0.1A	

5.3.7 JUMPER AND SWITCH SETTINGS - MP^2 CARD

JUMPER NO.	MOUNTED	A	B	JUMPER NO.	MOUNTED	A	B
1	X			26			
2	mounted in 'H'			27			
3	X			28			
4	X			29			
5		X		30			X
6				31			X
7			X	32		X	
8			X	33			X
9				34			
10				35*	X		
11	X			36			X
12	X			37			
13	X			38			
14*	X			39			
15		X		40			
16				41		X	
17				42		X	
18		X		43			
19		X		44		X	
20				45			X
21				46		X	
22			X	47			
23				48	X		
24			X	49		X	
25							

*) Two outermost pins connected

MP^2 BOARD (ISSUE 3) JUMPERS
WHEN INSTALLED IN THE CR8 CABINET MODEL OR THE 8-SLOT WORKSTATION

JUMPER NO.	MOUNTED	A	B	JUMPER NO.	MOUNTED	A	B
1	X			26			
2	mounted in 'H'			27			
3	X			28			
4				29			
5		X		30			X
6	X			31			X
7		X		32		X	
8		X		33			X
9		X		34			
10	X			35*	X		
11	X			36			X
12	X			37			
13	X			38			
14*	X			39			
15		X		40			
16				41		X	
17				42		X	
18		X		43			
19		X		44		X	
20	X			45			X
21	X			46		X	
22		X		47			X
23		X		48	X		
24		X		49		X	
25							

*) Two outermost pins connected

MP^2 (ISSUE 3) JUMPERS - WHEN INSTALLED IN THE CR8 6-SLOT INTEGRATED WORKSTATION MODEL

EXPLANATIONS OF JUMPERS & SWITCH SETTINGS
JUMPERS (SR1 TO SR51)

1. Mounted: 10 MHz oscillator's output is connected to the rest of the MP^2.
2. A: MP^2 uses Multibus interrupt line no. 6. B: Multibus int. 7. C: 4 D: 5 E: 2 F: 3 G: 0 H: 1.
3. Mounted: Connect 10 MHz from MP^2 oscillator to CCLK/ on Multibus.
4. Mounted: BPR0/ from MP^2 connected to Multibus. Serial priority. Not mounted: parallel priority.
5. A: 10 MHz clock on MP^2 is the source of Multibus BCLK/ signal. B: MP^2 receives 10 MHz from Multibus BCLK/.
6. Mounted: 8203A RAM controller in use. Not mounted: 8202A RAM controller in use.
7. A: 8203A RAM controller in use with 64K RAM. B: 8203A with 16K; or 8202A controller in use.

8: Set this as for jumper SR7.

9: A: 8203A RAM controller in use with 64K RAM. B: 8203A with 16K. Not mounted: 8202A in use.

10: Mounted: 8203A RAM controller in use with 64K RAM. Not mounted: any other memory size/controller-type combination.

11: If neither A nor B is mounted on SR27, or no Multimodule in connector J1, then mount this.

12: If neither A nor B is mounted on SR26, or no Multimodule in connector J2, then mount this.

13: If neither A nor B is mounted on SR25, or no Multimodule in connector J3 then mount this.

14: Issue 3 (and earlier) of MP^2 only: There is no SR14 on Issue 4 but see SR50. A: the main processor interrupts the front processor. B: The main processor interrupts the Multibus as per jumper SR2. Wirewrap outermost pins together: Multibus (as per SR2) interrupts front processor.

15: A: 2732A boot PROM (5 Volt supply). B: Address line 13 is connected.

16: A: 2764A boot PROM. "1" to pin 27. B: Address line 14 is connected.

17: A: 2764A boot PROM (5 volt supply). B: Address line 15 is connected.

18: MP^2 address on Multibus: see pgs. 5-104..105.

19: MP^2 address on Multibus: see pgs. 5-104..105.

20: Mounted: 8203A RAM controller in use. Unmounted: 8202A RAM controller in use.

21: Set this as for jumper SR20.

22: A: 8203A RAM controller in use with 64K RAM. B: 8202A RAM controller in use; or 8203A controller with 16K RAM.

23: A: 8203A RAM controller in use with 64K of RAM. B: 8203A Controller with 16K RAM; Not mounted: 8202A controller.

24: Set this as for jumper SR22.

25: A: Select interrupt 1 from Multimodule connector J3. B: Select int. 2 from J3.

26: A: Select interrupt 1 from Multimodule connector J2. B: Select int. 2 from J2.

27: A: Select interrupt 1 from Multimodule connector J1. B: Select int. 2 from J1.

28: Mounted: NMI from pin 19 on P2 connector is used as power fail interrupt.

29: Mounted: Parity error in the dynamic memory of main processor will cause an NMI.

30: A: 8088 CPU is not mounted but its memory is and access is required to that memory. B: 8088 is mounted.

31: A: line C115 (input) is used as receiver clock on channel B (V24 interface). B: CLK2 from PIT on MP^2 is used as receiver clock on channel B (V24).

32: A: CLK1 from PIT is used as transmitter clock on channel A (V24 interface) B: Line C114 (input) is used as transmitter clock on channel A (V24).

33: A: line C115 (input) is receiver clock on channel A.
B: CLK1 from PIT is receiver clock on ch. A.

34: Mounted: CLK1 can be used as output on line C114.

35: MP^2 issue 3. A: 1797 floppy controller. Outer 2 pins wirewrapped together: 1793 Floppy controller
MP^2 issue 4. A: not used. B: 1793 floppy controller in use.

36: A: DR2/ signal is used as drive select. B: DR2/ used for side select.

37: Mounted: Parity error in dynamic memory of front processor will cause an NMI.

38: Mounted: Watchdog logic connected to NMI line.

39: Mounted: Bus timeout will cause an NMI.

40: Mounted: No Z80A but access is required to front processor memory or I/O.

41: A: Double density. B: Single density floppies.

42: Mounted: 5 1/4" floppy. Not mounted: 8" floppy (INDEX/ is on different pins on the 2 types).

43: Mounted: use DR3/ drive select signal on 5 1/4" floppy.

44: A: Set this if SR43 is mounted. B: Ready signal from floppy is connected to controller (8").

45: A: 4 MHz 5 1/4" single density. B: 8 MHz 5 1/4" double or 8" single. C: 16 MHz 8" double density.

46: A: early/late write on floppy disk. B: Normal write.

47: A: 8 MHz B: 4 MHz 5 1/4" doubled sided.

48: Mounted: 16 MHz clock connected.

49: A: 1 MHz 5 1/4 floppy disk interface. B: 2 MHz 8" floppy.

50: Issue 4 of MP^2 only. See also SR14. A: Multibus interrupts front processor on the line as defined by jumper SR2. B: Main processor interrupts front processor. C: Main processor interrupts Multibus (on line specified by SR2).

51: Issue 4 of MP^2 only. A: Use scheme "A" (old) Z80A address mapping. B: Use scheme "B" (MP/M compatible). Z80A mapping. Further details on pgs. 5-107 to 5-110.

SWITCHES

There is just one set of four DIP switches whose functions are as follows:

1. Closed: Boot an operating system including a Winchester driver from the Winchester.
   Open: Boot a floppy-only system from the floppy drive. Normally this is simply set depending on the presence or absence of a Winchester drive; however a Winchester/floppy system can be booted from the floppy (Switch 1 OPEN) if for example the Winchester disk is unformatted. However, the Winchester drive would not be recognized by the operating system in such a case.
2. Closed: One or more SCI boards on the Multibus
   Open: No SCI boards present
3. See table below
4. See table below.

Switch 3	Switch 4	Effect
Open	Open	Operating system expects no extra RAM on the Multibus RAM.
Open	Closed	128K bytes extra installed on Multibus.
Closed	Open	256K extra
Closed	Closed	512K extra.

5.3.8 DIFFERENCES BETWEEN MP^2 ISSUE 3 & 5

The issue number is printed in the top right hand corner of the board - see illustration on pg. 5-76. The issue number is the last number that appears there. The illustration is actually of an issue 4 board, which was never released but looks exactly like an issue 5 board.

Issue 3	Issue 5
Scheme "A" only in effect when using address mapping registers with Z80A. See pgs. 5-109...110.	Scheme "A" or "B" can be used-selectable by jumper 51.
Half of Z80A's memory and half of the 8088's memory (see pg.5-104) is protected so that it can only be accessed by the processor it belongs to.	No protection of either processor's memory (even when switched to scheme "A" mapping)
Jumpers SR14, SR35, SR50, SR51 have differences - see pages 5-115 to 5-117.	

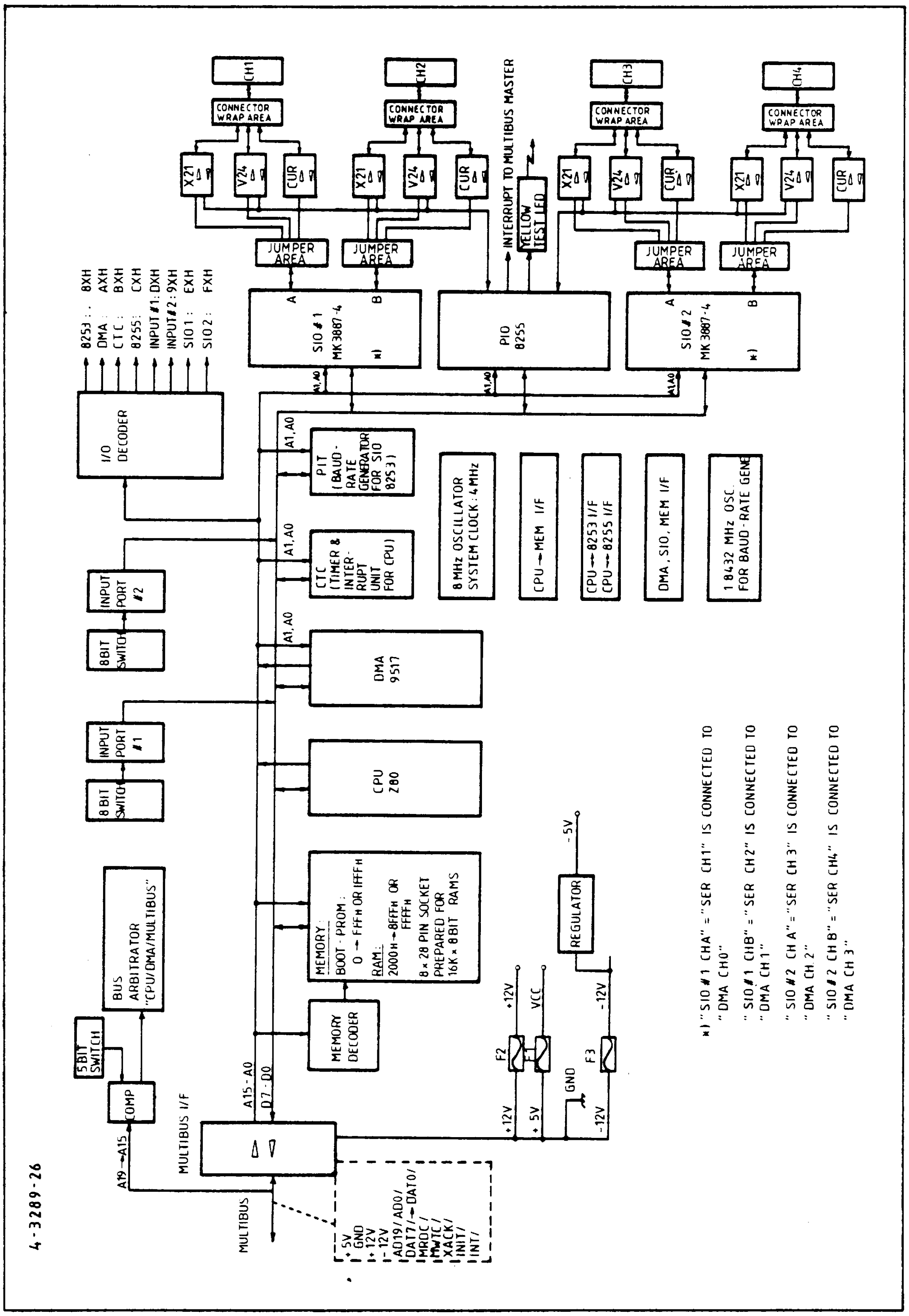
It is possible to convert an Issue 3 board to the functions of an issue 5 board (in particular to use it with MP/M). This involves changing a PROM. Please consult CR for details.

5.4 THE SCI BOARD

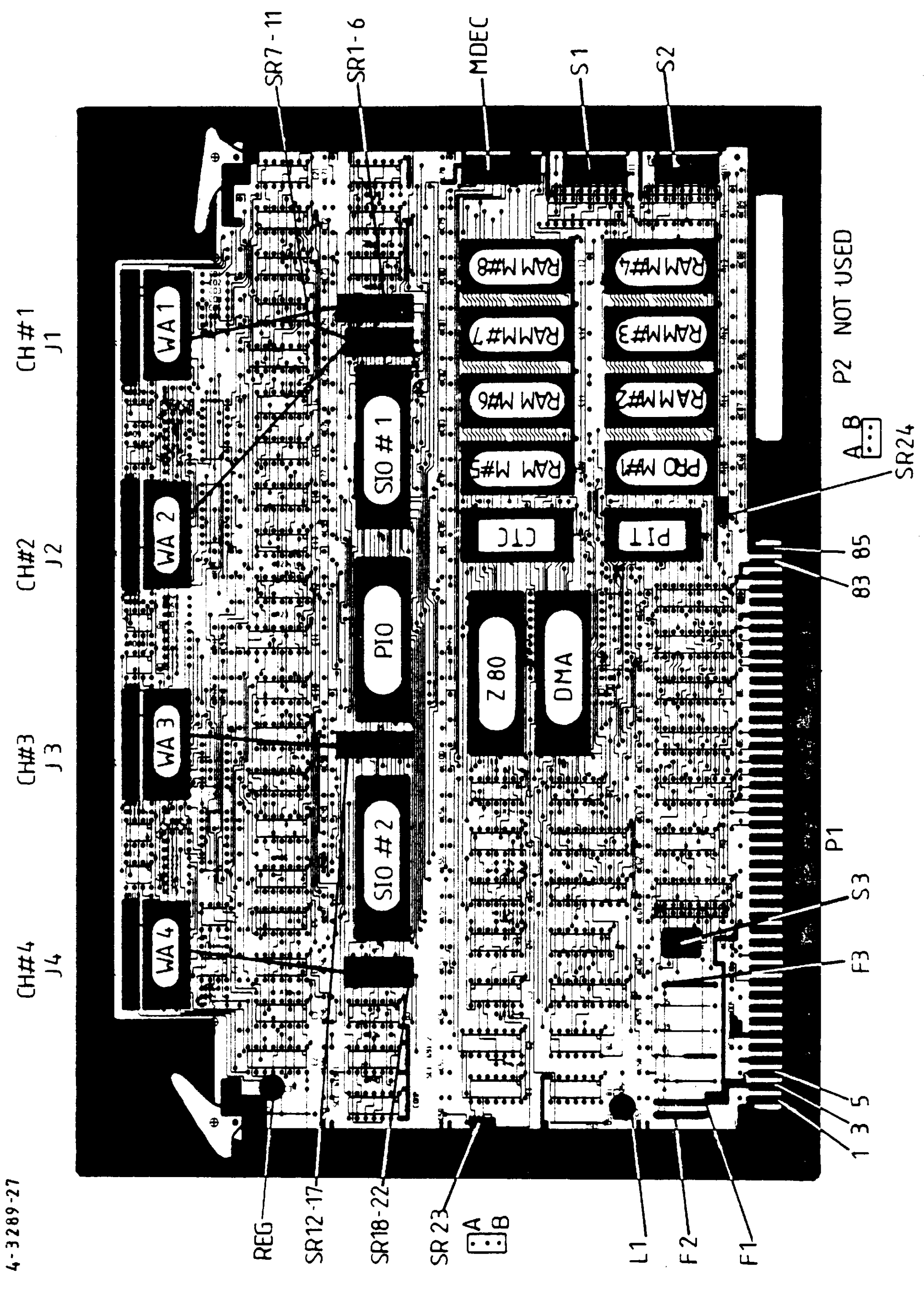
5.4.1 INTRODUCTION

The Serial Communication Interface (SCI) board provides an extra four serial channels in the CR8, for the connection of terminals, printers, modems, etc. Each channel is independently strappable to X21/DTE, V24 (DTE or DCE) or 20mA current loop.

The board itself is a Z80A-based microcomputer system running a subset of the host computer's operating system, downloaded during system boot. It has 4K or 8K PROM address space and 28K (optionally 56K) bytes of RAM address space. The Multibus master can address all of the on-board RAM. The SCI can optionally protect specified areas of memory from access from the Multibus master. Communications software and programmable link characteristics are downloaded during system boot. The SCI will typically handle the lower levels of the protocol, while the connected MP^2 will handle the higher levels of protocol routines. The SCI is designed to execute existing programs from similar modules in the CR80 computer. This means that X25, ICL CO1/2/3 and IBM 2780 interfaces are already developed, tested and ready for implementation in the SCI.


4-3289-26
CH1
CH2
CH3
CH4
CONNECTOR WRAP AREA
X21
V24
CUR
JUMPER AREA
INTERRUPT TO MULTIBUS MASTER
YELLOW TEST LED
SIO #1 MK 3887-4
PIO 8255
SIO #2 MK 3887-4
A
B
*)
A1,A0
8253 : 8XH
DMA : AXH
CTC : BXH
8255 : CXH
INPUT #1 : DXH
INPUT #2 : 9XH
SIO 1 : EXH
SIO 2 : FXH
I/O DECODER
PIT (BAUD-RATE GENERATOR FOR SIO 8253)
CTC (TIMER & INTERRUPT UNIT FOR CPU)
8 MHz OSCILLATOR SYSTEM CLOCK: 4 MHz
CPU→MEM I/F
CPU→8253 I/F
CPU→8255 I/F
DMA, SIO, MEM I/F
1.8432 MHz OSC. FOR BAUD-RATE GENE
INPUT PORT #2
8 BIT SWITCH
DMA 9517
INPUT PORT #1
8 BIT SWITCH
CPU Z80
MEMORY: BOOT-PROM: 0 → FFFH OR 1FFFH
RAM: 2000H → 8FFFH OR FFFFH
8 × 28 PIN SOCKET PREPARED FOR 16K × 8 BIT RAMS
MEMORY DECODER
BUS ARBITRATOR "CPU/DMA/MULTIBUS"
5 BIT SWITCH
COMP
A19 → A15
A15 - A0
D7 - D0
MULTIBUS I/F
MULTIBUS
+5V
GND
+12V
-12V
AD19/AD0/
DAT7/→DAT0/
MRDC/
MWTC/
XACK/
INIT/
INT/
F2
F3
+12V
VCC
+5V
GND
-12V
REGULATOR
-5V
*) "SIO #1 CHA" = "SER CH1" IS CONNECTED TO "DMA CH0"
" SIO #1 CHB" = "SER CH2" IS CONNECTED TO "DMA CH 1"
" SIO #2 CH A" = "SER CH 3" IS CONNECTED TO "DMA CH 2"
" SIO #2 CH B" = "SER CH4" IS CONNECTED TO "DMA CH 3"

SCI BLOCK DIAGRAM



SCI BOARD LAYOUT

Key to illustration above:

J1, WA1, SR1-6	On board connector and wrap/jumper area for serial channel #1
J2, WA2, SR7-11	On board connector and wrap/jumper area for serial channel #2
J3, WA3, SR12-17	On board connector and wrap/jumper area for serial channel #3
J4, WA4, SR18-22	On board connector and wrap/jumper area for serial channel #4
SR23	Selecting clock for PIT
SR24	Selecting memory type
LI	Yellow light emitting diode connected to PIO
F1	Fuse 5A for 5V
F2	Fuse 0.5A for +12V
F3	Fuse 0.5A for -12V
P1	43 x 2-pin Multibus board connector
P2	Not used
S1	8 bit DIL switch ... Input port #1
S2	8 bit DIL switch ... Input port #2
PROM M#1 to RAM M#8	PROM/RAM memory area
MDEC	Memory Decoder
Z80	Z80A Central Processing Unit ... MK 3880/Mostek
SIO#1, SIO#2	Serial input output controller..MK 3884/Mostek
PIO	Programmable Parallel input output port .. 8255/Intel
DMA	Direct memory access controller ..AM9517A/AMD
CTC	Counter/timer circuit..MK3882/Mostek
PIT	Programmable Interval Timer.. 8253/Intel
REG	-5V regulator ... from -12V
S3	5 bit DIL switch .. selecting Multibus address space for SCI.

5.4.2 FUNCTIONAL DESCRIPTION

This description refers to the block diagram on page 5-120.

The SCI acts as an intelligent slave on the Multibus, that is, all data transfers are initiated by the master device (generally the MP^2 card). However, the SCI can attract the attention of the Multibus master via the Multibus interrupt line.

Seen from the Multibus master, the SCI is just a part of the Multibus memory. This means that only 5 protocol signals from the Multibus are meaningful to the SCI. These signals are: INIT, MRDC, MWTC, INT1 and XACK (all active low). A short description of these signals are given below:

INIT is activated by the master to ensure a well-defined state of all MULTIBUS slaves, i.e. power-on.

MRDC/MWTC are activated by the master when the address lines are stable during read (MRDC) or write (MWTC).

XACK is the slave acknowledgement of a master command. XACK indicates that data has been placed on or accepted from the data lines.

INT1 is a request for master attention. The slave activates this line under software control.

For further details, refer to Multibus specifications.

The SCI is equipped with 28K bytes of static RAM. The area is under software control, divided into a common buffer area and an SCI program area. This program is downloaded from the MP^2 via the Multibus. Once loaded, the SCI can optionally protect the program area via the parallel port (PIO). The MP^2 access takes place as follows:

The MP^2 sets the address lines and appropriate control signal (MRDC or MWTC). The SCI compares the 5 MSBs of the address to a 5-bit switch. (Thus several SCIs may to be connected to the Multibus on different addresses). If a match is detected, the **Comparator** signals the **Bus Arbitrator** circuit. This circuit then requests the Z80 CPU to release the SCI internal bus. The Z80 CPU finishes its current operation and then acknowledges the **Bus Arbitrator** to indicate that the bus is available. The Bus Arbitrator enables the address drivers. The Multibus master (normally the MP^2) is also allowed to enable either the data drivers or the data receivers according to the read/write control signals. The address lines are decoded by a small control circuit to determine if access to a protected area is being attempted. If it is, the MP^2 board will time out. If not, access is allowed and XACK is activated when the transaction is completed.

If the DMA or one of the two serial I/O controllers (SIO1, SIO2) wants to interrupt the CPU, this is done in a daisy-chain mode (see pg. 5-126). When the Z80 acknowledges the interrupt, the interrupting device must put its interrupt vector out on the datalines so that the Z80 can find the program to execute as a result of the interrupt.

The DMA device does not have this capability of displaying an interrupt vector as described above, so the CTC is used. The CTC is a counter/timer circuit but in this application is used as an intermediary between the DMA & the CPU. The DMA circuit interrupts the CTC, and the CTC then forwards the interrupt to the CPU. When the CPU requests the vector of the interrupting device, the CTC outputs a prestored vector, telling the CPU that it was the DMA device that interrupted.

Note that the DMA controller can only move data between the four channels and memory; it cannot perform direct memory access from memory to memory.

The clock signal for the SIOs is generated by the **Baud Rate Generator** circuit. This programmable device has only 3 timers; thus channels 1 and 2 have their own clock frequencies but channels 3 and 4 must share a common frequency. Input to the Baudrate generator is strap-selectable to either 2MHz or 1,8432 MHz. Two 8-bit switches are implemented as read-only ports. The interpretation of the switches is under full software control.

To support X21, V24 or current loop the parallel port **(PIO)** is implemented. The PIO has 3 eight-bit ports. During initialization each port can be pre-programmed to act as either input or output. Two of the ports are used for additional control signals to support the 4 serial channels, while the third port has 3 active output lines. One controls the yellow LED, one will activate the Multibus interrupt line and the last will activate the protect mechanism of the Z80 program area in memory. The use of these 3 lines is under full software control.

Some of the more important SCI components are considered in more detail below.

5.4.2.1 Counter/timer circuit

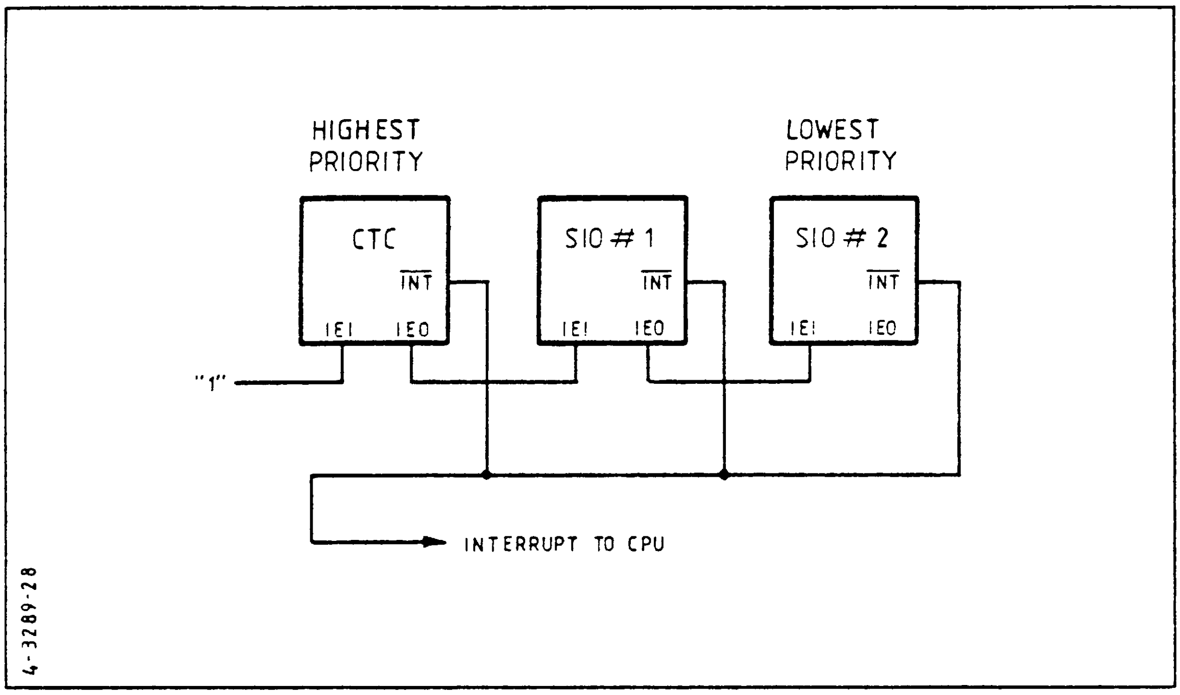
The Z80-counter/timer circuit (CTC-MK3883) is a programmable component with four independent channels that provide counting and timing functions for the CPU. All modes of operation are software programmable by I/O operations. CTC channel #0 is connected to the DMA's "end of process line" (EOP/ - active low) which gives the opportunity for making an interrupt to the CPU when a DMA job has finished. All other CTC channels are reserved as timers. See also "Daisy chain interrupt priorities" (pg 5-126), "I/O Address MAP" (pg. 5-131) and "MOSTEK 1979 -Microcomputer components data book".

5.4.2.2 Serial Input/Output Controllers

The serial input output controllers (Z80-SIO-MK3887) are two I/O programmable, dual channel devices which provide formatting of data for serial communication in many synchronous and asynchronous modes. The SIO's can operate in 3 different ways - either polled, interrupt controlled or controlled by DMA. For DMA control, the SIO "WAIT/READY" line is connected to the DMA.

5.4.2.3 "Daisy Chain" interrupt priorities

Interrupts from the CTC and the two SIO's are organized in a daisy chain as shown below.



DAISY CHAIN INTERRUPT STRUCTURE ON SCI

Thus devices of lower priority have to "obtain permission" from all devices of higher priority before they can interrupt the CPU.

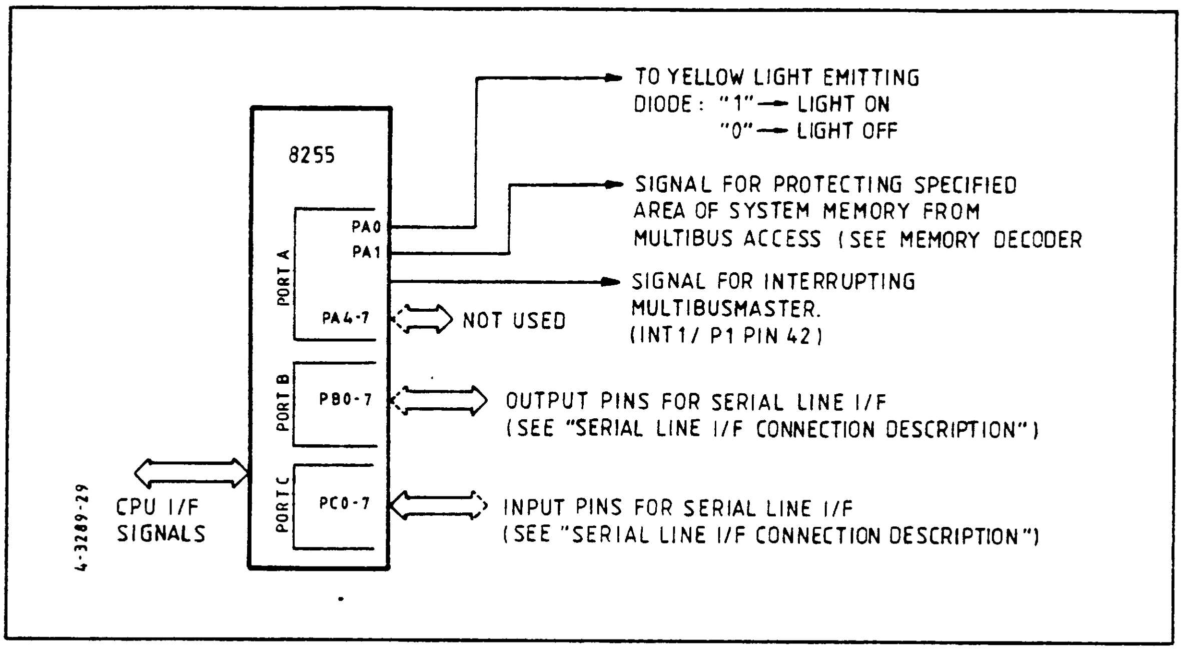
5.4.2.4 Programmable Interval Timer

The PIT serves as the baud rate generator for SIO1 and SIO2. The INTEL 8253 is a programmable counter/timer organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable by I/O operations.

The clock frequency for the baud rate generator is selected by jumper SR23. Alternatively, external clock pulses may be used from the X21 or V24 interfaces. The choice is made by means of jumpers (nos. 5,6,10,11,16,17,21,22). Full details may be gleaned from the circuit diagrams on pgs. 5-140 to 5-143.

5.4.2.5 Programmable Parallel I/O Port

This INTEL-8255A PIO chip is a general purpose programmable I/O device with 24 I/O pins. On the SCI-board the pins is assigned as shown below.


8255
PORT A
PA0
PA1
PA4-7
NOT USED
PORT B
PB0-7
PORT C
PC0-7
CPU I/F SIGNALS
TO YELLOW LIGHT EMITTING DIODE: "1" → LIGHT ON "0" → LIGHT OFF
SIGNAL FOR PROTECTING SPECIFIED AREA OF SYSTEM MEMORY FROM MULTIBUS ACCESS (SEE MEMORY DECODER
SIGNAL FOR INTERRUPTING MULTIBUSMASTER. (INT1/ P1 PIN 42)
OUTPUT PINS FOR SERIAL LINE I/F (SEE "SERIAL LINE I/F CONNECTION DESCRIPTION")
INPUT PINS FOR SERIAL LINE I/F (SEE "SERIAL LINE I/F CONNECTION DESCRIPTION")
4-3289-29

PIN ASSIGNMENT OF PIO

For further information see "I/O ADDRESS MAP" (section 5.4.2.11), "INTEL COMPONENT DATA CATALOG 1982", and the Serial Line Interface Descriptions on pgs. 5-140..143. The yellow LED may be programmed to go on or off; it is normally switched on during the bootload procedure and off at other times.

5.4.2.6 Direct Memory Access Controller - DMA

The Direct Memory Access Controller (DMA 9517A) is a peripheral interface circuit for the CPU - designed to improve system performance by allowing external devices to directly transfer information to or from the system memory. On the SCI the external devices will be the four SIO-channels. Memory to memory transfer is not implemented on the SCI. The DMA contains four independent channels which in all modes of operations are software programmable by I/O operations.
Each DMA channel is hard-wired to a separate SIO channel by linking the SIO WAIT/READY line to the DMA REQUEST line. This provides for data-interchange

between RAM and SIO in half duplex mode. The WAIT/READY line in the SIO must be programmed for Ready Operation and will necessarily be active low which means that the DMA REQUEST line on the DMA must also be programmed as active low.

The DMA's END OF PROCESS line is connected to the CTC channel #0 which gives the opportunity for interrupting the CPU when the DMA job has finished.

DMA/CH # 0	is connected to	SIO # 1/CH # A
DMA/CH # 1	is connected to	SIO # 1/CH # B
DMA/CH # 2	is connected to	SIO # 2/CH # A
DMA/CH # 3	is connected to	SIO # 2/CH # B

DMA/SIO CONNECTION

For further information see "I/O Address Map" (section 5.4.2.11) and external reference: AMD -The Designers' Guide '80".

5.4.2.7 Booting the SCI

When RESET is set on the Multibus (by the MP^2), then the yellow test LED on the SCI will be lit. PROM-based bootload routines for the SCI will be entered and the SCI board will wait for the communications system and software-settable parameters to be downloaded into its RAM by the MP^2. The SCI will be told, by means of a handshake using 2 bytes in the SCI RAM as flags, when the bootload is complete. The SCI's Z80A will jump to the beginning of its code; the yellow LED will be switched off.

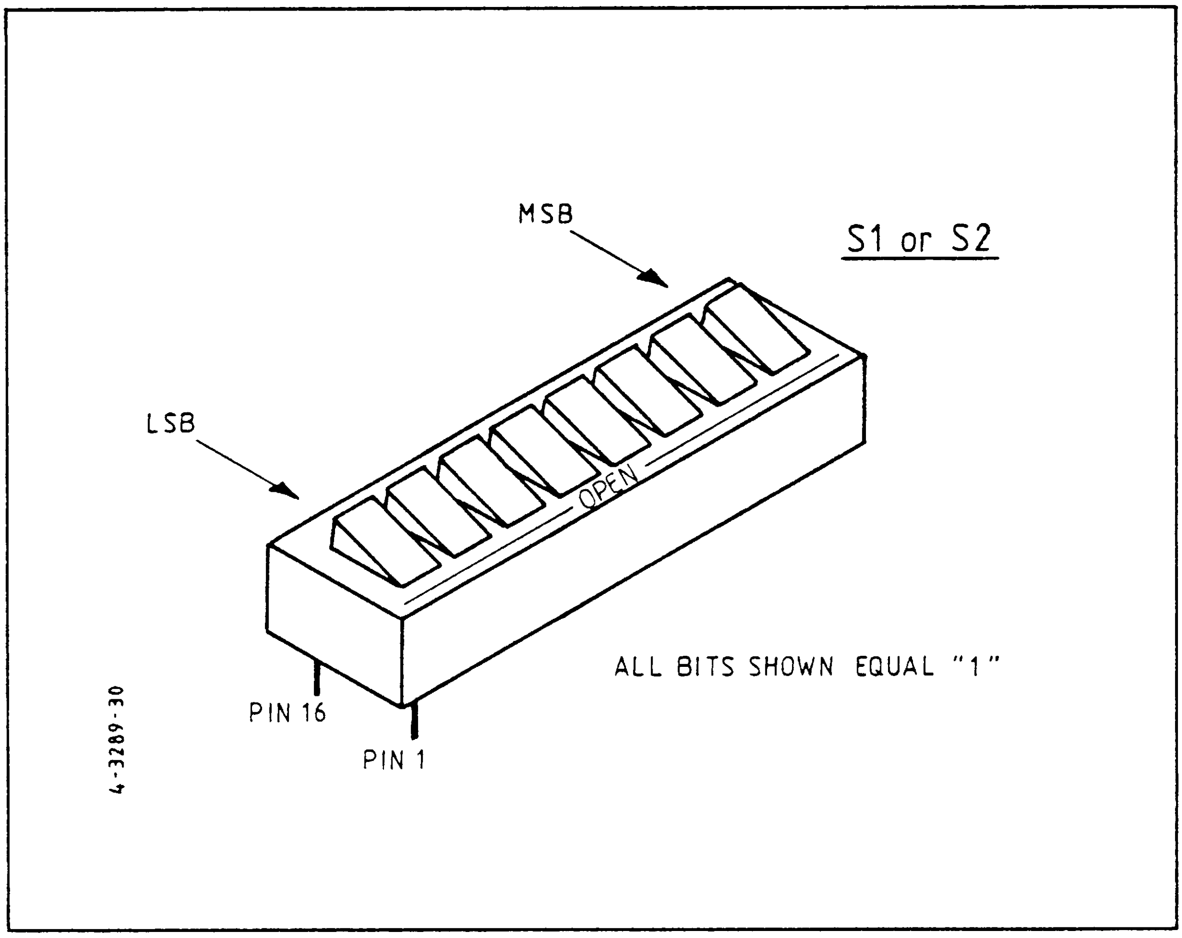
When bootloading is complete, the program memory area may be protected against corruption from other devices on the Multibus. This feature however requires the replacement of a small PROM on the SCI board (the firmware has to be changed).

5.4.2.8 Pin Assignments on Edge Connector

		(COMPONENT SIDE)			(CIRCUIT SIDE)	
	PIN	MNEMONIC	DESCRIPTION	PIN	MNEMONIC	DESCRIPTION
	1	GND	Signal GND	2	GND	Signal GND
	3	+5V	+ 5Vdc	4	+5V	+ 5Vdc
POWER	5	+5V	+ 5Vdc	6	+ 5V	+ 5Vdc
SUPPLIES	7	+12V	+12Vdc	8	+12V	+12Vdc
	9			10		
	11	GND	Signal GND	12	GND	Signal GND
	13			14	INIT/	Initialize
	15			16		
BUS	17			18		
CONTROLS'	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21			22		
	23	XACK/	XFER Acknowledge	24		
	25			26		
	27			28	ADR16/	
	29			30	ADR17/	Address
ADDRESS	31			32	ADR18/	Bus
	33			34	ADR19/	
	35			36		Parallel
	37			38		Interrupt
INTERRUPT	39			40		Request
	41			42	INT1/	
	43	ADR14/		44	ADR15/	
	45	ADR12/		46	ADR13/	
	47	ADR10/	Address	48	ADR11/	Address
ADDRESS	49	ADR8/	Bus	50	ADR9/	Bus
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADR0/		58	ADR1/	
	59			60		
	61			62		
	63		Data	64		Data
	65		Bus	66		Bus
DATA	67	DAT6/		68	DAT7/	
	69	DAT4/		70	DAT5/	
	71	DAT2/		72	DAT3/	
	73	DAT0/		74	DAT1/	
	75	GND	Signal GND	76	GND	Signal GND
	77			78		
POWER	79	-12V	-12Vdc	80	-12V	-12Vdc
SUPPLIES	81	+5V	+ 5Vdc	82	+5V	+ 5Vdc
	83	+5V	+ 5Vdc	84	+5V	+ 5Vdc
	85	GND	Signal GND	86	GND	Signal GND

PIN ASSIGNMENT OF BUS SIGNALS ON MULTIBUS BOARD CONNECTOR (P1)

5.4.2.9 On-board Switches S1 and S2

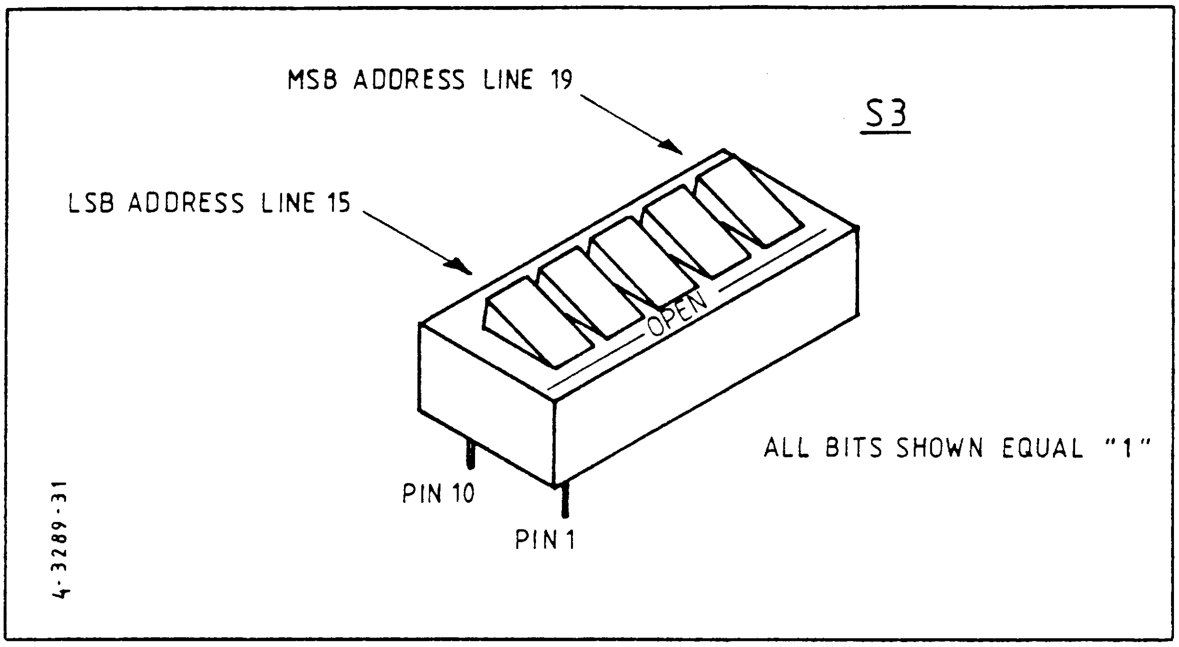


SWITCHES S1, S2

The positions of these switches may be read by software; they have no specified use but could be used for example to set up serial interface parameters such as baud rate, etc.

5.4.2.10 Switch S3 - Board Address

The five most significant bits of the MULTIBUS address line specify the SCI board address. The address is set up on the 5-bit switch S3.



SWITCH S3 - BOARD ADDRESS

5.4.2.11 I/O Address Map

Programmable Interval Timer - Pit - 8253:

80_H	Output:	load counter no. 0
	Input:	read counter no. 0
81_H	Output:	load counter no.1
	Input:	read counter no.1
82_H	Output:	load counter no.2
	Input:	read counter no.2
83_H	Output:	write mode word
	Input:	NOP

Counter Timer Circuit - CTC - MK3882:

$B0_H$	CTC CH # 0
$B1_H$	CTC CH # 1
$B2_H$	CTC CH # 2
$B3_H$	CTC CH # 3

Programmable Parallel Input Output Port - PIO - 8255:

CO_H	Output:	DATA BUS --> PORT A
	Input:	PORT A --> DATABUS
$C1_H$	Output:	DATA BUS --> PORT B
	Input:	PORT B --> DATABUS
$C2_H$	Output:	DATA BUS --> PORT C
	Input:	PORT C --> DATABUS
$C3_H$	Output:	DATABUS --> CONTROL
	Input:	ILLEGAL CONDITION

Input Port #1 - 8 Bit Switch - S1:

DO_H	Input Only - Switch READ

Input Port # 2 -8 Bit Switch - S2

90_H	Input only - Switch READ

Serial Input Output Controller SIO1 - MK3887:

EO_H	CH#A - Data
$E1_H$	CH#B - Data
$E2_H$	CH#A - Control
$E3_H$	CH#B - Control

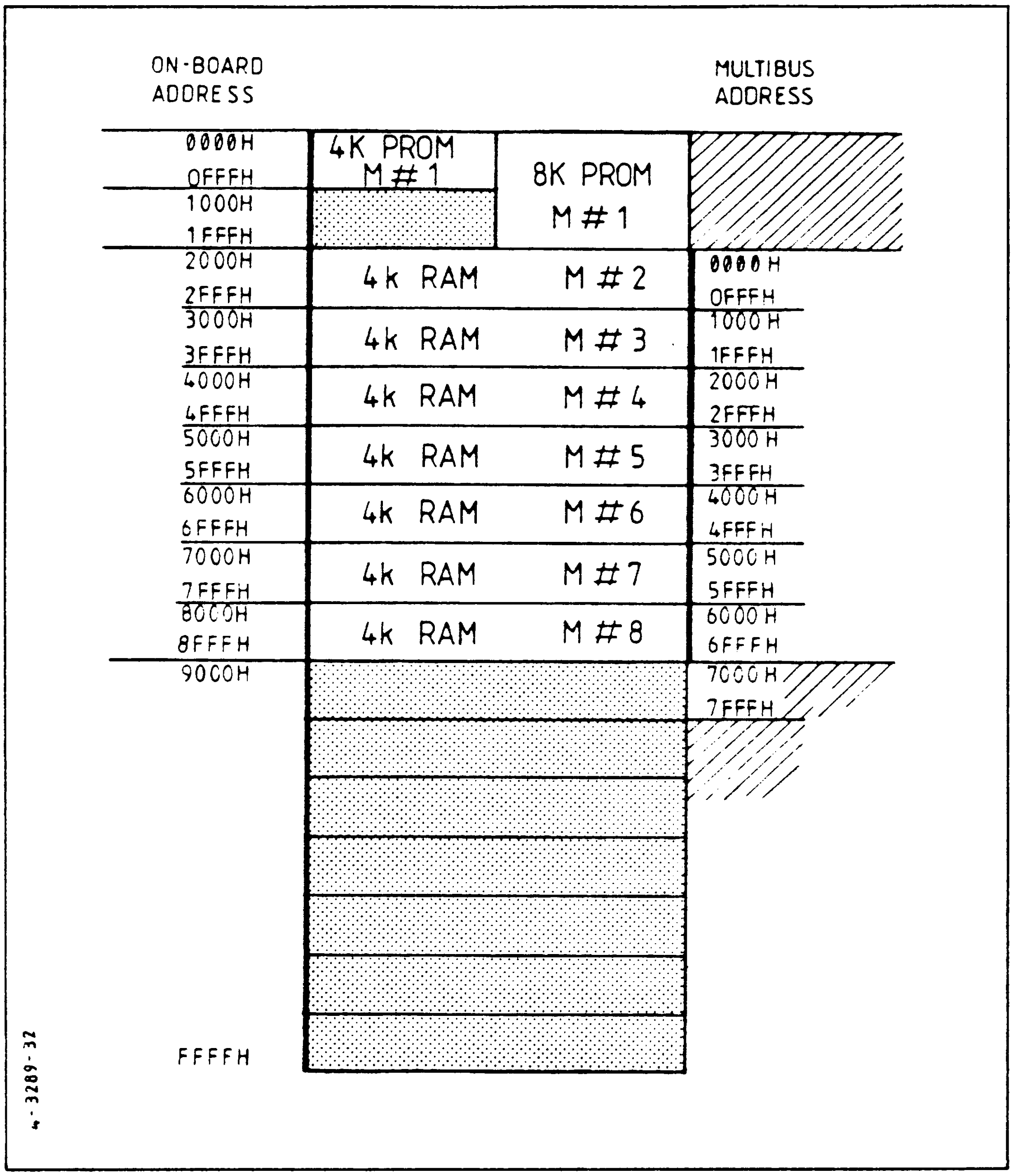
Serial Input Output Controller - SIO2 - MK3887:

FO_H	CH#A - Data
$F1_H$	CH#B - Data
$F2_H$	CH#A - Control
$F3_H$	CH#B - Control

Direct Memory Access Controller - DMA - 9517A:

AO_H	CH#0	BASE AND CURRENT ADDRESS
$A1_H$		BASE AND CURRENT WORD COUNT
$A2_H$	CH#1	BASE AND CURRENT ADDRESS
$A3_H$		BASE AND CURRENT WORD COUNT
$A4_H$	CH#2	BASE AND CURRENT ADDRESS
$A5_H$		BASE AND CURRENT WORD COUNT
$A6_H$	CH#3	BASE AND CURRENT ADDRESS
$A7_H$		BASE AND CURRENT WORD COUNT
$A8_H$		
$A9_H$		
AA_H		
AB_H		
AC_H	REGISTER AND FUNCTION ADDRESSING	
AD_H		
AE_H		
AF_H		

5.4.2.12 Memory Address Map



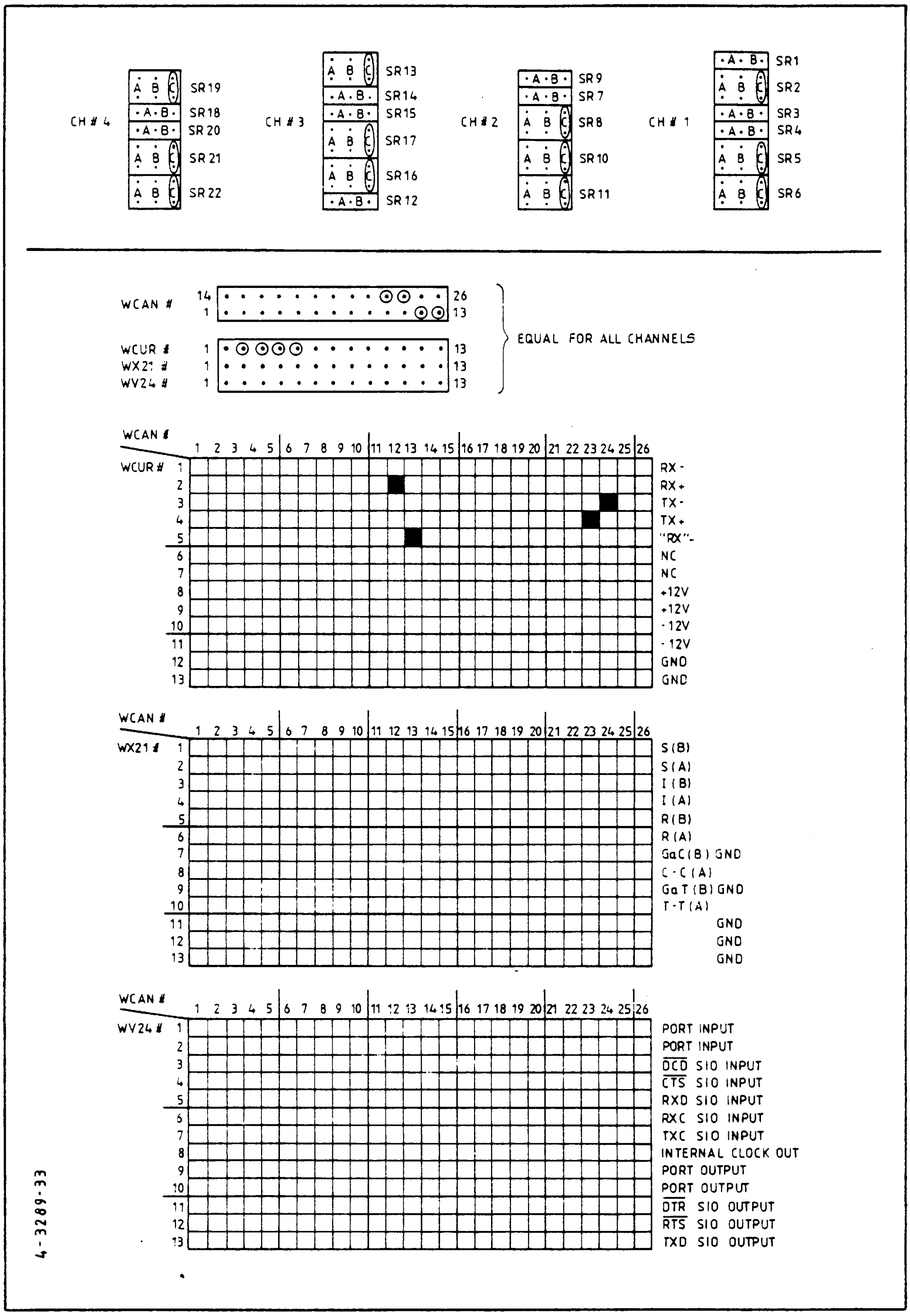
SCI MEMORY MAP

Note that the Multibus addresses specified above assume a base address of 00000H. Generally the base address would be some other number - as specified by the switch S3 (see page 5-131).

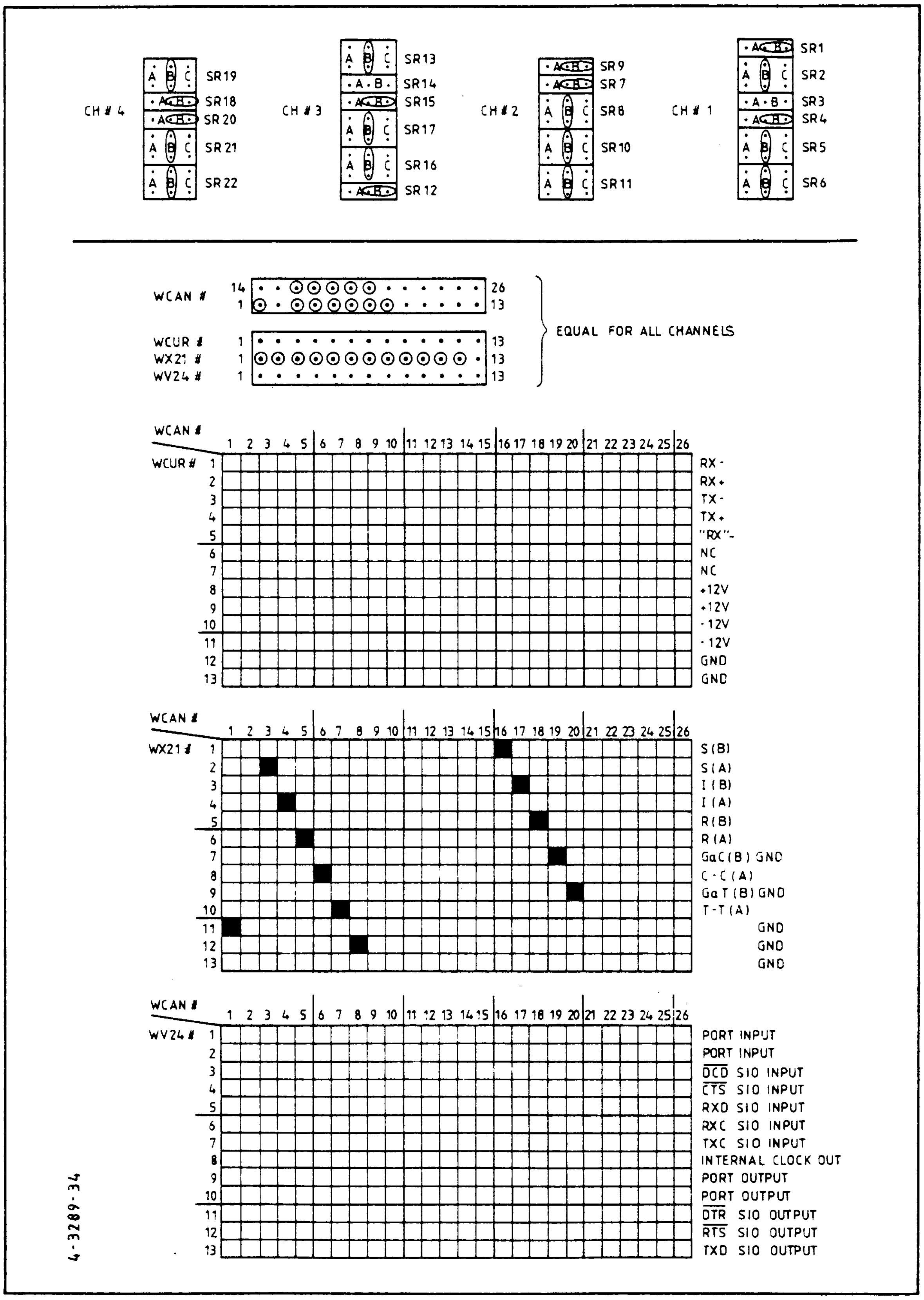
5.4.2.13 Jumper Settings and Wire-wrap connections

The following diagrams show the wire-wrap and jumper settings for various interface types. There are two additional jumpers not mentioned in the diagrams, numbers 23 and 24.

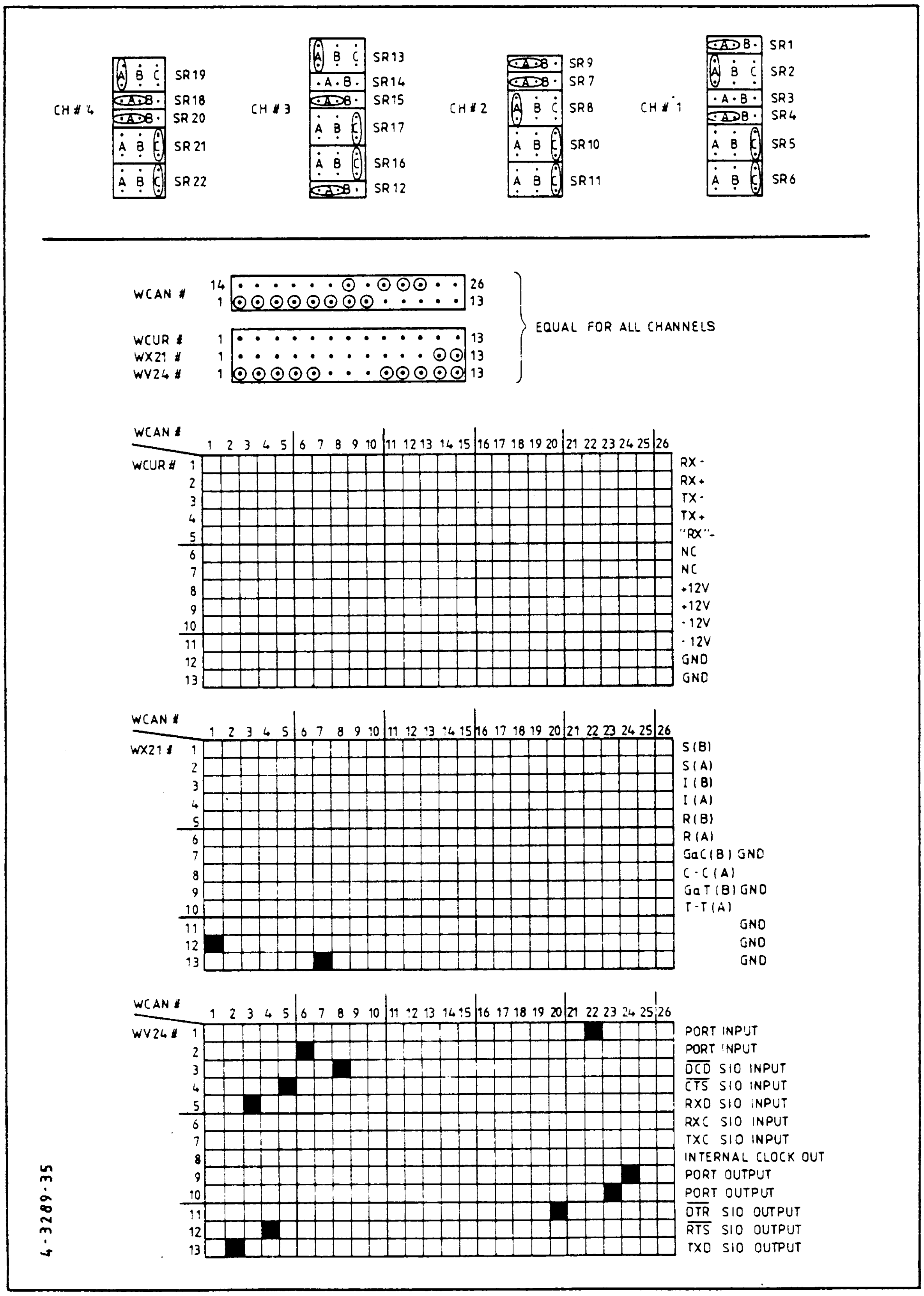
Jumper	Use
SR1 to 22	For setting the four channels to the desired interface type (V24, current loop, etc.) See the next 8 pages.
SR23	A: Input frequency to Baud rate generator (PIT) is 1.8432 MHz. B: frequency is 2 MHz. Should normally be set to A.
SR24	A: Memory type is 4K byte chips. (should be set to this) B: Not used.



WRAP/STRAP LIST FOR SCI 20mA CURRENT LOOP INTERFACE



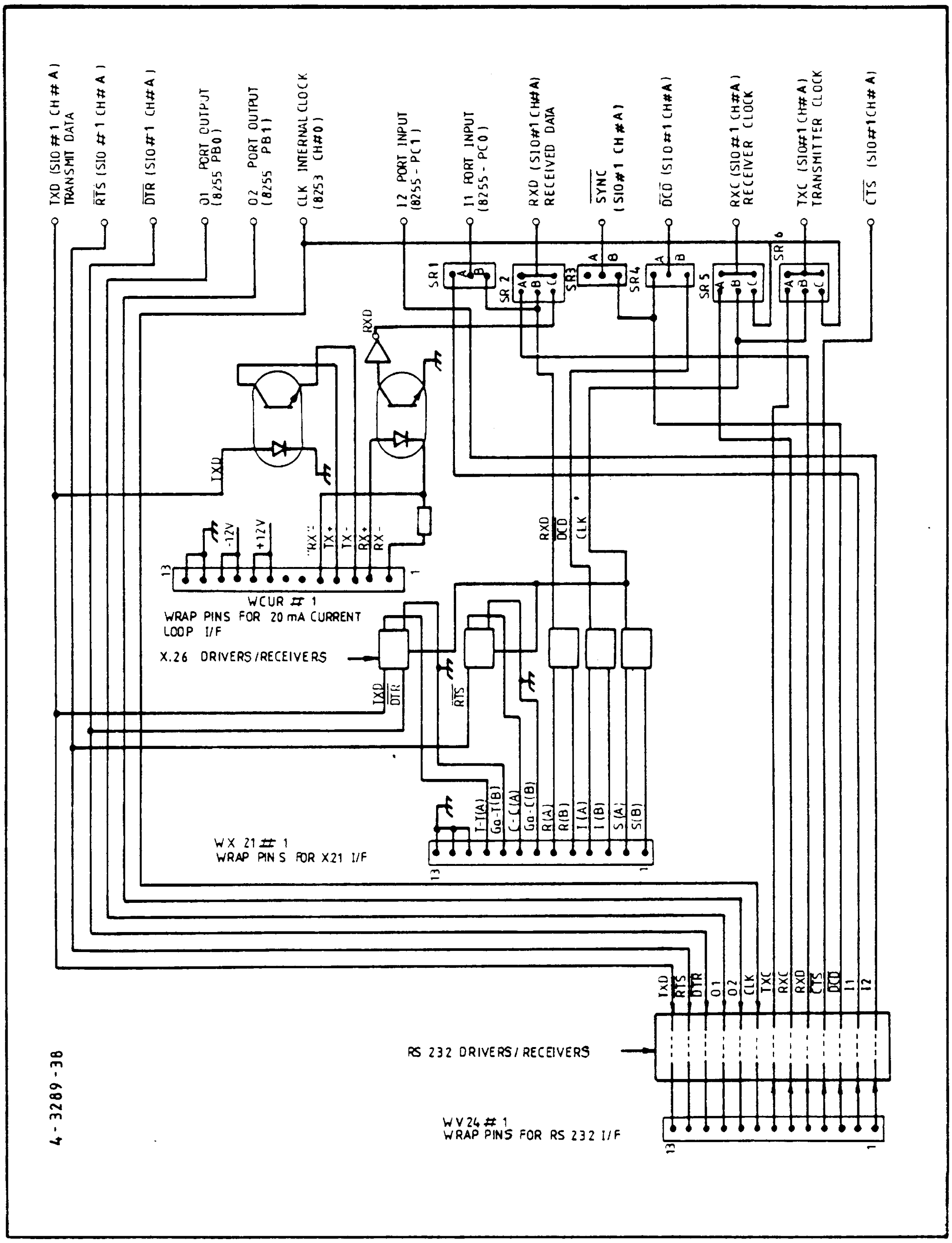
WRAP/STRAP LIST FOR SCI X21/DTE USING X.26. BYTE TIMING NOT INCLUDED



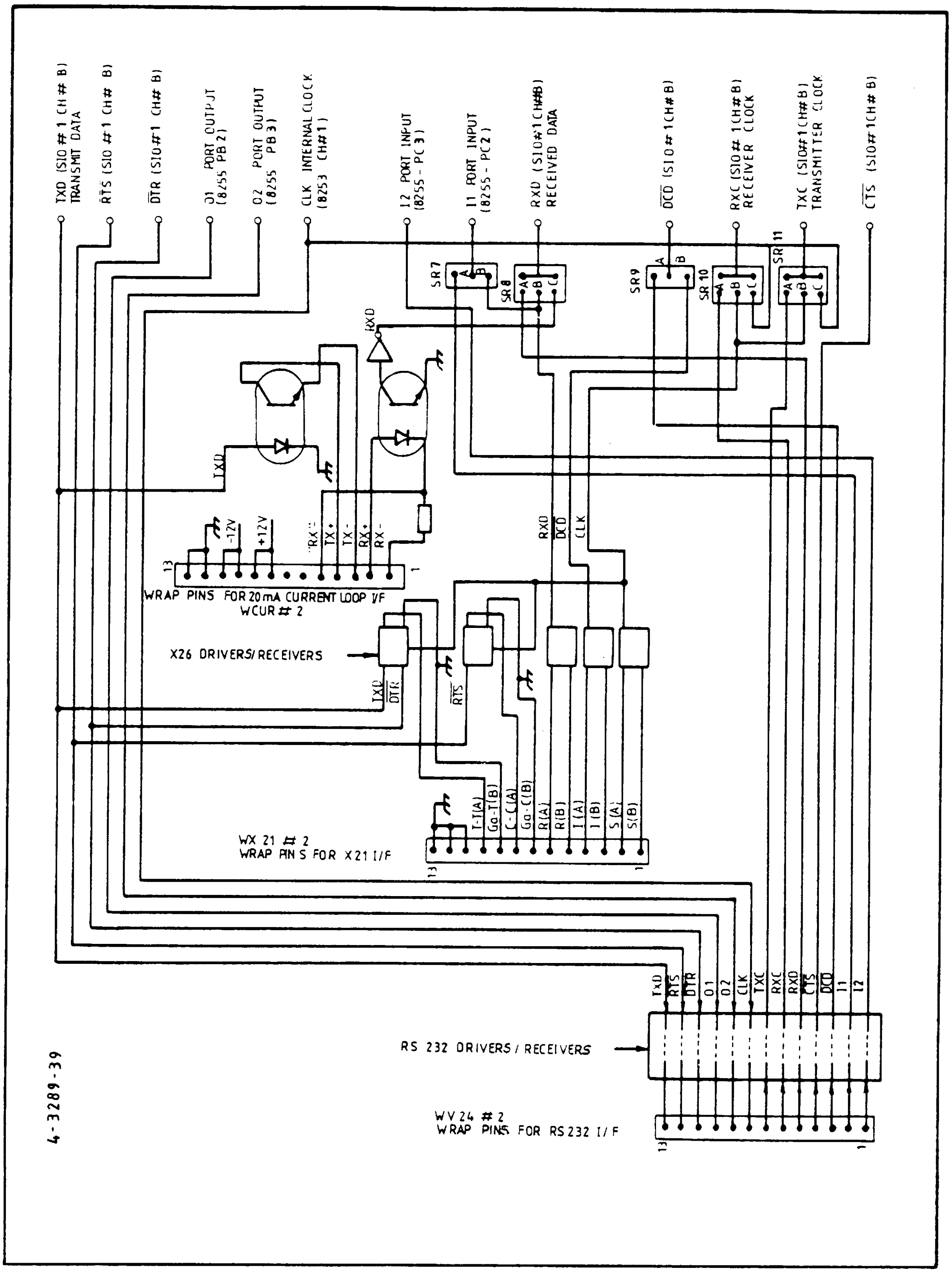
WRAP/STRAP LIST FOR SCI V24/DTE - ASYNCHRONOUS

5.4.2.14 Description of Wire-Wraps and Jumpers for the Four Channels

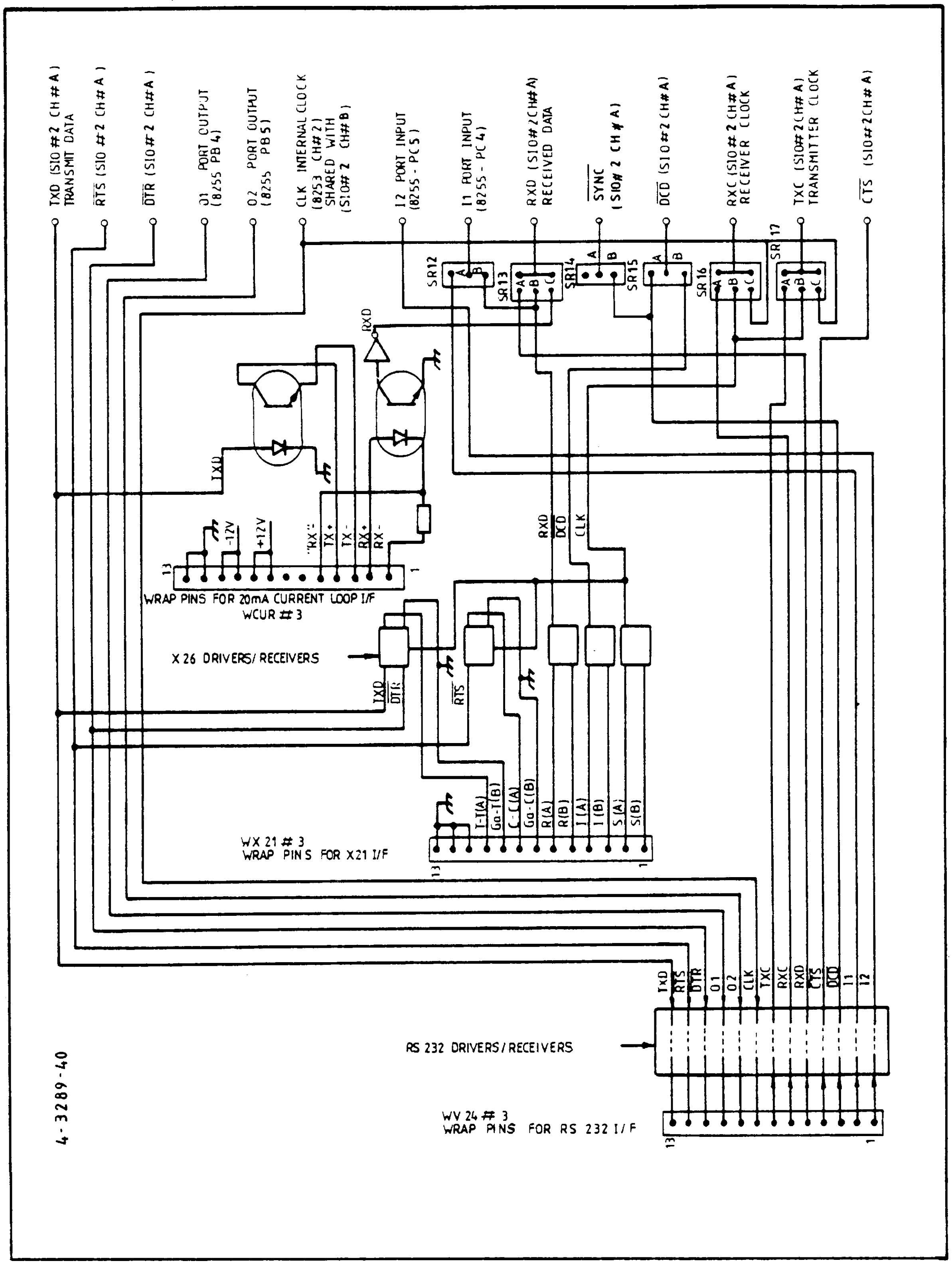
The details in the following diagrams may be of interest if a non-standard interface is required.



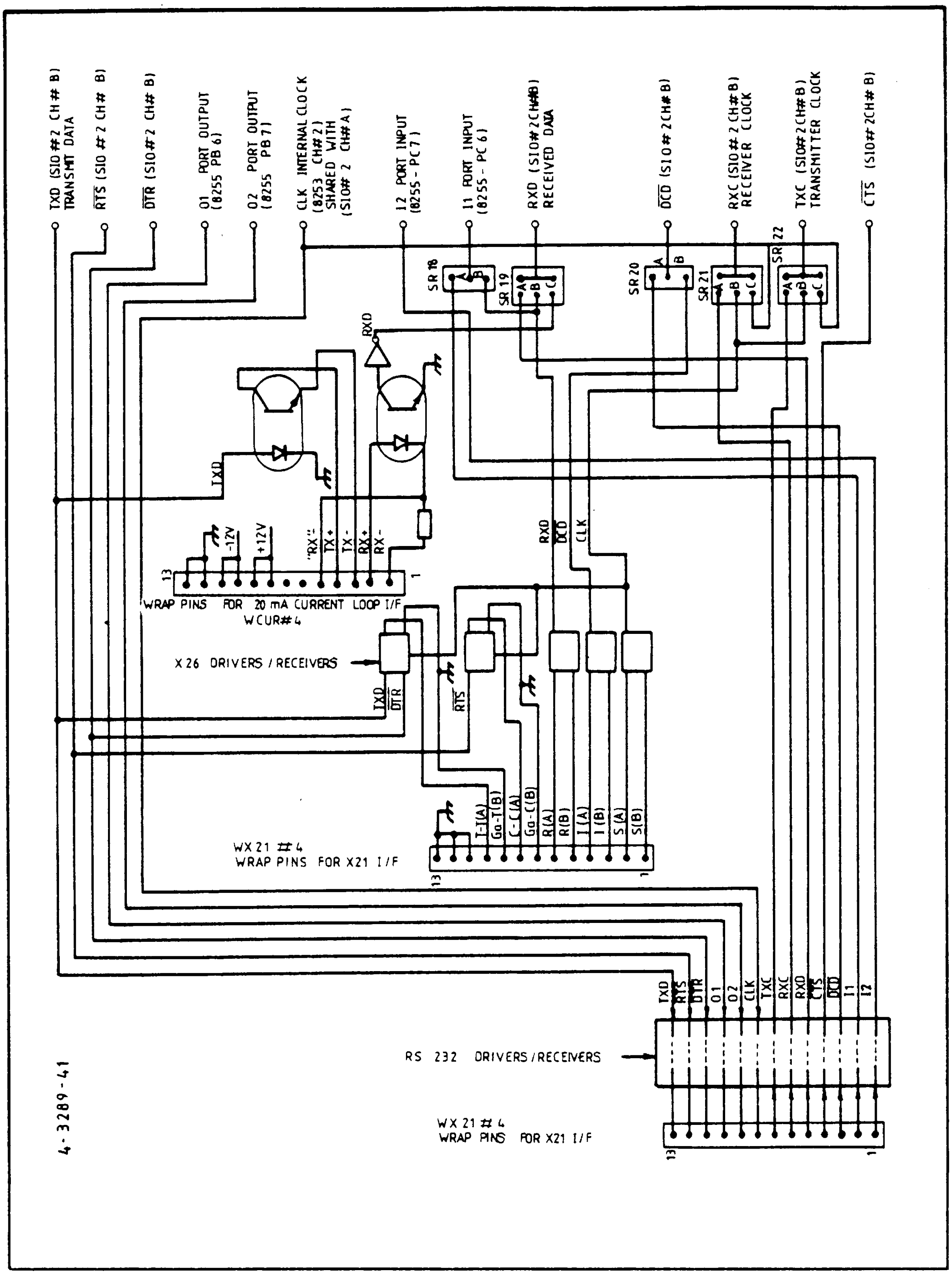
WRAP PIN/JUMPER AND ADDRESS DESCRIPTION FOR SERIAL LINE INTERFACE CHANNEL NO. 1


TXD (SIO #1 CH# B) TRANSMIT DATA
RTS (SIO #1 CH# B)
DTR (SIO#1 CH# B)
01 PORT OUTPUT (8255 PB2)
02 PORT OUTPUT (8255 PB3)
CLK INTERNAL CLOCK (8253 CH#1)
I2 PORT INPUT (8255 - PC3)
I1 PORT INPUT (8255 - PC2)
RXD (SIO#1 CH#B) RECEIVED DATA
DCD (SIO#1 CH#B)
RXC (SIO#1 CH#B) RECEIVER CLOCK
TXC (SIO#1 CH#B) TRANSMITTER CLOCK
CTS (SIO#1 CH#B)
SR7
SR8
SR9
SR10
SR11
RXD
TXD
-12V
+12V
RX
TX+
TX-
RX+
RX-
WRAP PINS FOR 20mA CURRENT LOOP I/F
WCUR # 2
X26 DRIVERS/RECEIVERS
RTS
DTR
CLK
T-T(A)
Ga-T(B)
C-C(A)
Ga-C(B)
R(A)
R(B)
I(A)
I(B)
S(A)
S(B)
WX 21 # 2
WRAP PINS FOR X21 I/F
RS 232 DRIVERS / RECEIVERS
WV 24 # 2
WRAP PINS FOR RS232 I/F
4-3289-39

WRAP PIN/JUMPER AND ADDRESS DESCRIPTION FOR SERIAL LINE INTERFACE CHANNEL NO.2



WRAP PIN/JUMPER AND ADDRESS DESCRIPTION FOR SERIAL LINE INTERFACE CHANNEL NO.3


TXD (SIO #2 CH # B) TRANSMIT DATA
RTS (SIO #2 CH # B)
DTR (SIO #2 CH # B)
O1 PORT OUTPUT (8255 PB 6)
O2 PORT OUTPUT (8255 PB 7)
CLK INTERNAL CLOCK (8253 CH #2) SHARED WITH (SIO # 2 CH # A)
I2 PORT INPUT (8255 - PC 7)
I1 PORT INPUT (8255 - PC 6)
RXD (SIO #2 CH #B) RECEIVED DATA
DCD (SIO #2 CH # B)
RXC (SIO # 2 CH # B) RECEIVER CLOCK
TXC (SIO # 2 CH # B) TRANSMITTER CLOCK
CTS (SIO # 2 CH # B)
SR 18
SR 19
SR 20
SR 21
SR 22
RXD
TXD
-12V
+12V
"RX"
TX+
TX-
RX+
RX-
WRAP PINS FOR 20 mA CURRENT LOOP I/F
WCUR#4
X 26 DRIVERS / RECEIVERS
TXD
DTR
RTS
RXD
DCD
CLK
T-I(A)
Ga-T(B)
C-C(A)
Ga-C(B)
R(A)
R(B)
I(A)
I(B)
S(A)
S(B)
WX 21 # 4
WRAP PINS FOR X21 I/F
TXD
RTS
DTR
O1
O2
CLK
TXC
RXC
RXD
CTS
DCD
I1
I2
RS 232 DRIVERS / RECEIVERS
WX 21 # 4
WRAP PINS FOR X21 I/F
4-3289-41

WRAP PIN/JUMPER AND ADDRESS DESCRIPTION FOR SERIAL LINE INTERFACE CHANNEL NO. 4

5.5 WINCHESTER DISK

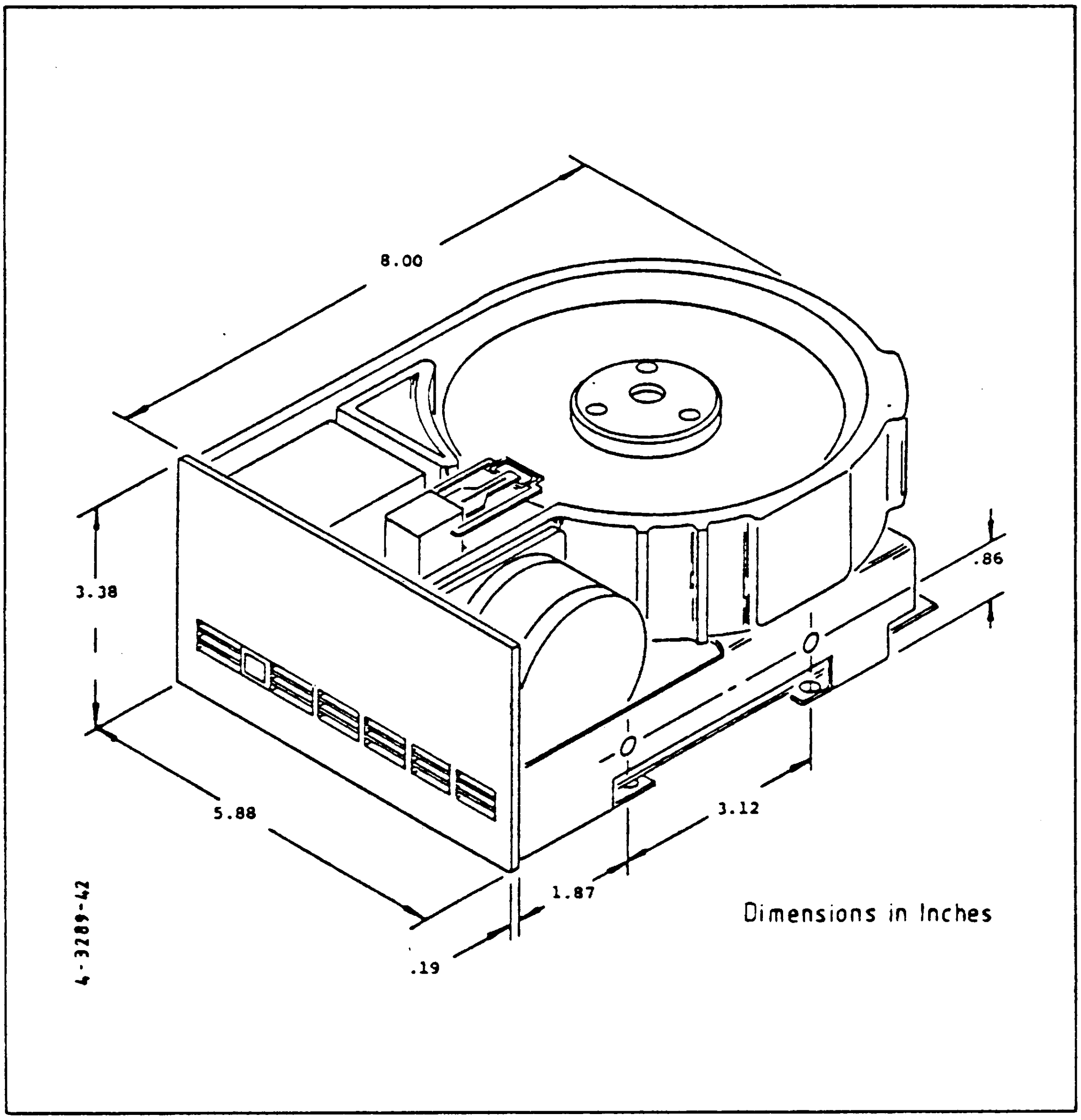
5.5.1 The Drive

The CR8 Winchester disk drive is a random access device with one or two non-removable 5 1/4" disks. Both surfaces on each disk are used, with one read/write head for each surface. There are two models available, one of 5 Mbytes and one of 10 Mbytes formatted capacity. Note that these specifications will change from time to time as improvements are made. Consult CR for the latest details.

Statistics	Option 31	Option 21	
Recording density	9074	7690	bpi
Track density	374	255	tpi
Access (average)	85	85	ms
Track to track access	3	3	ms
Latency	8.33	8.33	ms
Data transfer rate	5	5	megabits/s
Number of surfaces (heads)	4	2 or 4	
Tracks per surface (cylinders)	306	153	
Sectors per track	32	32	
Bytes per sector	512	512	
Unformatted capacity	12.76	6.38	megabytes
Formatted capacity	10.0	5.0	megabytes

The physical dimensions and positions of the mounting holes are identical to the industry standard for a mini-floppy drive, allowing a direct physical replacement in the same enclosure.

Each surface of each physical disk in the drive is treated by CP/M or MP/M as a logical drive; thus in CR8s containing one Winchester drive, that drive is referred to by software as drive A and B of the physical drive is the one-plate 21 option, or drives A, B, C, and D if Option 31 is installed.



WINCHESTER DISK DRIVE - DIMENSIONS (INCHES)

5.5.2 DISK CONTROLLER BOARD

The Winchester disk controller board is a Multibus compatible board able to support four disk drives. The board may be configured as either a master or a slave device on the Multibus interface.

A number of installation-dependent variables are selectable by jumpers W0 to W27. The board has the same configuration in the integrated workstation and the cabinet models of the CR8, except that jumper W4 is unmounted in the integrated workstation model. This configures the board as a master Device on the Multibus interface in the integrated workstation model (W4 unmounted). The board is configured as a slave device in the cabinet model (W4 mounted). The table overleaf summarizes the installed jumper options.

Note that the version of this board that was current in April, 1983 is so thick that it uses up two Multibus card positions (even though it is a single card). However, if installed in the Workstation in the rearmost position, there is enough room for it to only use one slot.

It is proposed that in May, 1983, a new controller board will be released. It will only occupy one Multibus slot in any CR8 and will have a SASI interface. The jumper settings, etc. in this section refer to the current (April 1983) version.

Jumper	Mounted	Not Mounted	Jumper	Mounted	Not Mounted
1	X		15		X
2	X		16		X
3	X		17		X
4	CABINET	Int.WS	18		X
5		X	19		X
6	X		20		X
7	X		21	X	
8		X	22		X
9		X	23		X
10		X	24		X
11		X	25		X
12	X		26		X
13		X	27	X	
14	X				

WINCHESTER CONTROLLER JUMPER LIST

JUMPERS	FUNCTION
4	Master/slave device
5-18	I/O base address
12, 17	Sector size
14, 15	Number of heads
19-16	Interrupt level

JUMPER-SELECTABLE OPTIONS

Master/slave device

The controller will be configured as a slave Device on the Multibus if jumper block W4 is mounted.

I/O base address

The CPU communicates with the controller board through a set of eight read/write ports, with addresses BASE+0 to BASE+7.

The I/O BASE address is selected by jumper W5 to W10. The board as supplied is configured with base address 80H.

BASE ADR.	W7	W8	W9	W10	W5	W6	BASE ADR.	W7	W8	W9	W10	W5	W6
00	0	0	0	0	0	1	80	1	0	0	0	0	1
08	0	0	0	0	1	0	88	1	0	0	0	1	0
10	0	0	0	1	0	1	90	1	0	0	1	0	1
18	0	0	0	1	1	0	98	1	0	0	1	1	0
20	0	0	1	0	0	1	A0	1	0	1	0	0	1
28	0	0	1	0	1	0	A8	1	0	1	0	1	0
30	0	0	1	1	0	1	B0	1	0	1	1	0	1
38	0	0	1	1	1	0	B8	1	0	1	1	1	0
40	0	1	0	0	0	1	C0	1	1	0	0	0	1
48	0	1	0	0	1	0	C8	1	1	0	0	1	0
50	0	1	0	1	0	1	D0	1	1	0	1	0	1
58	0	1	0	1	1	0	D8	1	1	0	1	1	0
60	0	1	1	0	0	1	E0	1	1	1	0	0	1
68	0	1	1	0	1	0	E8	1	1	1	0	1	0
70	0	1	1	1	0	1	F0	1	1	1	1	0	1
78	0	1	1	1	1	0	F8	1	1	1	1	1	0
0 = mounted, 1 = unmounted													

I/O BASE ADDRESS

Sector size

Jumpers W12 and W17 select the number of bytes per sector on the disk drive, either 256 bytes (32 sector per track) or 512 bytes (17 sectors per track). The board is supplied with a sector size of 512 bytes.

SECTOR SIZE	W12	W17
256	OUT	IN
512	IN	OUT

Number of heads

Jumpers W14 and W15 specify the number of heads in the disk drive (4, with the supplied Winchester disk drive.

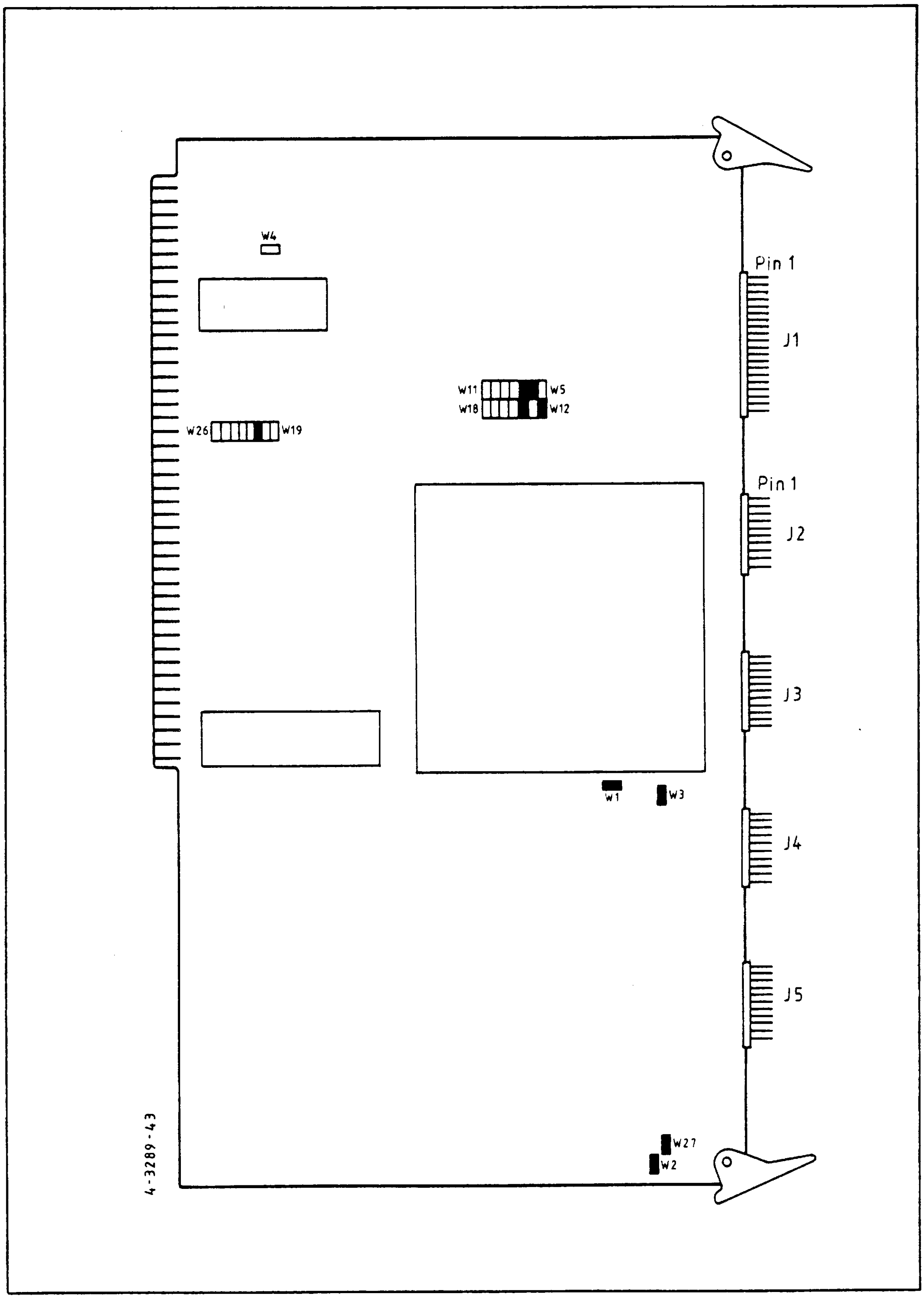
NUMBER OF HEADS	W14	W15
2	IN	IN
4	IN	OUT
6	OUT	IN
8	OUT	OUT

Interrupt Level

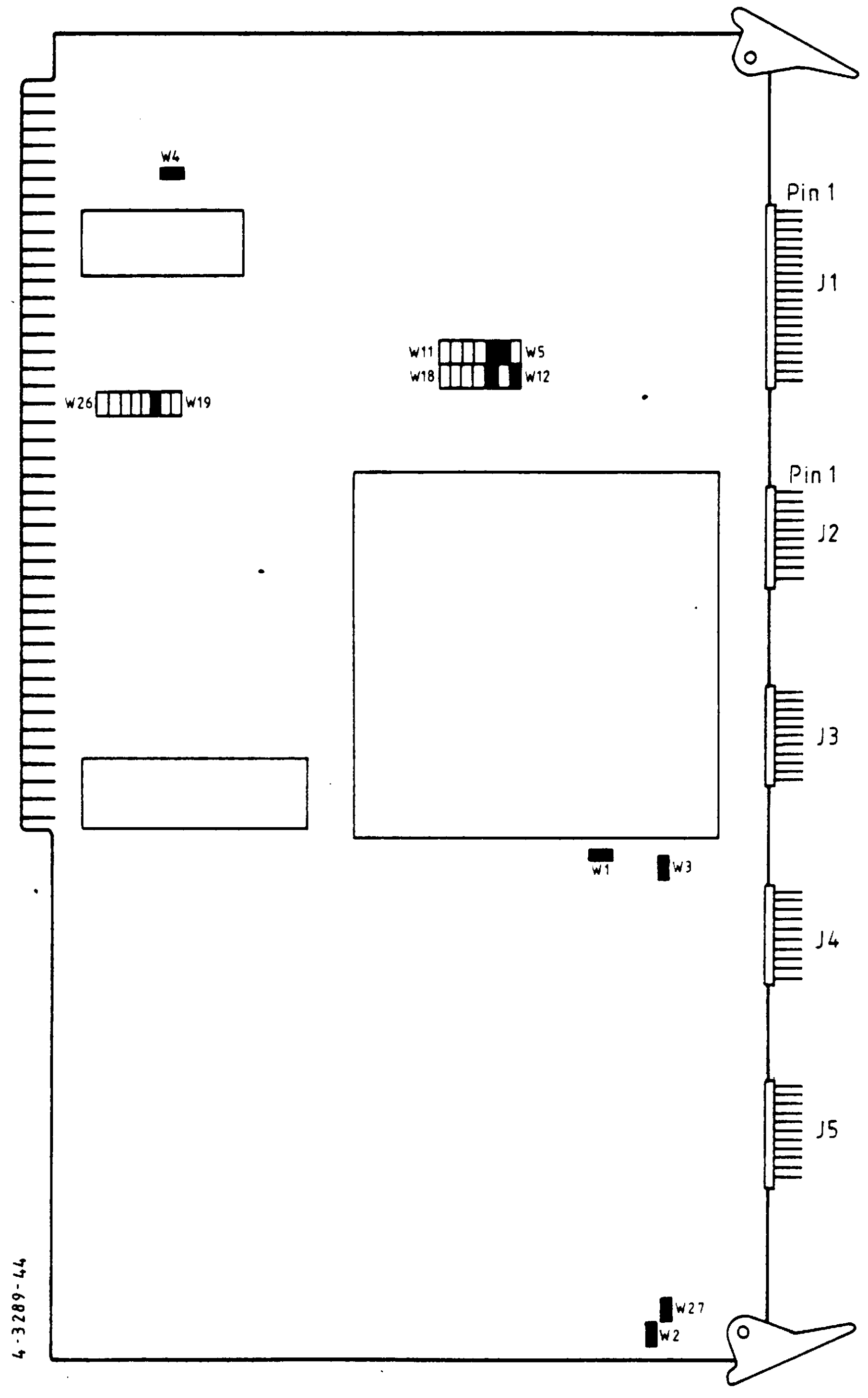
The interrupt request level is selected by mounting a jumper on one of the eight jumper on one of the eight jumper blocks W19 to W26 (Int. 0 to Int.7). The supplied board has a jumper block on W21 (Int. 2).

Cabling

The controller board requires one 34-core flat cable that is tapped by each drive, (in "daisy-chain" fashion), installed in position J1. This carries commands for all drives. In addition there must be one 20-core data cable for each connected drive installed in positions J2 to J5. The cables should be installed with the correct orientation, otherwise the drives or controller may be damaged. The board layout, with correct pin orientation, is shown overleaf. See also the illustration on page 5-9.



JUMPER POSITIONS FOR WINCHESTER CONTROLLER BOARD - WORKSTATION MODEL



JUMPER POSITIONS FOR WINCHESTER CONTROLLER BOARD - CABINET MODEL

5.5.3 FORMATTING

The Winchester disk is "soft sectored", which means that sectors are identified from specially recorded signals placed there by a formatting program. Normally the Winchester disk will not need reformatting, but the process is described below in case a hardware error has disrupted the disk. Note that the operating systems are configured to treat the Winchester drive as four separate drives A - D. Each disk surface is a different CP/M or MP/M drive and is formatted independently.
The formatting program, FORMAT6, runs under CP/M or MP/M. The operating instructions are as follows. (FORMAT12 is similar and is used for the Option 31 Winchester; FORMAT6 for the Option 21 Winchester.)

1. Type FORMAT6 <CR> after the CP/M (or MP/M) prompt.

   ```
   E >FORMAT6
   ```

   The FORMAT6 program is loaded from floppy disk drive E.

2. The program displays the following message:

   ```
   CHRISTIAN ROVSING A/S - CR8 DISK FORMATTING UTILITY
   FOR SEAGATE ST506 HARD DISK/TANDON TM100-4 FLOPPY

   SELECT OPTION-
   A) FORMAT HARD DISK A:
   B) FORMAT HARD DISK B:
   C) FORMAT HARD DISK C:
   D) FORMAT HARD DISK D:
   E) FORMAT MINI FLOPPY E:

   OPTION(A,B,C,D, OR E) =
   ```

   Enter option A,B,C or D to format the Winchester drive (e.g. B)

   ```
   OPTION (A,B,C,D, OR E) = B
   ```

3. The program now displays a warning that all data on the disk will be erased. This is your last chance to check that the selected drive is the one you really want to format. Type G to proceed with formatting or N to stop.

   ```
   ALL INFORMATION ON DISK WILL BE ERASED.
   TYPE "G" TO FORMAT OR "N" TO STOP (G/N) G
   ```

4. Disk formatting will now begin.

   ```
   FORMATTING COMPLETE
   ```

5. You may now format another drive by entering Y in response to the next message. The program will return to step 2. Enter N to exit from the formatting program.

   ```
   MORE DISKS TO BE FORMATTED? (Y/N) N
   E>
   ```

   Formatting removes all files, and erases the contents of the reserved tracks, on the selected drive (i.e. the selected disk surface).

The formatting and Sysgen programs will shortly be subsumed into the Toolbox utility.

5.6 FLOPPY DISK DRIVE

The floppy disk drive used in the CR8 is the standard size for a mini-floppy drive (the same size as the Winchester described in the previous section). The heads are ferrite/ceramic, bonded in glass which gives a head life of over 20,000 hours. Recording is by Modified Frequency Modulation (MFM). Diskettes must be of the standard 5 1/4" type with a single index hole. Soft-sectoring is employed - in other words the different areas of the disk are defined by writing special signals to the disk. This is known as formatting and was described in section 3.3. The care and use of diskettes was described in section 3.2. Note that the controller logic for one or two floppy diskette drives is included within the MP^2 card - please refer to the index. Specification for the drive follows. Note that this may change from time to time, as improvements are made. Consult CR for the most up-to-date information.

Media	Industry-Standard 133.4 mm (5.25 inch) diskettes
Dimensions	
Height	85.85 mm (3.38 inches)
Width	149.1 mm (5.87 inches)
Length	203.3 mm (8.00 inches)
Weight	1.45 Kg (3.2 pounds)
Head Setting Time	15 msec (Last Track Addressed)
Error Rates, Maximum	
Soft Read	1 per 10^9 bits, Recoverable
Hard Read	1 per 10^{12}, Nonrecoverable
Head Life	20,000 Hours, Normal Use
Media Life (For Reference Only)	3.6×10^6 Passes Per Track
Disk Speed	300 rpm $\pm$ 1.5% Long Term
Instantaneous Speed Variation	$\pm$ 3%
Start/Stop Time	250/150 msec., Maximum
Transfer Rate	FM: 125,000 BPS MFM: 250,000 BPS
Bytes per Disk, unformatted, double density	1,000,000
Recording Mode	MFM
Power	
+ 12 VDC	$\pm$ 0.6 VDC at 900 mA (Average Maximum)
+ 5 VDC	$\pm$ 0.25 DC at 600 mA (Average Maximum) with 100 mV Peak-to-Peak Ripple

FLOPPY DISK DRIVE SPECIFICATION

5.7 SYSTEM BOOTING

When the Reset button on the CR8 is pressed (or power is switched on) or the Workstation's Reset button is pressed, then the MP^2 will hold the Multibus Reset signal low for about 40 msec. This is a signal to all boards in the Multibus to set themselves to a known, initialized state.

The MP^2 board executes a self-test program at boot time. If the test fails, a red LED will light on the board, and one of the following messages will appear on the screen:

```
Z80 MEM. ERR. WRITE = xx    READ = yy
```

or

```
NO CTC INTERRUPTS
```

The sequence of events for bringing up the system after a reset will depend upon the operating system in use. Please refer to the appropriate section below. Note also 'Selecting the Boot Source' on page 3-14.

5.7.1 CP/M 2.2

1. The system PROM executes memory and other hardware tests.

2. CP/M is read in from the two reserved "System" tracks on disk drive B (Winchester) or A in the dual floppy system.

3. CP/M is entered at its entry point in the BIOS (Basic I/O System).

4. A start-up message is output on the console:
```
CR8 BOOTLOADER VERS. 2.0
CHRISTIAN ROVSING A/S 64K CP/M VERS. 2.2 820105
SEAGATE ST506 HARD DISK / 5 1/4 MINI FLOPPY BIOS, V.1.0
B>
```
   and the system is ready for use.

The above procedure is known as a "cold boot". A subset of this may be carried out by pressing CONTROL-C on the console. This is known as a "warm boot". In this case the procedure starts within step 3. Only the BDOS (Disk Operating

System), and CCP (Console command processor) are read in - the BIOS is assumed to be still there. The warm start is used after using certain utilities such as the debugger which overwrite the CCP code.

5.7.2 CP/M 86

1. The system PROM executes memory and other hardware tests.

2. SCRIMP/M is read in from the two system tracks on drive B into Z80A memory.

3. CP/M 86 is read in by SCRIMP/M into the 8088's memory from the two system tracks on drive A.

4. CP/M is entered at its entry point in the BIOS (Basic I/O System).

5. A start-up message is output on the console:

```
CR8 BOOTLOADER VERS. 2.0
CHRISTIAN ROVSING A/S 64K CP/M VERS. 86 820105
SEAGATE ST506 HARD DISK / 5 1/4 MINI FLOPPY BIOS, V.1.0
B>
```

   and the system is ready for use.

The above procedure is known as a "cold boot". A subset of this may be carried out by pressing CONTROL-C on the console. This is known as a "warm boot". In this case the procedure starts within step 4. Only the BDOS (Disk Operating System), and CCP (Console command processor) are read in - the BIOS is assumed to be still there. The warm start is used after using certain utilities such as the debugger which overwrite the CCP code.

5.7.3. MP/M II

1. The system PROM executes memory and other hardware tests.

2. An MP/M loader routine is read in from the two reserved System tracks on disk drive B (Winchester) or A (Dual floppy systems). This is necessary because the system itself is too big to fit onto the two system tracks. The loader is read into an area of memory which is to be used later on for user programs.

3. The loader is merely a cut-down operating system that is nevertheless capable of understanding the disk filing system; thus it is able to execute the next stage, namely, reading in MP/M from a named file - MPM.SYS:

4. SCI boards are booted as appropriate (see also "Booting the SCI on page 5-128)

5. MP/M is entered at its entry point in the BIOS (Basic I/O System).

6. A start-up message is output on all the consoles:
   ```
   CR8 BOOTLOADER VERS. 2.0
   CHRISTIAN ROVSING A/S 64K MP/M VERS. II 820105
   SEAGATE ST506 HARD DISK / 5 1/4 MINI FLOPPY BIOS, V.1.0
   0B>
   ```
   and the system is ready for use.

The above procedure is known as a "cold boot". A subset of this may be carried out by pressing CONTROL-C on the console. This is known as a "warm boot". In this case the procedure starts within step 5. Only the BDOS (Disk Operating System), and CCP (Console command processor) are read in - the BIOS is assumed to be still there. The warm start is used after using certain utilities such as the debugger which overwrite the CCP code.

5.7.4 MP/M 86

1. The system PROM executes memory and other hardware tests.

2. SCRIMP/M is read in from the two reserved system tracks on Winchester drive B, to the Z80As memory.

3. SCRIMP/M reads in an MP/M loader routine from the system tracks on drive A. This is necessary because the system itself is too big to fit onto the two system tracks. The loader is read into an area of memory which is to be used later on for user programs.

4. The loader is merely a cut-down operating system that is nevertheless capable of understanding the disk filing system; thus it is able to execute the next stage, namely, reading in MP/M from a named file - MPM.SYS on drive A.

5. SCI boards are booted as appropriate (see also "Booting the SCI" on page 5-128).

6. MP/M is entered at its entry point in the BIOS (Basic I/O system).

7. A start-up message is output on all the consoles:
   ```
   CR8 BOOTLOADER VERS. 2.0
   CHRISTIAN ROVSING A/S 64K MP/M VERS. 86 820105
   SEAGATE ST506 HARD DISK / 5 1/4 MINI FLOPPY BIOS, V.1.0
   0B>
   ```
   and the system is ready for use.

The above procedure is known as a "cold boot". A subset of this may be carried out by pressing CONTROL-C on the console. This is known as a "warm boot". In this case the procedure starts within step 6. Only the BDOS (Disk Operating System), and CCP (Console command processor) are read in - the BIOS is assumed to be still there. The warm start is used after using certain utilities such as the debugger which overwrite the CCP code.

5.8 ADDING MORE MEMORY

It is simple to add extra RAM to the CR8 on Multibus-compatible cards. The Multibus address must be set up on the card (see the documentation supplied by the memory supplier). Any unused address range can be chosen, provided it is catered for by the memory board manufacturer and that addresses never exceed Hex FFFFF (1 megabyte). Remember that the MP^2 occupies 256 Kbytes, normally starting at Multibus address 0, and that SCI boards occupy 32 Kbytes. The Winchester controller does not use any memory addresses.

Remember, of course, to disconnect the CR8 from the mains before removing or inserting any boards, indeed before opening the cabinet or workstation case.

DIP switches 3 and 4 on the MP^2 card will have to be set appropriately so that the operating system can recognize the existance of the new memory. Details of switch settings are on page 5-117. The settings available limit the choices of memory size to 128, 256 or 512 Kbytes (excluding MP^2 and SCI memory).

If using the single user CP/M operating system then nothing further need be done. If using MP/M however, it will be necessary to generate a new system using the GENSYS command. Remember always to make a copy of the old operating system before running GENSYS. Full details of GENSYS are given in Digital Research's MP/M-86 or MP/M II System Guide. Remember also not to add all the new memory as one partition but to break it up into smaller units. This will enable more efficient use to be made of the new memory.

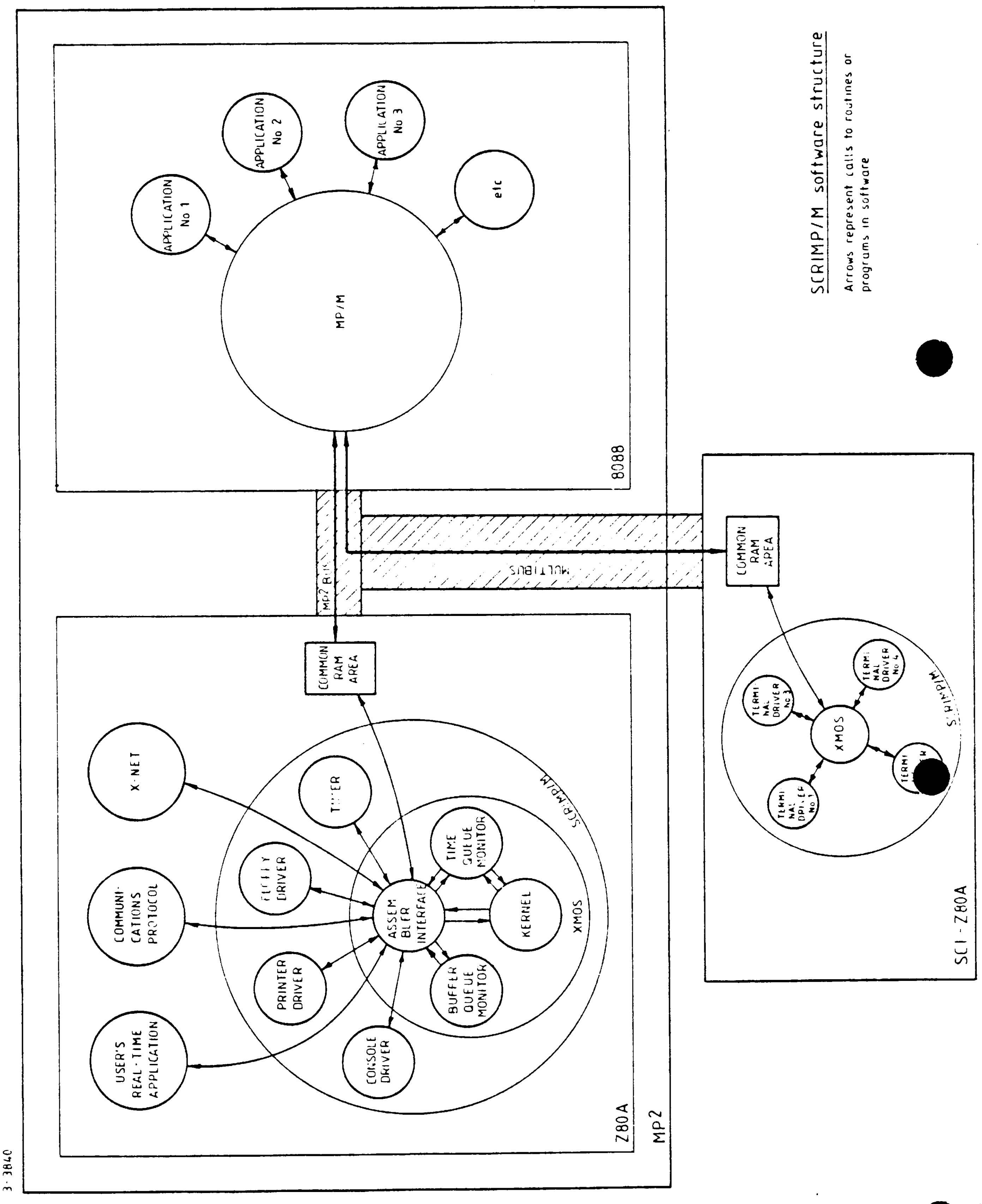
5.9 SCRIMP/M

This section is indended for users who want to write their own real time systems software to run in the Z80A on the MP^2 and/or SCI boards - for instance communications protocol handling. A knowledge of real-time operating systems architecture and assembler programming is assumed.

An introduction to SCRIMP/M was given in section 2.2. That section dealt mostly with its function in the CR8, namely, doing most of the I/O processing (thus releasing CPU time in the 8088 for other work.) This current section will concentrate more with the structure of SCRIMP/M, leading onto a more detailed guide to programming under SCRIMP/M.

It can be seen from the 'Software Structure' diagram overleaf that the heart of SCRIMP/M is XMOS (and the heart of XMOS is known as the Kernel.) XMOS is a general-purpose operating system used in various microprocessors and applications throughout CR's product range. This particular application of XMOS, namely the handling of I/O in the CR8 microcomputer, calls upon very few of XMOS' facilities (for instance the facility to create 'Child Processor' is not used.) However, all four XMOS code modules are present in the CR8 implementation and may be called upon as required. Of course, some of the more advanced XMOS features are used when the CR8 is used for communications protocol handling, X-Net, etc.

The name 'SCRIMP/M' derives from the fact that it services MP/M rather than replacing it in any way. SCRIMP/M contains no disk filing routines, so could not be used by itself as an environment for the normal types of end-user software such as text processing, accounting packages, etc. SCRIMP/M takes over the device-dependent parts of I/O processing which would normally be done by MP/M or CP/M. In the latter case of course this does not amount to a great deal of work since CP/M only handles one user at a time and maybe four I/O devices. However, SCRIMP/M really comes into its own when using MP/M since the volume of I/O is so much greater.



SCRIMP/M software structure

Arrows represent calls to routines or programs in software

5.9.1 XMOS

Although SCRIMP/M only uses a small subset of XMOS (namely the functions CONFIG, CURRENT and RELEASE), the whole of XMOS is described here. This is because the other facilities may be required by users who need to write their own communications protocol handling software, complex device drivers, or whatever. There now follows a description of what XMOS does and how it does it; after that are given details of the user interface. This latter section will not make much sense unless the former section (how XMOS works) has been understood.

5.9.1.1 THE PURPOSE OF XMOS

The purpose of XMOS, in the CR8, is to implement multiprogramming within the SCRIMP/M operating system (whose purpose has been described at the beginning of section 5.9). XMOS fulfills the following functions:

- Implementation of software processes - that is, dividing the work to be done into discrete tasks which are handled separately.
- Communication between these processes.
- Synchronization of the processes.
- Timing out of unsatisfied conditions.
- CPU management - i.e. sharing out CPU time sensibly between the competing processes - otherwise known as scheduling.

In effect, the last of these functions, CPU management, includes all the others. By using waiting points in the coding of the different processes, and by shifting the use of the CPU rapidly between processes, it is possible for it to appear that several processes are executing simultaneously. In other words, several other processes can run while one process is waiting for an external event to oocur - such as a disk drive completing an I/O operation, or a person pressing RETURN on the console, or the time becoming 8:42 a.m.

Of course it would be useful if the CPU could do the work for which it was purchased such as handling an X25 protocol, as well as scheduling the timer, synchronising the creation of processes and all those other vital but unproductive duties. There should be plenty of CPU time left for the useful work if the user's code is written sensibly. Note that the user is responsible for allowing XMOS to 'do its own thing' from time to time. Thus if the user's process runs for ever (which a real-time process often will), the user process must have at least one release point (see RELEASE in the User Interface section) or, preferably, the Time Slice option must be specified. If neither of these is done, and the user's program has disabled interrupts, then the program could go into an endless execution loop whereby it would <u>completely</u> monopolize the CPU indefinately (until 'Reset' was pressed).

5.9.1.2 SOME TERMS EXPLAINED

Note that most of these terms are discussed further in later sections.

A **process** is the entity that is scheduled by XMOS. It includes not only a program but data, register values, etc. - in fact everything pertaining to one task or job. A process is defined by a data area in memory called a **process descriptor.** This includes the addresses of: the next instruction to be executed, the stack, the reserved data, and tables of other fixed addresses. The process descriptor also contains the current status information for the process.

There are two types of process - **parent** and **child.** Parent processes are created at system start-up time by the CONFIG routine and remain permanently in the system, whereas child processes are created dynamically (i.e. during run-time) by a parent process, and may be deleted by the parent at any time. Note that child processes are invisible to all other processes but their own parents, and that XMOS itself does not know of their existance except when they are actually running - when they seem like any other process (except for one bit in the process descriptor which indentifies them as children). Note also that in XMOS, you don't have to have children to be a parent! (That is, all processes defined by CONFIG may be termed 'parent processes'.) Furthermore, due to the unusual biological properties of microcomputers, only one parent is required in order to generate children. Child processes always occupy a part of their parent's memory (they have none of their own).

An example of the allocation of work between parents and children might be in a communications system, where one parent would handle one level of protocol for one channel. If the level could be divided into several activities, each of these activities could be handled by a different child process.

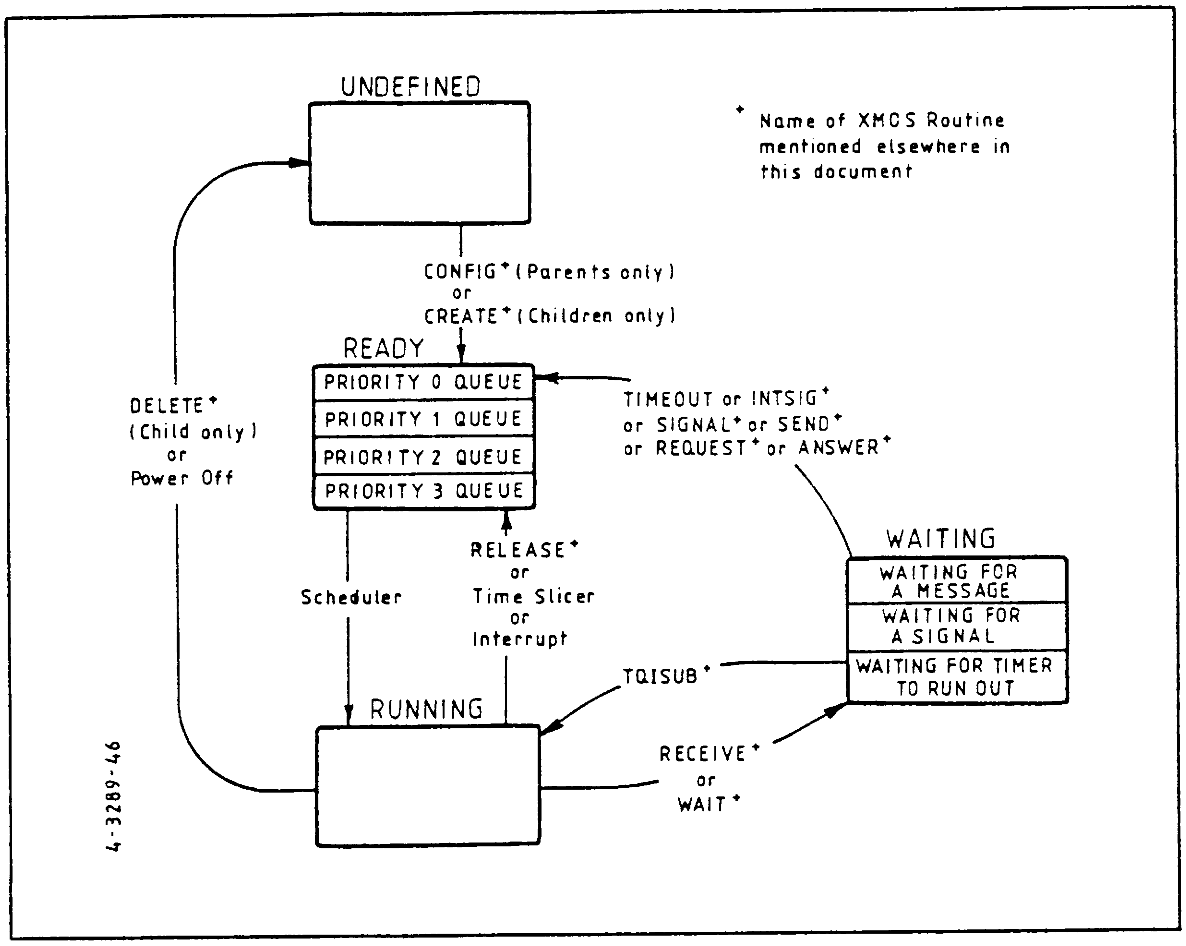
A **program** is a sequence of instructions, assembled to perform a certain task. A program may be shared by several processes.

A process is assigned one of four **priorities** when it is created. 0 is the highest, 3 the lowest. The priority is used by the scheduler to decide which of the ready processes is to be the next running process.

XMOS provides two modes of inter-process communication: **signal/wait** and **messages.** The signal/wait mechanism is simply a means whereby a process can be put to sleep (Wait) or woken up again by another process (Signal). If any more complex command is to be sent between processes then the Message mechanism must be used. This is similar to signal/wait except that an opcode is sent between the two processes. This opcode is the 'message' and has some pre-defined meaning which will vary with the application.

Signal/wait commands and messages are queued by XMOS using semaphores - **general semaphores** and **message semaphores** respectively. A semaphore is an 8-byte queue descriptor which contains the addresses of the first and last queue entries plus a byte to say what sort of queue it is. The queue entries referred to are either process descriptors (either type of semaphore) or messages (message semaphores). XMOS includes a set of routines to manipulate such queues and their semaphores. It is called the **BQM** (buffer queue monitor); it is an optional piece of code.

5.9.1.3 PROCESS STATES


UNDEFINED
* Name of XMCS Routine mentioned elsewhere in this document
CONFIG* (Parents only) or CREATE* (Children only)
READY
PRIORITY 0 QUEUE
PRIORITY 1 QUEUE
PRIORITY 2 QUEUE
PRIORITY 3 QUEUE
DELETE* (Child only) or Power Off
TIMEOUT or INTSIG* or SIGNAL* or SEND* or REQUEST* or ANSWER*
RELEASE* or Time Slicer or Interrupt
Scheduler
WAITING
WAITING FOR A MESSAGE
WAITING FOR A SIGNAL
WAITING FOR TIMER TO RUN OUT
TQISUB*
RUNNING
RECEIVE* or WAIT*
4-3289-46

PROCESS STATES

The figure above shows the four possible states that a process can be in - undefined, ready, running, or waiting.

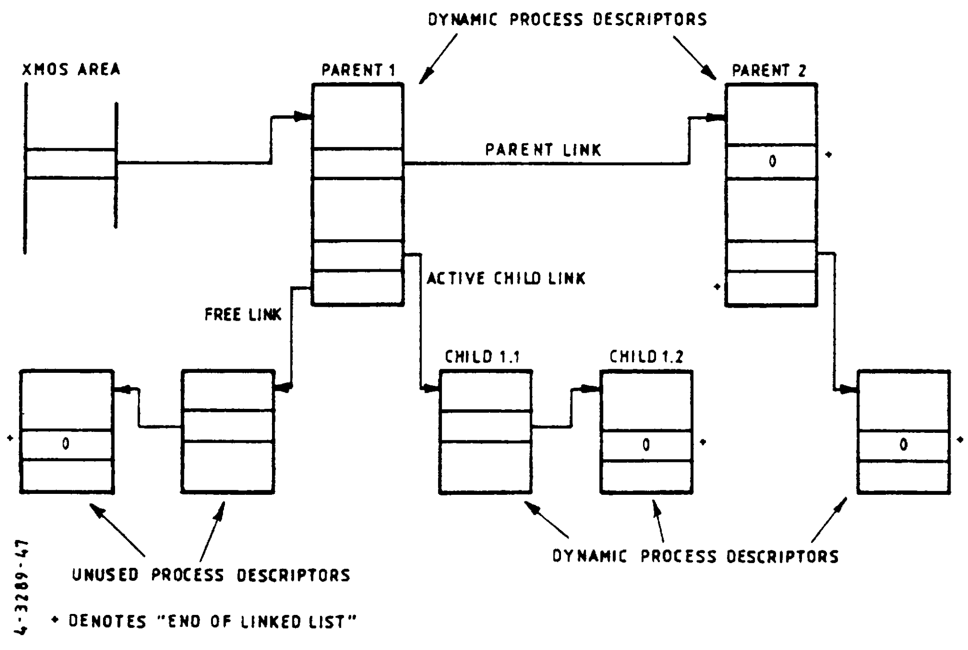
Only child processes can be in the "undefined" state in a working system. The state of a process is recorded in the process descriptor. When a process is created, it is in the "ready" state and the system scheduler will change it to "running" when it has the highest priority of all the "ready" processes.

A process will leave the running state when a waiting point routine is called (RELEASE, RECEIVE, WAIT) or when it is interrupted by the time slicer.

If a process is in the "wait" state, it will become "ready" when the running process sends a message or signal to it or if it is timed out by the timer. The process will then run when the scheduler decides to run it. Of course only one process can be in the 'running' state at any moment, though several could be in any of the other three states.

5.9.1.4 PROCESS HIERARCHY

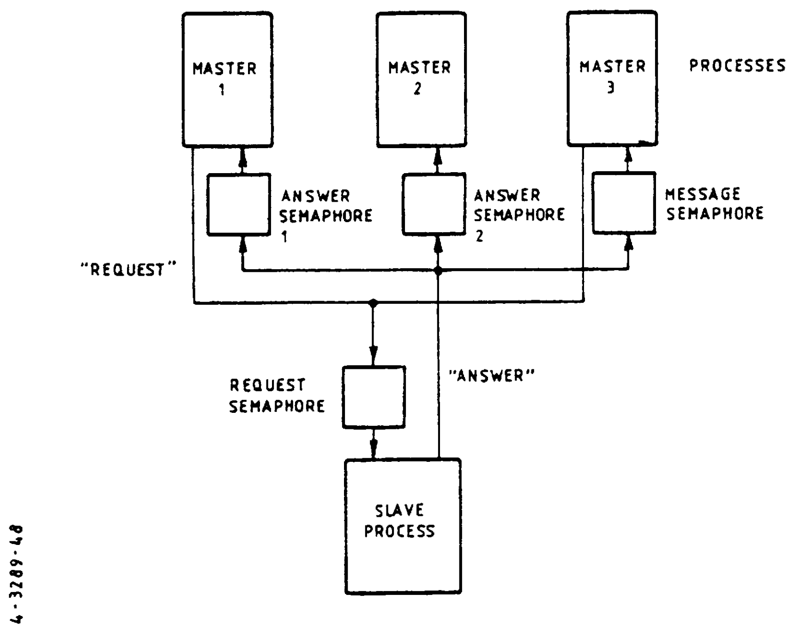
The relationship between the two types of process was explained in section 5.9.1.2 (parent and child processes). Note that at CONFIG time the parent process includes some blank process descriptors for use by children if they are created. A child process can delete itself by calling DELETE. This will free its process descriptor.


DYNAMIC PROCESS DESCRIPTORS
XMOS AREA
PARENT 1
PARENT 2
PARENT LINK
0
*
ACTIVE CHILD LINK
FREE LINK
CHILD 1.1
CHILD 1.2
0
UNUSED PROCESS DESCRIPTORS
DYNAMIC PROCESS DESCRIPTORS
* DENOTES "END OF LINKED LIST"
4-3289-47

LINKED LISTS OF PROCESS DESCRIPTORS IN AN EXAMPLE XMOS SYSTEM

XMOS keeps track of its parent processes by means of a linked list, each parent has two linked lists of its own - one of active child processes and one of free process descriptors (i.e. potential child processes). The figure below shows a system configured with two parent processes and three child processes. Parent 1 could create up to two more children, but parent 2 has been configured in such a way that it can never have more than one child.

Very often one process can be used as a 'slave' process that performs a specific task (the same one every time) for various 'master' processes. This would be analogous to various programs calling a subroutine in a library that maybe works out square roots. The difference is that by having a separate process for working out square roots, it means that this work can be done while the master process is getting on with some other work. Another example of such a slave process would be a line driver. In fact, the majority of processes in a system might well be slave processes, each with its own specific task. The synchronisation of master and slave activities in XMOS is implemented by means of semaphores. These are queues which are serviced by the REQUEST and ANSWER routines. The mechanism is illustrated below.



MASTER-SLAVE COMMUNICATION

5.9.1.5 SCHEDULING

Scheduling is the means by which the operating system decides which process is to run and when. In a real-time system such as XMOS, each process will only run for a small period - generally, much less than a second - before it gets 'swapped out' by the scheduler. There are two questions to consider - when to swap a process out and which process to choose to replace it in the CPU.

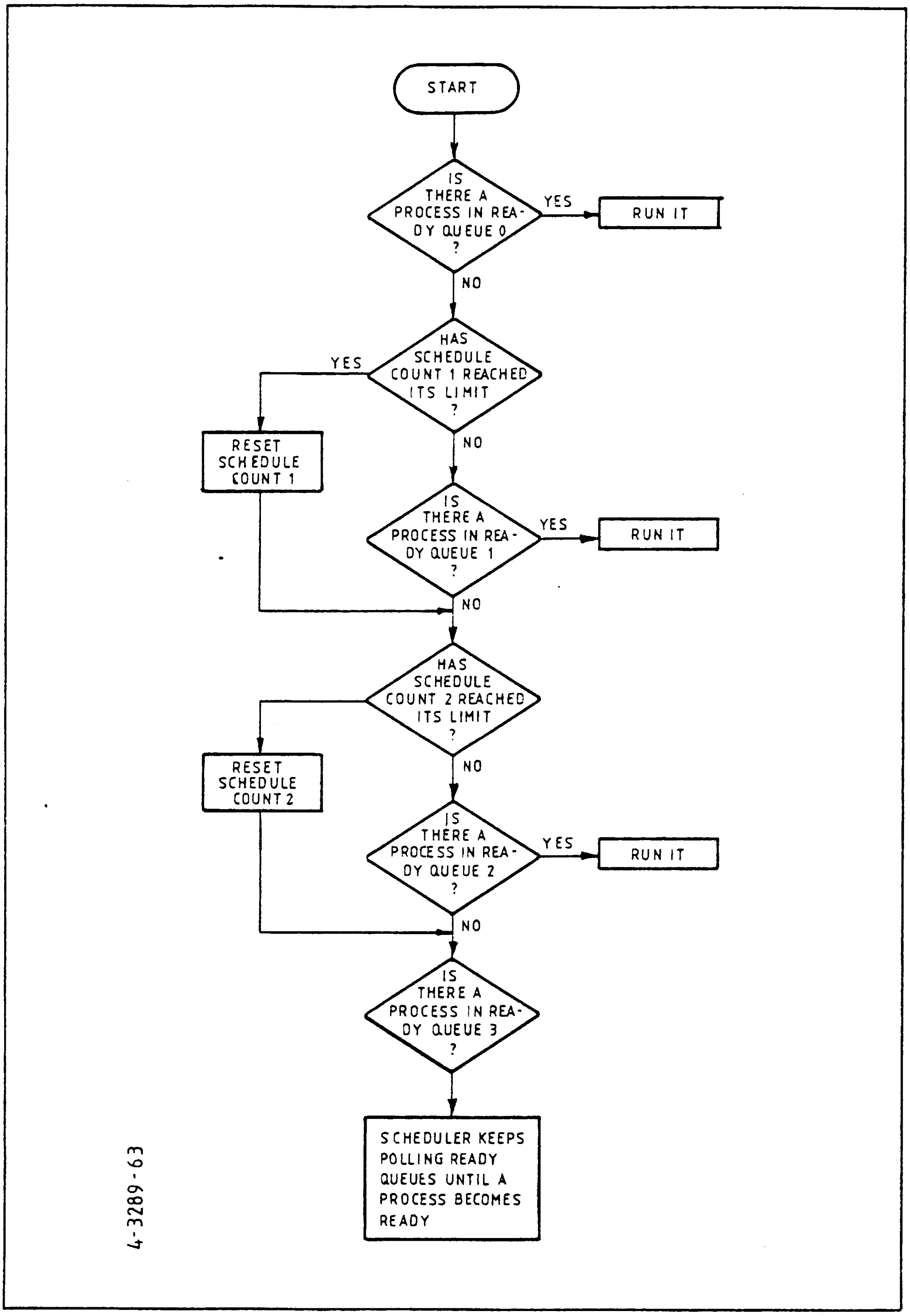
There are two ways of deciding when to swap a process out - that is, to change its state from 'running' to 'ready' (as described in sect. 5.9.1.3). Note that in XMOS, this decision can be taken by the process itself, rather than by the Scheduler. One of the ways is to carry on processing until a 'natural break' is reached, in other words, until the process is forced to wait for some external event such as the completion of a disk write. This method should generally only be used when the processing time between waits is small. The other method which is used if a process is CPU-bound (that is, it seldom has to wait for external events) is to rely on the time-slice function provided by XMOS 'Tick' routine. In this case, the Scheduler will swap out any process that has been running for more than a pre-defined number of 'ticks' provided by a hardware clock. Note that time-slicing is optional, and if chosen applies to all processes in the system.

It is possible to temporarily stop the execution of one process by means of the interrupt mechanism. In this case a time-critical routine can interrupt the running process and run itself. Then the use of the CPU is relinquished in favour of the process that was running before the interrupt occured. Note that

a) The interrupting routine is not treated as a separate process but rather a temporary part of the current process.

b) After the interrupt has run there is no rescheduling, i.e. the process is not changed for another.

The scheme that XMOS uses to decide which process to run next is as follows:-


START
IS THERE A PROCESS IN READY QUEUE 0 ?
YES
RUN IT
NO
HAS SCHEDULE COUNT 1 REACHED ITS LIMIT ?
YES
RESET SCHEDULE COUNT 1
NO
IS THERE A PROCESS IN READY QUEUE 1 ?
YES
RUN IT
NO
HAS SCHEDULE COUNT 2 REACHED ITS LIMIT ?
RESET SCHEDULE COUNT 2
NO
IS THERE A PROCESS IN READY QUEUE 2 ?
YES
RUN IT
NO
IS THERE A PROCESS IN READY QUEUE 3 ?
SCHEDULER KEEPS POLLING READY QUEUES UNTIL A PROCESS BECOMES READY
4-3289-63

SCHEDULING

Note the use of the Schedule Count to prevent low priority processes from being totally ignored in a system which has many high priority processes. A typical Schedule Count might be 8, so that one in 8 times, a job of priority 2 is chosen when there is one at priority 1 which is ready. Similarly, once in 64 times a priority 3 job would oust a priority 1 job. A guide to choosing a suitable priority for different types of processes is given below:-

Priority 0:	(Interrupt Priority): Process needing to deliver rapid responses, such as servicing interrupt routines, DMA, etc.
Priority 1:	(User Priority 1): Processes servicing requests from other processes, and jobs which are important for the system speed.
Priority 2:	(User Priority II): Processes that have uncritical response time.
Priority 3:	(On-line Diagnostic): Processes that will be running if the system has nothing else to do.

Processes at priority 0 should have short execution times since they can interrupt the running process. Processes with extremely long execution times can be placed at priority 1 and 2 depending on the job; the time slice function will then interrupt the running process if a specified time has elapsed.

5.9.1.6 XMOS MEMORY & TIME REQUIREMENTS

The following estimates are based on the Z80A processor as used in the CR8. The timings include execution time both in the Kernel and Assembler Interface modules.

Program Size	Kernel	900 bytes
	BQM	200 bytes
	Time Queue Monitor	800 bytes
	Assemb. I/F	900 bytes
	BQM Utility	500 bytes
	Calendar	250 bytes
	In all, less than	4 Kbytes
Data Size	Kernel	70 bytes
	1 parent process	29 bytes + stack
	1 child process	25 bytes + stack
	TQM (13 timers)	128 bytes
	History (16 elements)	100 bytes
Timing	Send a message	101/180 microsecs.
	Wait for message	285/350 microsecs.
	Accept a message	15/83 microsecs.
	Send a signal	101/180 microsecs.
	Wait for signal	285/350 microsecs.
	Release	368 microsecs.
	Time Slice	300 microsecs.
	Stack check	50 microsecs.
	History	85 microsecs.
	Activate timer	170+Nx35 microsecs. (N = timer number)
	Suspend timer	100+Nx35 microsecs.
	Tick routine	100 microsecs.

5.9.1.7 STORAGE ALLOCATION

The operating system is designed to work in a ROM/RAM based environment but in the CR8 it is entirely in RAM. The XMOS modules are all relocatable so they can be located where the user has available ROM and RAM space.

The figure overleaf shows the XMOS data structure. The system program module is the basic firmware for the hardware. The "Start" module, which is called when the hardware is checked and initialized, will configure the whole system by calling the "CONFIG" function and then hand over the control to the operating system by a jump to the system scheduler via the XMOS entry list.

The RAM part of the whole system consists of the XMOS system memory, semaphores and one data area for each process. The user is responsible for the RAM layout. If a process uses special data areas, the address can be included in the user specified "External table" which will also contain channel dependent addresses and hardware addresses. The use of an External Table for each process (which contains all the data unique to that process) means that the code for many similar processes may be shared.

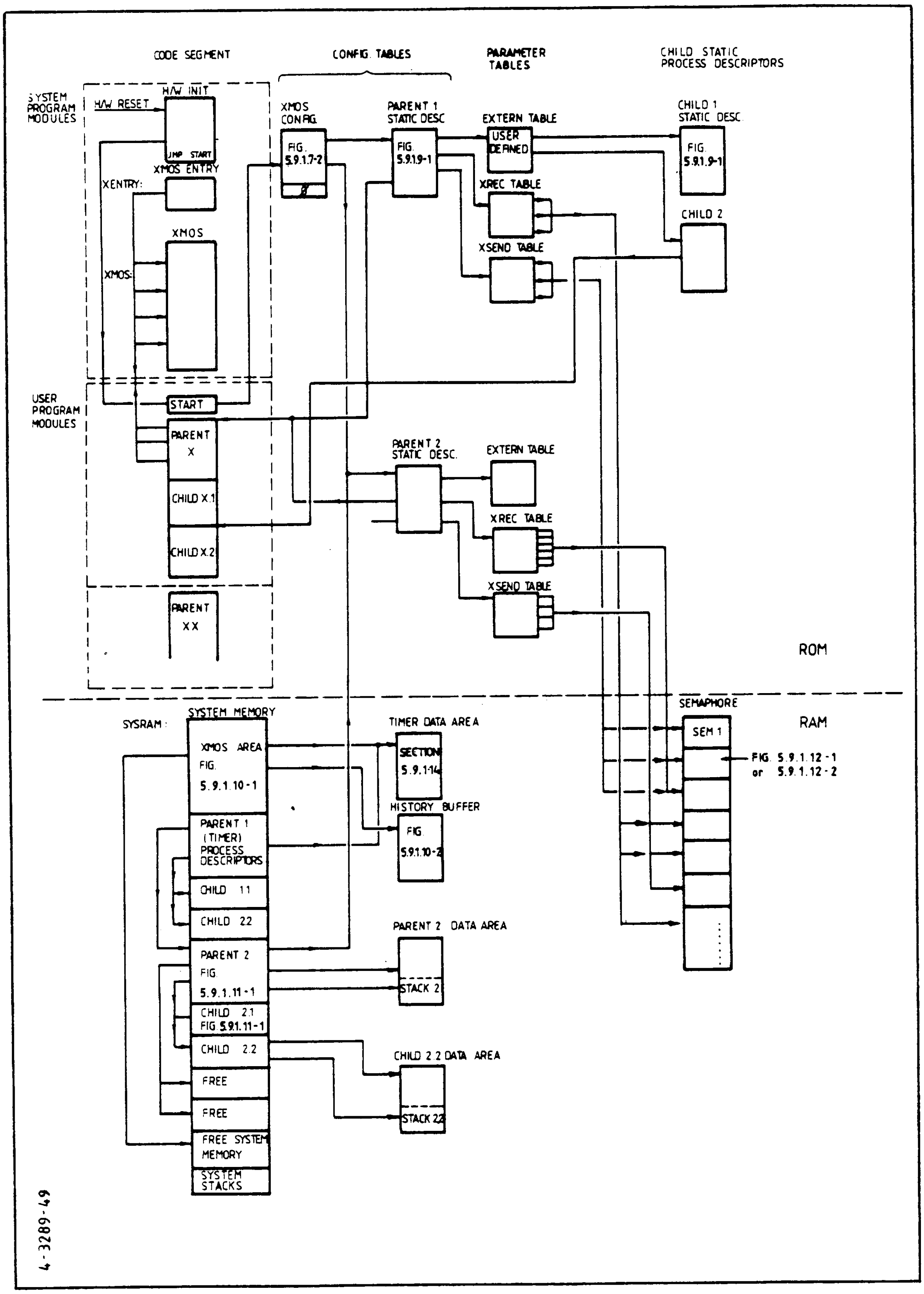
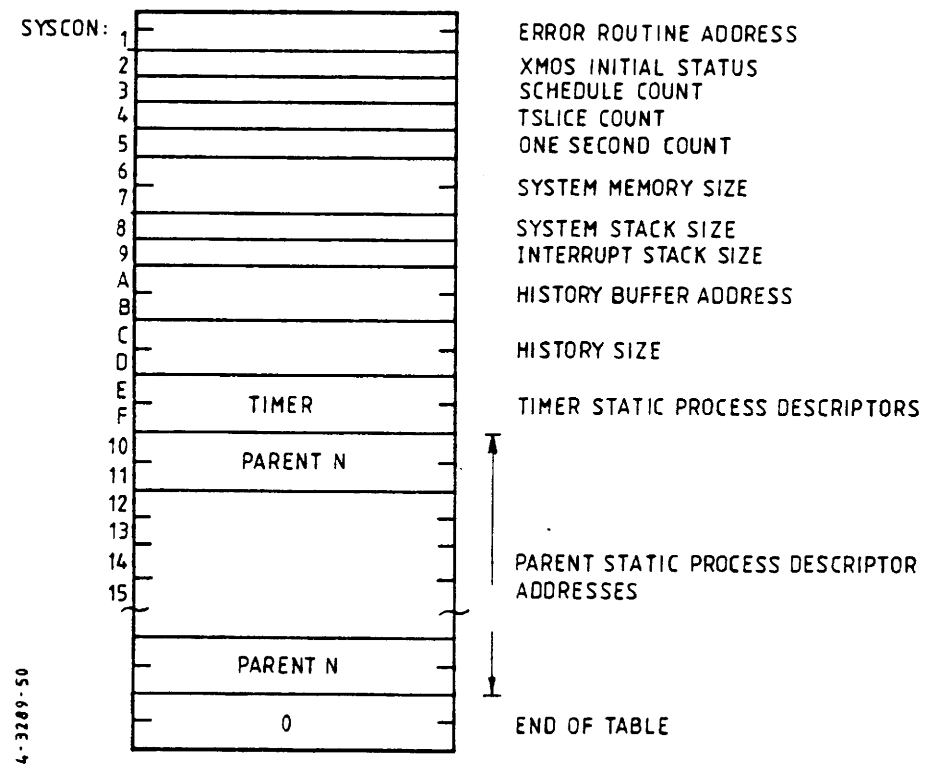


Fig. 5.9.1.7-1 AN OVERVIEW OF XMOS MEMORY LAYOUT

5.9.1.8 CONFIGURATION TABLE

The XMOS configuration table contains all the information required to generate a system including a list of the static parent process descriptors. The address of this table is the input parameter to the "CONFIG" function.



XMOS CONFIGURATION TABLE

Error routine address:

The operating system will make a call to this user written routine if an fatal error is detected.

XMOS initial status:

This byte is copied to the system memory and selects the functions the user wishes to be enabled.

bit 0:	stack check
bit 1:	history
bit 2:	priority 0 interrupt
bit 3:	time slice

If the bit is '1' the function is enabled.

Schedule count:

The count is used to weight the ready queues, avoiding blocking of lower priority processes.

Tslice count:

This count is the maximum no. of ticks a process can run for without being swapped out. Then the running process is interrupted and routed to the ready queue as if the "RELEASE" function was called. All the main registers are saved on the process' stack.

One second count:

This is the no. of ticks to make exactly one second. The count is used by the timer to update the calender. The tick length itself is also programmed into hardware timers such as the PIT chip on the MP^2 card.

System memory size (bytes):

The system memory start address (SYSRAM:) is inserted by the linker. The system memory size will depend on the number and types of processes since it includes the process descriptors.

System stack size (bytes):

The stack area size reserved for the Kernel. The stack is at the top of the system RAM area.

Interrupt stack size (bytes):

The stack area reserved for interrupt routines. This stack area can be used by interrupt routines instead of using the interrupted process' stack area. The stack area is placed below the system stack area.

History buffer address:

The address of the history data area.

History buffer size (words):

The memory size reserved for the history buffer. The number of elements in the history buffer is: (buffer size - 4) / 6.

Parent N:

The "CONFIG" function will create all the parent processes specified in the configuration table, and make a list of free child process descriptors connected to each parent.

The timer process is the first parent process that is created and it is assigned priority 0; it will be the very first running process.

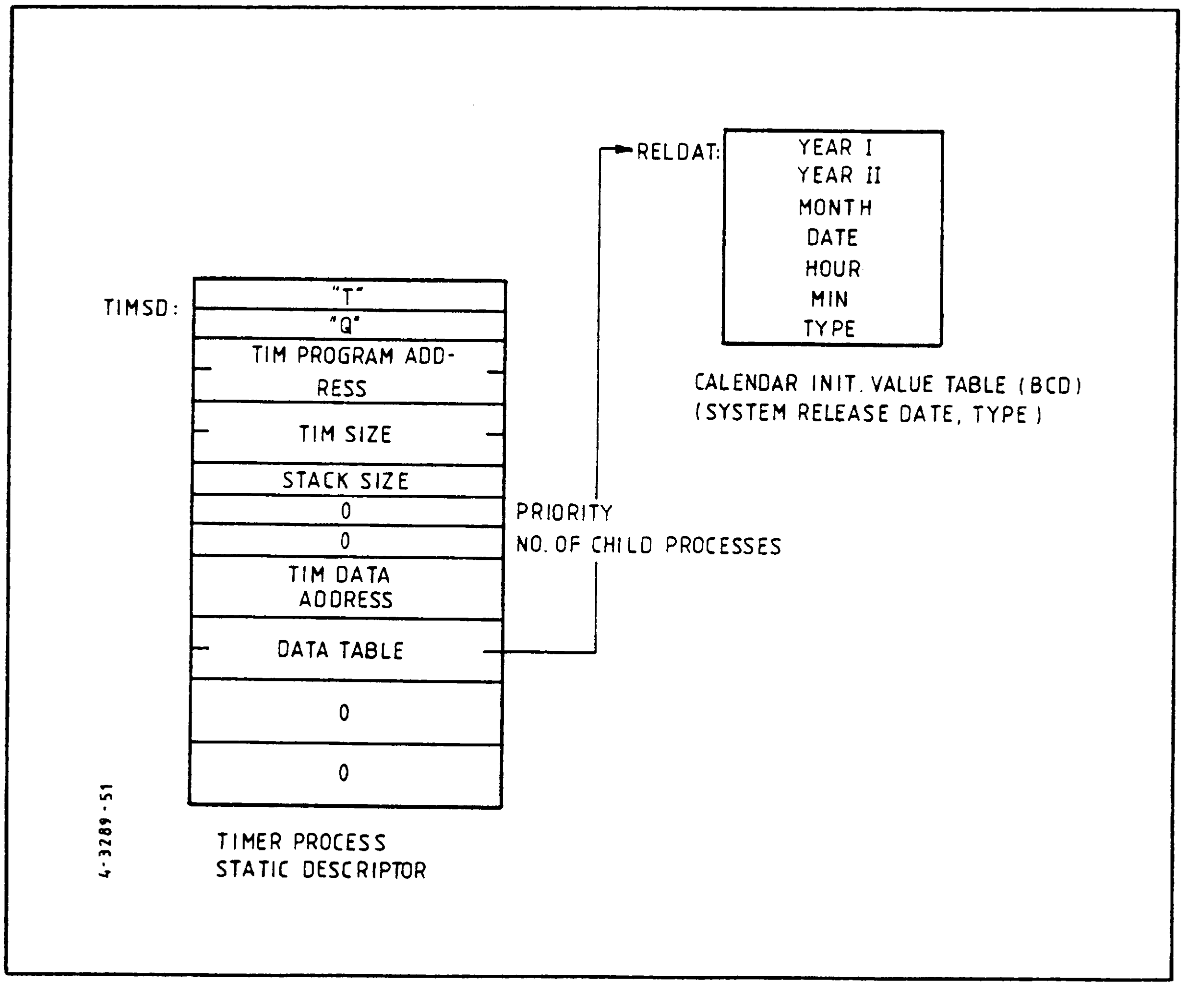


Fig. 5.9.1.8-2 STATIC TIMER TABLES

5.9.1.9 STATIC PROCESS DESCRIPTORS

The static process descriptor is used by the kernel when a process is created.

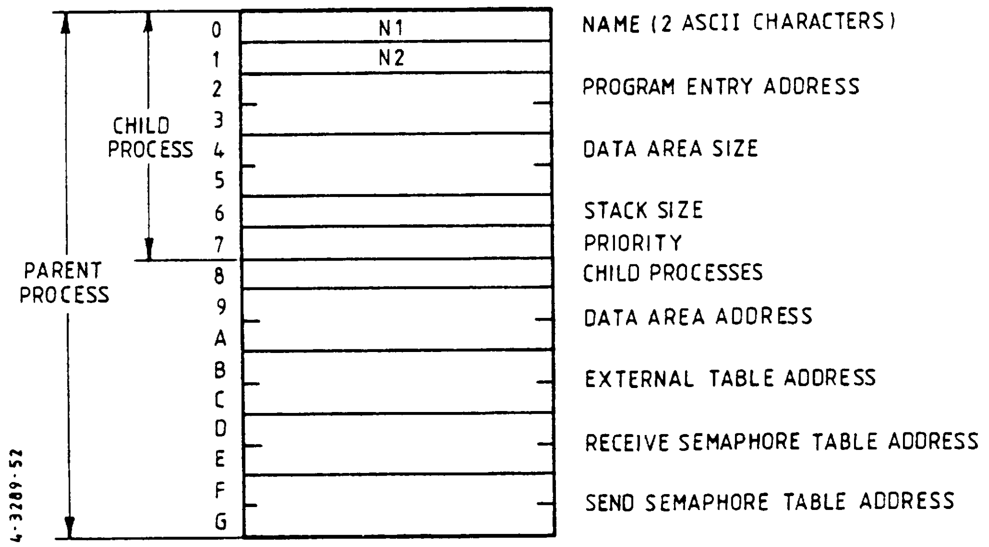


Fig. 5.9.1.9-1 STATIC PROCESS DESCRIPTOR

The parent static process is addressed from the configuration table. The address of the child static process descriptor is used as entry parameter to the "CREATE" function.

Name:

2 ASCII characters defining the process name. The name is only used by debugging tools.

Program entry address:

The kernel will jump to this address when the process is set running the very first time.

Data area size (words):

The RAM memory size needed as work area for the process. The user must remember that the stack area is part of the data area.

Stack size (bytes):

The stack size reserved for the process. Stack area used by interrupts must be included.

Priority (bytes):

bit 0, 1 define the process priority.

Parents only:

Child processes: This byte defines the number of free process descriptors that will be reserved for the parent process to use for its children.

Data area address: The address of the work area reserved for the parent process. The initial stack pointer will be: Data area address + Data area size.

External table address: The external table includes the parameters needed for the process program such as hardware addresses, child static process descriptor addresses, special data areas, etc.

Receive semaphore table address: The receive semaphore table contains the addresses of the semaphores in which the parent process and their children will send messages and events to.

Send semaphore table address: The send semaphore table contains the addresses of the semaphores to which the parent process and its children will send messages and events.

The receive and send semaphore tables are used to establish static connections between processes.

5.9.1.10 THE "SYSRAM:" AREA

The term 'System Memory' used in this section refers to a subset of the total memory used by XMOS. See figure 5.9.1.7-1 (page 5-173).

The data memory used by the operating system consists of three blocks: System memory, history buffer and the timer data area.

The system memory include pointers and the stack area used by the operating system. The dynamic process descriptors are also included in the system memory. The start address of the system memory (SYSRAM:) is fixed whereas the end of the system memory, the history buffer and the timer data area are all located during system configuration. Pointers to the data areas and the configuration tables are loaded into the first part of the system memory, followed by the dynamic process descriptors. These are all created during configuration. The stack area used by the operating system is located at the top of the system memory (high addresses).

The timer data layout is shown on pages 5-188 and 5-189.

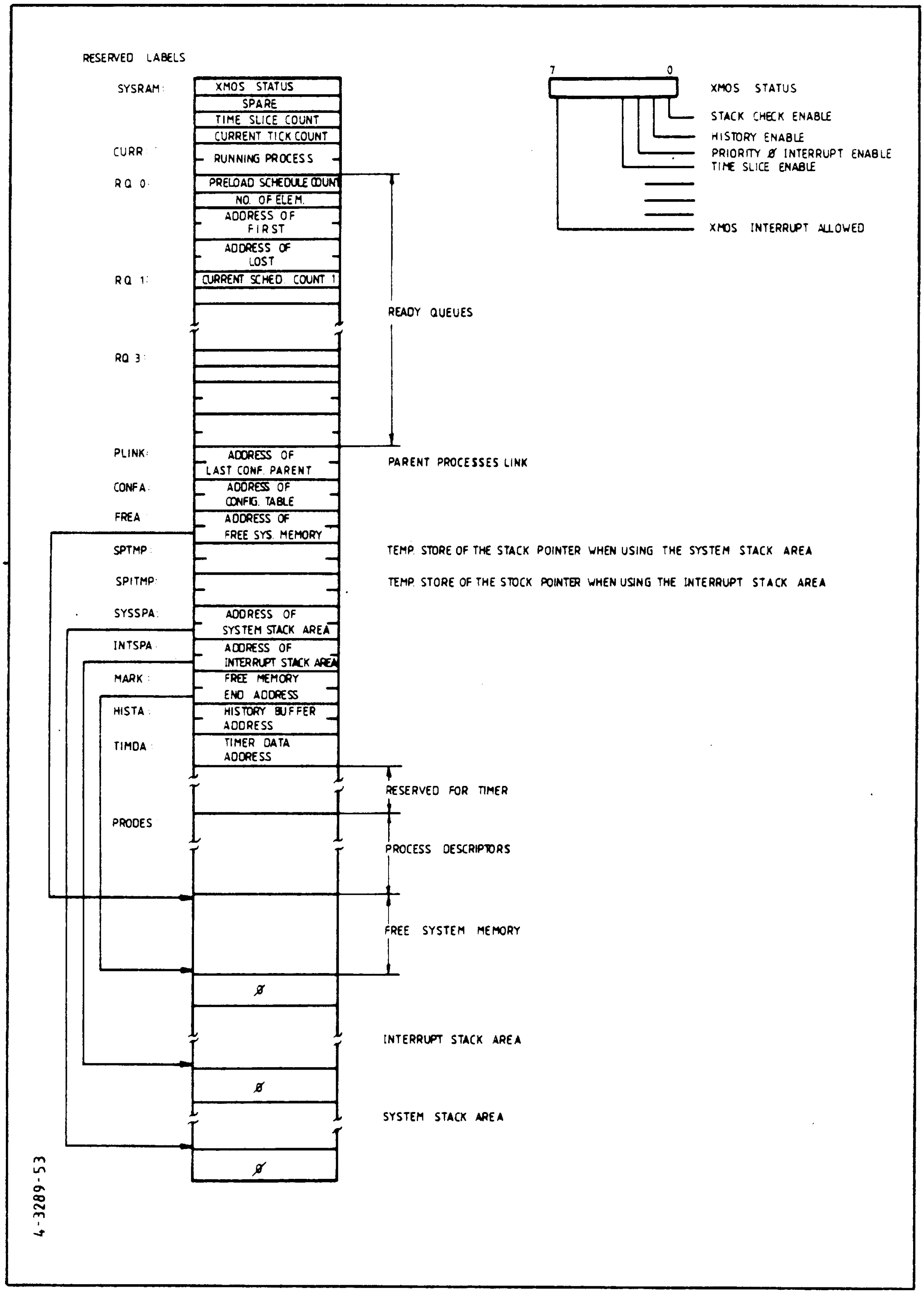


Fig. 5.9.1.10-1 XMOS SYSTEM MEMORY

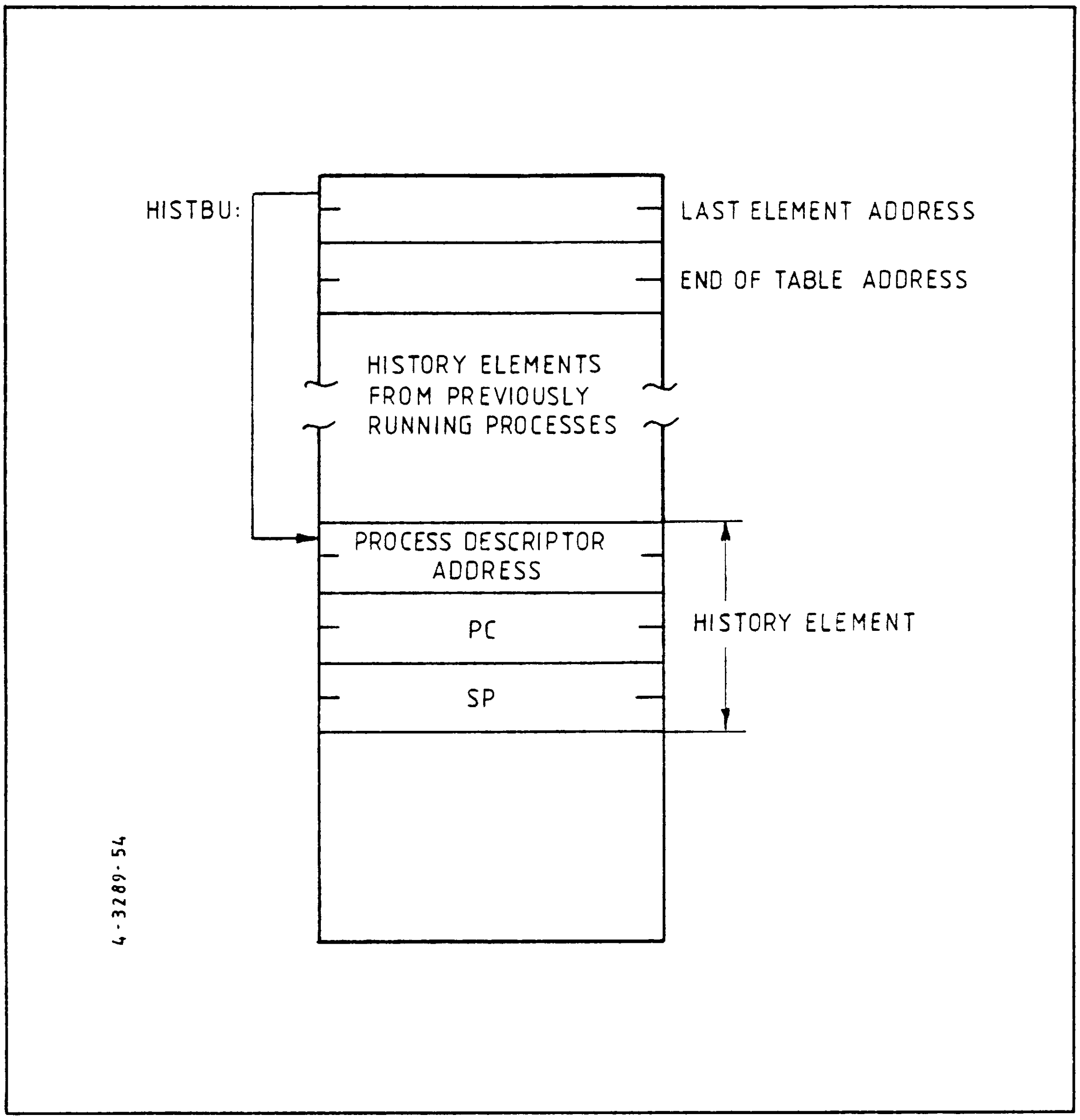

HISTBU:
LAST ELEMENT ADDRESS
END OF TABLE ADDRESS
HISTORY ELEMENTS FROM PREVIOUSLY RUNNING PROCESSES
PROCESS DESCRIPTOR ADDRESS
PC
SP
HISTORY ELEMENT
4-3289-54

Fig. 5.9.1.10-2 HISTORY BUFFER

5.9.1.11 DYNAMIC PROCESS DESCRIPTORS

A process is identified by the address of its dynamic process descriptor. The first 25 bytes of the two types of process descriptor are identical, except that a parent process descriptor is destinguished by bit 7 of the process status being set.

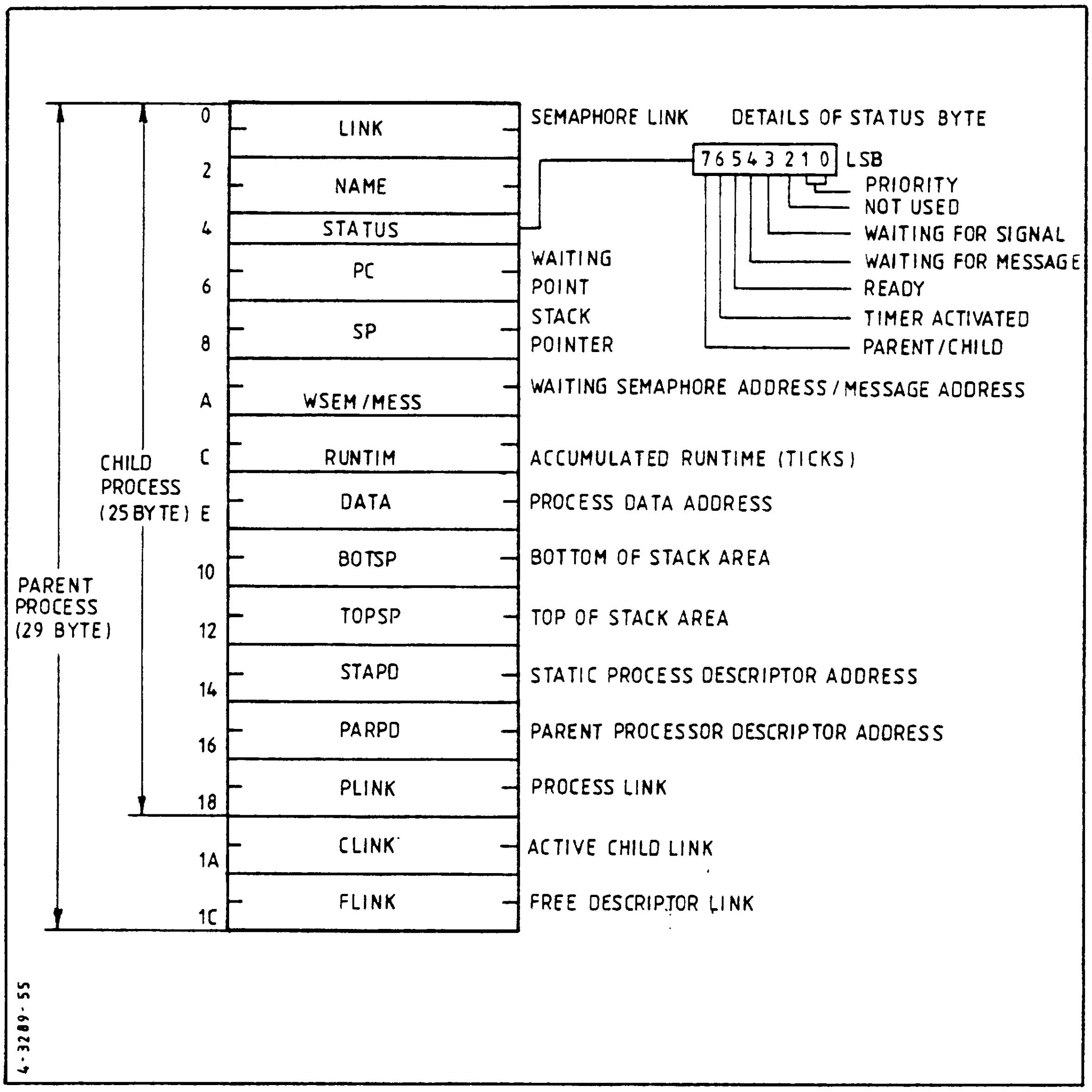


Fig. 5.9.1.11-1 DYNAMIC PROCESS DESCRIPTOR

5.9.1.12 SEMAPHORES AND QUEUES

XMOS deals with three types of queues which are handled by three types of descriptors. These are known as general semaphores, message semaphores and queue monitors. The two types of semaphore handle synchronisation of processes; queue monitors are used to queue buffers of data of various types, they are not, however, concerned with scheduling as semaphores are.

General semaphores are used to implement the SIGNAL and WAIT functions. Several processes could send Signals to a general semaphore. This means that the addresses of their process descriptors will be inserted into the queue of such descriptors. This queue of descriptors is described by the general semaphore (plus pointers or 'links' within the process descriptors themselves, to the next descriptor in the queue). The meaning of the semaphore at any given moment is either "these processes require attention when you get a chance" - or, "this many serving processes had a look, but no processes required attention".

Message semaphores are used to implement the SEND and RECEIVE functions. It can be in any one of three states at any time - either one or more processes are waiting for a message, or no process or messages are waiting for a message, or neither processes nor messages are waiting, or there is a queue of messages which have arrived and await processing. So depending on whether there are more processes seeking a message or the other way round, then the message semaphore implements a queue either of process descriptors or pointers to message buffers.

The third type of queue descriptor is the **queue monitor.** These queues are never of processes, just of data that is shuffled around within or between processes. The format of the data buffers used is described in the next section. This format is the same as that used for messages. The user must be careful not to use the BQENQ and BQDEQ (Queue Monitor) routines for queueing messages on a message semaphore, though it is OK to use those routines to queue a message from a message semaphore onto a Queue Monitor (non-semaphore) type of queue e.g. into an empty buffer pool. Another example of the use of the Queue Monitor facilities is to pass buffers of data between SCRIMP/M and MP/M in the common RAM area.

The formats of the three types of queue descriptors are given below.

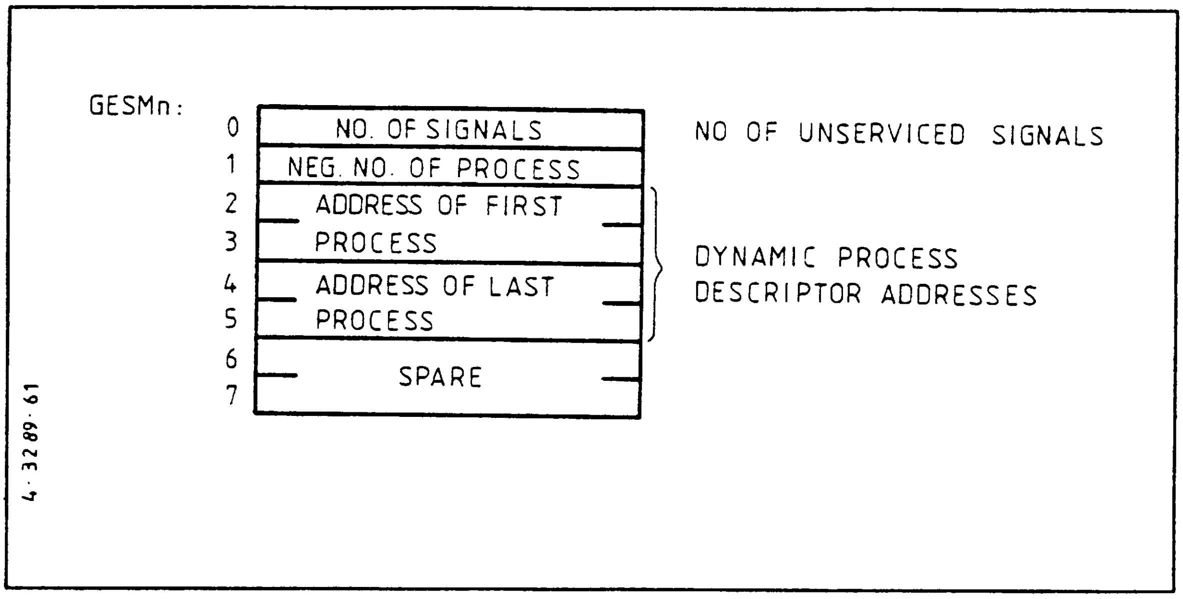


Fig. 5.9.1.12-1 GENERAL SEMAPHORE - (SIGNAL/WAIT FUNCTION)

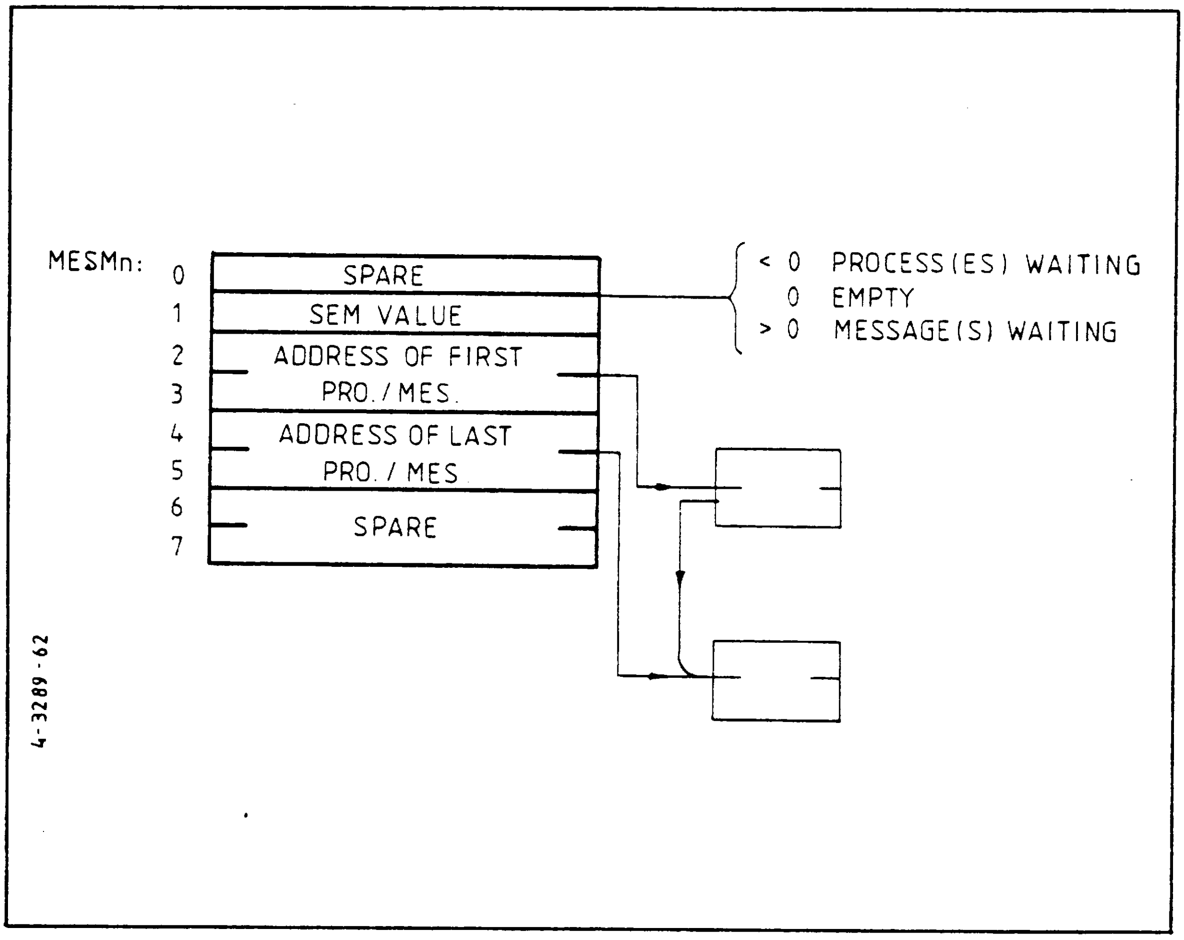


Fig. 5.9.1.12-2 MESSAGE SEMAPHORE - (SEND/RECEIVE FUNCTION)

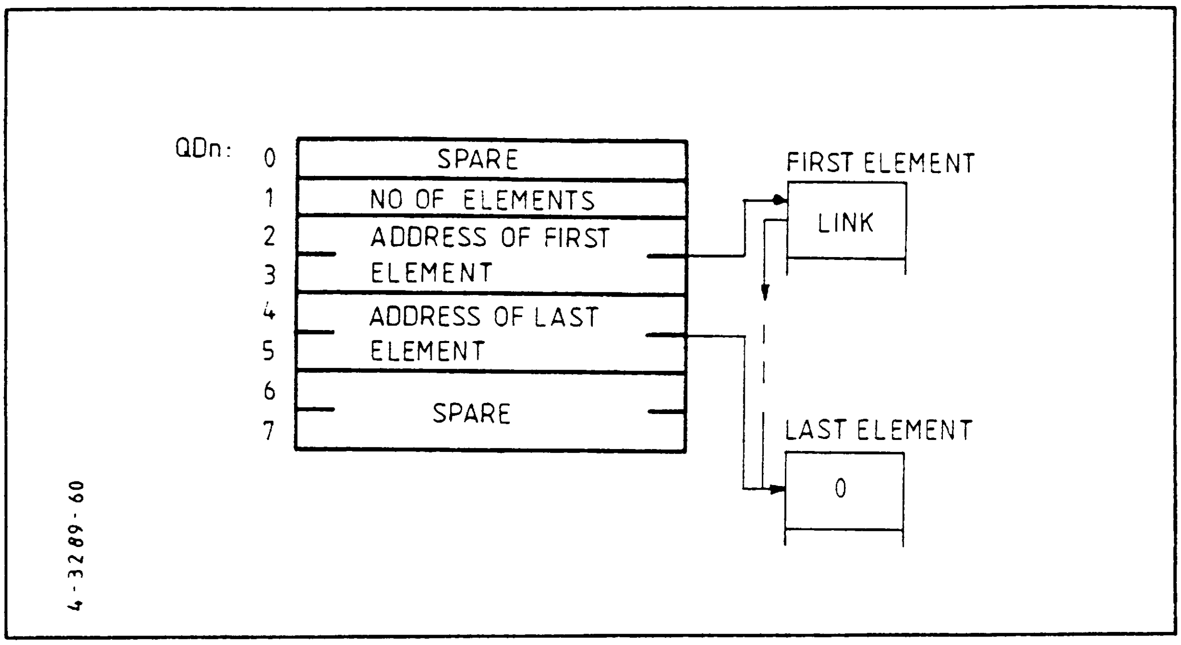


Fig. 5.9.1.12-3 QUEUE MONITOR - (FOR QUEUEING DATA BUFFERS)

5.9.1.13 MESSAGES

Interprocess messages are sent in data buffer format. This data buffer format is also used when delivering data to/from external computers via shared memory.

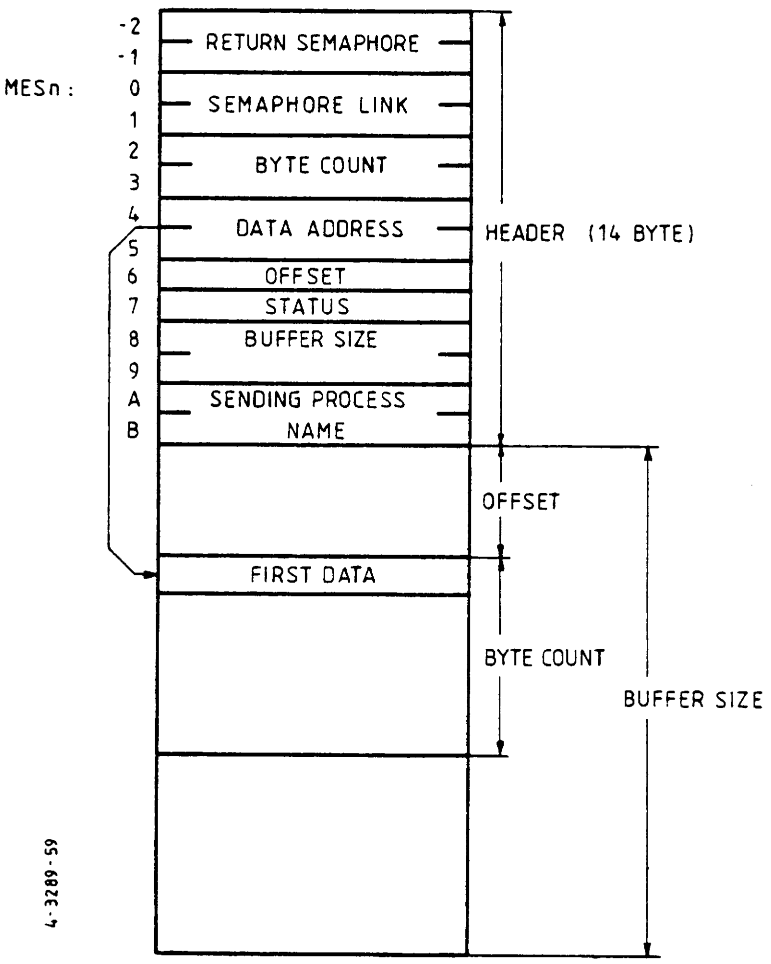


Fig. 5.9.1.13-1 DATA BUFFER FORMAT

5.9.1.14 TIME QUEUE MONITOR (TQM)

The TQM software consists of four modules that serve the following purposes:

1. Implementing the timeouts associated with XMOS's WAIT and RECEIVE routines (see section 5.9.16.1).

2. The 'Tick' routine. This is called as an interrupt to the current process. It decrements delay counts and activates functions that have timed out.

3. A calendar providing date and time to the nearest second.

4. User interface to the 'tick' routine.

The timer process is created during system configuration, and when the process starts to run it will initialize the timer data area and then wait indefinitely at the timer semaphore located in the timer data area. The timer process is sent a signal when the calendar needs servicing (once a second) and when user written subroutines must be called. The static process descriptor layout was shown in fig. 5.9.1.9-1 (page 5-177).

The TQM is based on two linked lists of data structures called **time descriptors.** The format of a time descriptor is shown below, and the way they are used in the following figure. One of the linked lists consists of unused descriptors and the other is of active descriptors that are currently timing events. Descriptors are transferred from one list to the other in accordance with current needs. The whole system is very efficient because only one value in memory has to be decremented with each tick of the hardware clock in order to time all the events on the system that require it. This is implemented by timing each event relative to its nearest neighbour (chronologically nearest, that is). In the example illustrated below, one request is for 3 ticks, one for 8 and one for 18 (and there are four free time descriptors). The queue is set up in the order that they must be serviced, with the time differences between then recorded, not the absolute waiting times. Thus only the '3' needs to be decremented each tick until that request is serviced. Then the value '5' is initialized into the tick counter, and that number only is decremented until the second timer request is serviced, and so on.

If a new request occurs for an absolute time that falls within the existing queue, only one time descriptor in the active descriptor list need be changed (the one for the request immediately following the new one) and of course one new descriptor generated (taken from the list of free descriptors). If there are no free descriptors left, then the requested action is executed immediately.

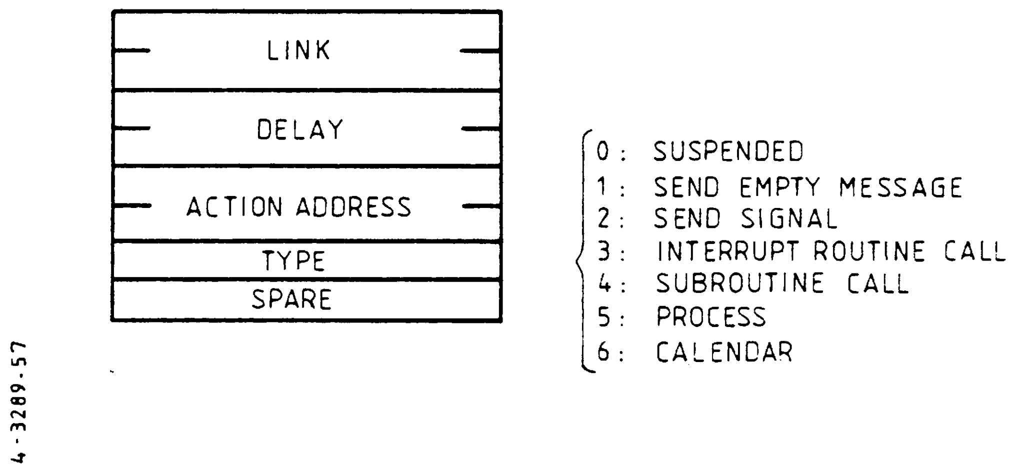


Fig. 5.9.1.14-1 TIME DESCRIPTOR FORMAT

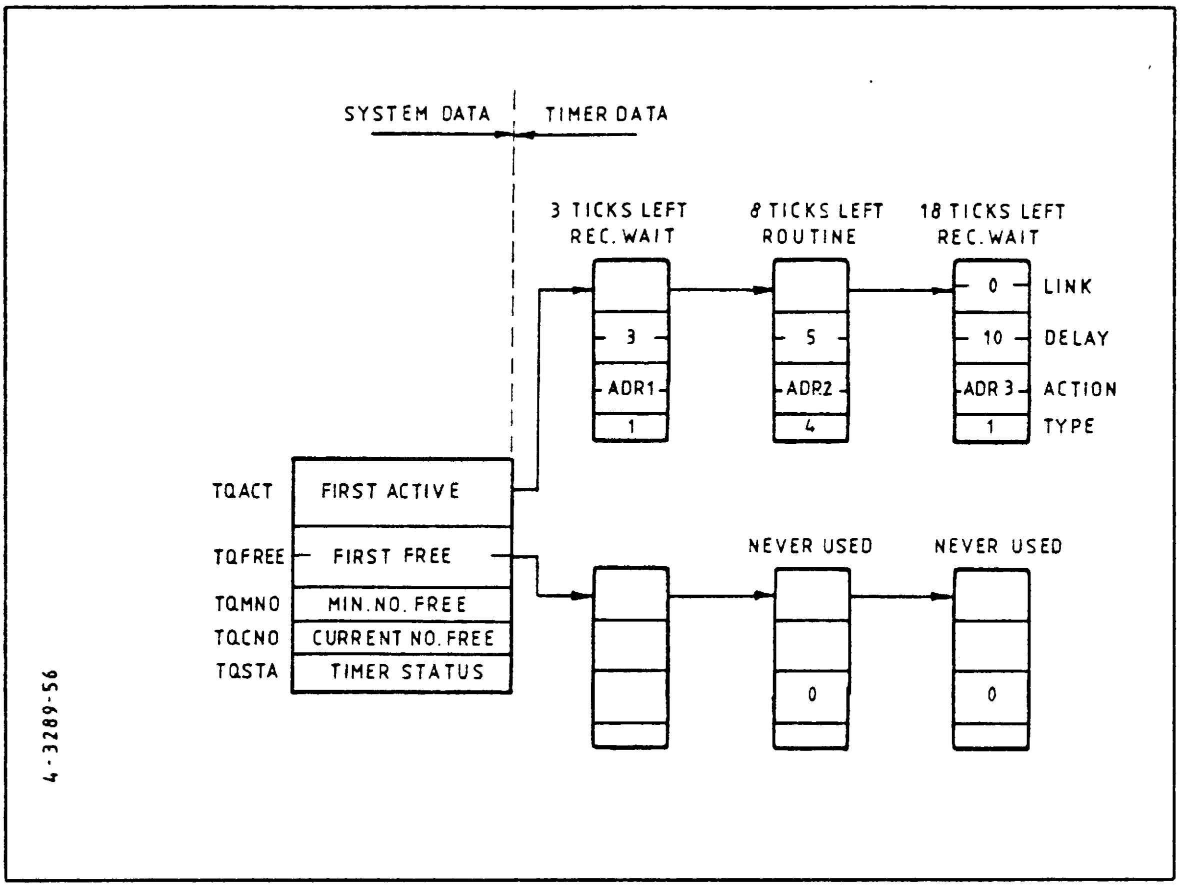


Fig. 5.9.1.14-2 HOW TIME DESCRIPTORS ARE USED

5.9.1.15 HOW TO USE THE XMOS SOFTWARE PACKAGE

The XMOS operating system is written in assembler in order to make it as efficient as possible in memory space and execution time. The source includes some conditional assembly directives so that it is easy to make a system that only contains the facilities that are required.

The functions that can be selected are:-

- o Timer for XMOS (SYSTQM)
- o Timer for user (USETQM)
- o Buffer handling routines (BQM)
- o Time slicer (TSLICE)

The output from the compilation is three relocatable blocks: the XMOS entry table, the operating system code and the operating system data memory. The XMOS entry table is the user interface to the operating system and consists of jumps to routines in the system code. This allows for the updating of the operating system without changing the entry addresses.

All interaction with XMOS is via calls to the routines outlined below. Remember that CONFIG must be called, once, at the beginning, in order to define which processes are to exist in the system.

Interrupts. CPU interrupts are always enabled when returning from an XMOS call except from the routines INTSIG and INTSEN. These two routines are provided so that one can service an interrupt without being interrupted oneself while doing it.

Error handling. If XMOS detects an error, a call is automatically made to the user-written error routine whose address appears in the Configuration Table.

Waiting Points. Some of the routines in the first table below include waiting points. This means that during the course of their execution, they save the current register values and parameters onto a stack then call the Scheduler. This normally means that this process gets 'scheduled out' for a while, during which time other processes will run. As soon as this process is set to 'Run' again by the Scheduler, control returns to the 'Waiting Point' routine where it left it and thence to the user's program in the normal way.

Calling Procedure. The method of passing parameters to and from these routines will of course depend on the programming language being used. However, in all cases the registers that are not used for passing parameters will be saved onto the process' own stack and restored before returning from the XMOS routine. The details that follow apply to Z80 assembler.

Timings. The time taken to execute each routine is given in 'T-cycles'. A T-cycle is 250 nanoseconds in the Z80A processor used in the CR8.

An overview of all the user callable XMOS routines is given in the tables below, followed by full details each one.

ENTRY POINT	FUNCTION	WAITING POINT
CONFIG	Informs XMOS about all parent processes	No
CREATE	Creates a Child process	No
DELETE	Deletes a Child process	Yes
CURRENT	Get information about this process	No
RELEASE	Go to sleep (free the CPU for other processes)	Yes
WAIT	Go to sleep until woken by another process' semaphore or timer	Yes
SIGNAL	Send a signal to wake up a process	No
INTSIG	Signal while disabling interrupts	No
RECEIVE	Wait for a message from another process	Yes
ACCEPT	Inspect a message semaphore to see if any messages have arrived	No
SEND	Send a message	No
INTSEN	Send a message while disabling interrupts	No
REQUEST	Send a message that requires a reply	No
ANSWER	Send a message as a reply to a REQUEST	No

"USER INTERFACE" ROUTINES (ALWAYS AVAILABLE)

ENTRY POINT	FUNCTION
TQEMES	Sends an empty message to a message semaphore after a fixed delay
TQISIG	Same effect, but using a signal instead of a message.
TQSUB	Execute user specified subroutine after specified delay
TQISUB	Causes timer interrupt routine to execute user's routine after a fixed time
TQSUSP	Cancels a pending timer job
——	Calendar service for time and date

TIME QUEUE MONITOR ROUTINES

ENTRY POINT	FUNCTION
BQDEQ	Gets an element from the front of a queue
BQENQ	Puts an element onto the back of a queue
BQSEND	Sends a buffer with address to which it must be later returned
BQRET	Receiving process sends buffer off to address specified by BQSEND
BQMOV	Moves all the elements in one queue to another
BQGET	Searches a queue for a particular element
BQCRE	Creates a data buffer and puts it into a queue
BQSET	Resets pointers in buffer header
BQFIFI	Finds first data in buffer
BQFILA	Finds last byte in buffer
BQLDTX	Copies a string of bytes into a data buffer at the beginning
BQADTX	Copies a string of bytes into a data buffer after previous data
BQPOTX	Adjusts first data pointer to somewhere outside the buffer
BQADFI	Adds one byte before first data (and adjusts pointers)
BQADLA	Adds one byte after last data (and adjusts pointers)
BQAJFI	Adjusts the offset to the 'first data in buffer' pointer
BQLDBC	Adjusts byte count (e.g. to throw away last byte of data)
BQLDRQ	Loads a return queue address in the buffer header
BQRDRQ	Reads a return queue address from the buffer header
BQRDNA	Reads the 2-byte ASCII name from the buffer header
BQLDNA	Puts such a name into a buffer header
SETSTA	Set Status byte in buffer header
SRESTA	Read Status byte in buffer header

BUFFER QUEUE MONITOR ROUTINES

5.9.1.15.1 THE USER INTERFACE ROUTINES

CONFIG

Entry parameters:	IX register contains configuration table address (see sect. 5.9.1.8).
Exit parameters:	None.
Description:	This routine initialises the operating system and creates the specified parent processes, including their free process descriptors for child processes. The parent processes are inserted into the assigned Ready queues. After returning from this routine, one must jump to the Scheduler (EXSHED). The routine checks that the memory used does not exceed the specified system memory size.
Registers Destroyed:	A,F, BC, DE, HL, IX, IY
Timing (T-cycles):	2300 + (1961 x No. of processes) + (140 x No. of child procs.)

CREATE

Entry parameters:	IX register - Static Process Descriptor address HL reg. - Process data area address
Exit parameter:	IX - Address of Dynamic Process Descriptor
Description:	Creates a Child Process using a free process descriptor from the Parent. It puts the child process in the appropriate Ready Queue.
Register Destroyed:	HL
Timing (T-cycles):	1470

DELETE

Entry parameters:	None
Exit parameters:	None
Description:	Deletes the process that runs it, which must be a Child Process (the routine checks that it is). The user must release any memory used by the process before calling the routine. Waiting point.

CURRENT

Entry parameters:	None
Exit parameters:	HL reg. - Address of Data area for process.
	IX reg. - Address of Dynamic Process Descriptor.
	BC reg. - External Receive Semaphore Table Address
	DE reg. - External Send Semaphore Table Address.
	IY reg. - External Parameter Table Address
Description:	This routine obtains the information as described above. The information refers to the process that calls the routine.
Register destroyed:	F
Timing (T-cycles):	245

RELEASE

Entry parameters:	None
Exit parameters:	None

Description:	Moves the current running process to the assigned Ready queue - i.e. puts the calling process to sleep thus releasing the CPU for other processes to use. Waiting point.
Registers destroyed:	None
Timing (T-cycles):	1470

WAIT

Entry parameters:	HL reg. -	Address of General Semaphore
	DE reg. -	Delay in 8 msec. ticks.

Exit parameters: None

Description:

The current running process is connected to the specified general semaphore. If that semaphore has already been Signalled, then the process calling this routine will stop running and be put onto one of the four ready queues (that is, it will be detached from the semaphore). Also the 'Unserviced Signals' count in the semaphore will be decremented by one.

On the other hand, there may have been no outstanding Signals, in which case the process will be put in the Wait state until a Signal arrives on that semaphore. If a delay is specified, the process will give up waiting for signals after that time out period, and go Ready anyway.

The routine also checks that the number of processes waiting in the semaphore does not exceed 127. Waiting point.

Registers destroyed:	DE, HL
Timing (T-cycles):	No signal on semaphore: 1230

	Signal(s) on semaphore: 1140
	Delay was specified: 1400

SIGNAL

Entry parameters:	HL register - Address of a general semaphore.
Exit parameter:	A register - Semaphore value (see fig. 5.9.1.12-1).
Description:	The specified general semaphore is inspected. If one or more processes are waiting (2nd byte in semaphore does not equal zero) then the first process is removed from the semaphore and its descriptor's address is placed on the relevant Ready queue. In other words, a call to SIGNAL will wake up whichever process has been waiting the longest at that semaphore. On the other hand, there may be no processes waiting. In this case, the first byte of the general semaphore, the 'No. of Signals' byte, is incremented. No record is kept of the identity of the calling process, but it does serve as an indicator of how much interest there has been recently in the semaphore. In either case, control returns immediately to the caller.
Registers destroyed:	F, HL
Timing (T-cycles)	Empty semaphore: 173
	Process waiting: 660

INTSIG

This routine is identical to SIGNAL except that interrupts remain disabled after returning (both routines disable interrupts while they are running themselves). INTSIG is for use by an interrupt routine so that a signal can successfully be sent without being interrupted itself.

RECEIVE

Entry parameters:	HL register - Address of Message Semaphore DE reg. - Delay in 8msec. ticks.
Exit parameters:	If Carry bit is set, then: HL reg. - Address of first data in message BC reg. - Message size (bytes) If Carry bit is clear: DE reg. - zero, i.e. 'No message'.
Description:	This routine causes the running process to wait for a message from another process by using a Message Semaphore. If a message is already attached to the semaphore then the calling process goes immediately to the 'Ready' instead of the 'Wait' state. If a delay value is specified, then the 'Wait' state will be timed out after that period and the process that called RECEIVE will be put on a Ready queue. The routine also checks that the number of processes waiting on the semaphore does not exceed 127. Waiting point.
Registers destroyed:	F, plus BC and HL if message received.
Timing (T-cycles):	Semaphore was empty: 1671 Process(es) waiting: 1882 Message(s) waiting: 1662

ACCEPT

Entry parameter:	HL register - Address of Message Semaphore
Exit parameters: A reg.	- Semaphore value
	(if negative, number of processes waiting
	if positive, number of messages waiting)
	If Carry bit is set:
	DE reg. - Message address
	HL reg. - Address of first data in message
	BC reg. - Message size (bytes)
	If Carry bit is clear:
	DE reg. - Zero (No messages)
Description:	This inspects the specified message semaphore but never waits. It returns the message if there is one. The purpose of this routine is to allow a process to receive messages into several semaphores; it could poll them to see which had received a message first.
Register destroyed:	F
Timing (T-cycles):	No message - 62
	Message(s) - 331

SEND

Entry parameters:	DE reg. - Address of Message
	HL reg. - Address of Message Semaphore
Exit parameters:	A reg. - Semaphore value (if negative, no. of process waiting. If poitive, no. of messages waiting).

Description:	The routine sends a message to the specified Message Semaphore. Then, if a process had been waiting on that semaphore it will be moved to a Ready queue (and if it had been waiting for a timer, that timer is stopped). The name of the current process is inserted in the message header. The routine checks that no more than 127 messages have been queued on the semaphore.
Registers destroyed:	DE, HL
Timing (T-cycles):	Semaphore was empty - 405 Message(s) were waiting - 476 Process(es) were waiting - 720

INTSEN

This routine is identical to SEND except that interrupts are disabled. (The explanation for "INTSIG" applies here too).

REQUEST

Entry parameters:	DE reg. -	Address of Message
	HL reg. -	Address of destination Message Semaphore
	BC reg. -	Address of answering Message Semaphore
Exit parameter:	A reg. -	Destination semaphore value (for details see 'SEND').

Description:	This routine sends a message that requires a reply. It inserts the address of the semaphore that must answer into the header of the message that it sends. The routine is used to command a Slave process to do something. Then the slave sends his reply on the specified answer semaphore. The master process would call RECEIVE after the call to REQUEST in order to obtain that reply; the slave process would call ANSWER in order to send that reply.
Registers destroyed:	F, DE, HL
Timing (T-cycles):	Destination semaphore was empty: 551 Destination semaphore had message(s) waiting: 622 Destination semaphore had process(es) waiting: 866

ANSWER

Entry parameter:	DE reg. -	Address of message
Exit parameter:	A reg. -	Value of answer semaphore (if negative, it is the number of messages. If positive, it is the number of waiting processes on the answer Message semaphore).

Description:	The routine sends the answer message to the semaphore specified in the REQUEST message header. See REQUEST.
Registers destroyed:	F, DE
Timing (T-cycles):	Destination semaphore was empty: 553 Destination semaphore had message(s) waiting: 624 Destination semaphore had process(es) waiting: 868.

5.9.1.15.2 TIME QUEUE MONITOR ROUTINES

The following routines are available if the appropriate conditional assembly flags were set when the operating system was assembled. See section 5.9.1.14 for a more general discussion of the Time Queue Monitor. It should be borne in mind that all these TQM routines disable interrupts while they are running.

TQEMES

Entry parameters:	HL reg. - Address of message semaphore DE reg. - Delay required (8 msec. ticks)
Exit parameters:	None
Description:	This routine is used to activate the first process that is hanging on the specified semaphore. A null message is sent to the semaphore, which causes the process on the semaphore to be put into a Ready queue. This procedure is carried out after the specified delay. N.B. Interrupts are disabled.
Registers destroyed:	DE, HL.

TQISIG

This routine differs from TQEMES only in that a general rather than a message semaphore is used (i.e. a signal rather than a message is sent).

TQSUB

Entry parameters:	HL register - Entry Address of subroutine to execute DE reg. - Delay required (8 msec ticks)
Exit parameters:	None

Description:	This routine will call a user-specified routine after a specified delay, using the mechanism described in section 5.9.1.14. The user routine is made 'Ready' at that time and will run soon thereafter. (For a faster response, use TQISUB.) The called routine runs 'in a vacuum', i.e. not as part of a process.
Registers destroyed:	DE, HL.

TQISUB

This routine differs from TQSUB only in that the specified user routine is executed within the timer software's interrupt routine. So the user routine should be as short as possible and should not enable interrupt (or it may be interrupted itself). The user routine is allowed to corrupt registers as these are saved before running the routine and restored afterwards.

TQSUSP

Entry parameters:	HL register - Action address as recorded in the time descriptor to be removed.
Exit parameters:	None
Description:	This routine searches the linked list of active time descriptors for the specified descriptor and removes it. This is used to cancel a pending timer request.
Register destroyed:	HL

Calendar

The TQM updates its calendar every second; however, the user must supply his own routines to initialise or read this part of XMOS' data area in memory. The following table shows where the different components are located. Each is one byte; the offsets are in hex.

Item	Offset from symbolic address TIMDA:
Millenium & century	8
Decade & year	9
Month	A
Date	B
Hour	C
Minute	D
Second	E

CALENDAR ADDRESSES

Each byte is coded in BCD format, that is, the byte is divided into 2 nibbles, each of which represents a decimal digit in binary. Thus at any time between now and the year 2000, if we last that long, the byte at address TIMDA:+8 ought to contain 00011001.

5.9.1.15.3 BUFFER QUEUE MONITOR ROUTINES

The following routines are available if the appropriate conditional assembly flags were set when the operating system was assembled. The reader should refer back to the diagram of the Data Buffer format (section 5.9.1.13, page 5-186) upon which all these routines are based. The way in which these buffers are queued is described just before that ('Queue Monitor', section 5.9.1.12).

BQDEQ

Entry parameter:	HL register - Address of Queue Monitor
Exit parameter:	A reg. - Number of elements in queue If Carry bit is set: DE reg. - Address of required data buffer (i.e. queue element) If Carry bit is clear: DE reg. - Zero. Queue is empty.
Description:	This routine removes the first element from the front of the specified queue.
Register destroyed:	F (always), and HL if queue element found
Timing (T-cycles):	30 to 133

BQENQ

Entry parameters:	DE reg. - Adress of queue element HL reg. - Address of queue monitor
Exit parameters:	A reg. - Number of elements in queue Carry bit set - More than one element in queue Carry bit clear - 1 element now in queue

Description:	A data buffer is added to the rear of the specified queue.
Register destroyed:	F, DE, HL
Timing (T-cycles):	135 to 206

BQSEND

Entry parameters:	DE reg. - Address of element for queue HL reg. - Address of Queue Monitor BC reg. - Return Queue address to be inserted
Exit parameters:	A reg. - Number of elements in queue
Description:	This routine, like BQENQ, adds a data buffer to the rear of a specified queue. However, the 'Return Semaphore' part of the buffer header is also filled in. This gives the address of another queue's monitor to which a reply must be sent.
Register destroyed:	F
Timing (T-cycles):	191 to 262

BQRET

Entry parameter:	DE reg. - Address of element
Exit parameter:	A reg. - No of elements in queue
Description:	This sends a data buffer back as a reply to a BQSEND. The address of the Queue Monitor to send it to is obtained from the 'Return Semaphore' field of the buffer header.
Register destroyed:	F
Timing (T-cycles):	183 to 254

BQMOV

Entry parameters:	HL reg.- Address of queue Monitor for destination queue.
	DE reg. - Address of Queue Monitor for source queue.
Exit parameters:	A reg. - No of elements in destination queue
Description:	This moves all the elements from the source queue and adds them to the rear of the destination queue.
Registers destroyed:	F, DE, HL.

BQGET

Entry parameters:	DE reg. - Element Address
	HL reg. - Address of queue monitor
Exit parameters:	A reg. - No of elements in queue
	Carry bit set: DE reg. - Address of the element
	Carry bit clear: DE reg. - Zero (Element not found)
Description:	A specified element on the queue is searched for, and if found it is removed from the queue (and its address given to the caller).
Register destroyed:	F, HL.

BQCRE

Entry parameters:	DE reg. - Starting address of available memory (i.e. the address of the space for the Return Semaphore)
	HL reg. - Address of queue monitor
	BC reg. - Buffer size (excluding header)
	A reg. - Offset

Exit parameters:	DE reg. -	First unused memory address
	A reg. -	Number of buffers in queue

Description: This routine creates a data buffer as per the format described in section 5.9.1.13. Then it queues this empty buffer on the specified queue.

Registers destroyed: F, BC, HL
Timing (T-cycles): 574 to 675

BQSET

Entry parameters:	DE reg. -	Buffer address
	A reg. -	New Offset value
Exit parameters:	HL reg. -	Address of First Data
	BC reg. -	Maximum possible value of Byte Count

Description: This is used to reset the header of a data buffer. For example, if one had inserted some data into the area which is skipped by the Offset, then the Offset could be reduced by the appropriate amount thus incorporating the new data into the area of 'known data'. The routine would adjust Data Address, Offset, and Byte Count appropriately. (Or one could use BQADFI).

Register destroyed: F
Timing (T-cycles): 266

BQFIFI

Entry parameter:	DE reg. -	Buffer head address
Exit paramenters:	HL reg. -	Pointer to First Data
	BC reg. -	Byte Count

Description:	This gets the pointer to the first data and the byte count from the buffer header.
Register destroyed:	None
Timing (T-cycles):	102

BQFILA

Entry parameter:	DE reg. -	Buffer-header address
Exit parameters:	HL reg. -	Last data pointer
	BC reg. -	Byte count
Description:	This routine generates a pointer to the current last byte of data.	
Registers destroyed:	None	
Timing (T-cycles):	119	

BQLDTX

Entry parameters:	DE reg. -	Buffer Head address
	HL reg. -	String Address
	BC reg. -	String length
Exit parameters:	HL reg. -	Address in data buffer immediately after the last new character
	BC reg. -	Number of remaining characters (normally zero).
	Carry bit set - OK. BC reg. is zero	
	Carry bit clear - Data buffer too small: string truncated.	
Description:	Copies user's string into the data buffer.	
Register destroyed:	F	
Timing (T-cycles):	451 to 680 + (21x No of bytes copied)	

BQADTX

Entry and exit parameters:	Exactly as for BQLDTX routine (above).
Description:	Exactly as for BQLDTX except that the text is added at the end of the previous data.
Register destroyed:	F
Timing (T-cycles):	511 to 740 + (21 x No. of bytes copied).

BQPOTX

Entry parameters:	DE reg. - Buffer header address
	HL reg. - Address of string to be incorporated
	BC reg. - Length of that string
Exit parameters:	None
Description:	This routine is used to incorporate an external string into the Data Buffer format. In other words, the pointer and counters in the buffer header are set up to refer to a data buffer which (including offset area) is <u>not</u> contiguous with the buffer header. The original data area is ignored until the situation is normalised using a call to BQSET.
Registers destroyed:	None
Timing (T-cycles):	97

BQADFI

Entry parameters:	DE reg. - Buffer header address
	A reg. - The datum (a byte)
Exit parameters:	HL reg. - Data pointer

	BC reg. -	Byte count
	Carry bit set	- Datum loaded OK
		- A reg. gives new value of Offset
	Carry bit clear	- There was no room for datum
		- A reg. set to FF Hex

Description: This is used to insert a single byte into the area skipped by the Offset. The byte is inserted immediately before the previous first byte. This could be used for example for inserting a special line-starting character into a text string for transmission over a communications link. The routine adjusts the data pointer, byte count and offset appropriately.

Register destroyed: F
Timing (T-cycles): 186 to 263

BQADLA

Entry parameters:	DE reg. -	Buffer header address
	A reg. -	Datum
Exit parameters:	HL reg. -	Data pointer
	Carry bit set	- Datum inserted OK
		BC reg. - byte count
	Carry bit clear	- No room for datum. Not loaded.
		BC reg. - zero

Description: One byte of data is appended into the Data buffer (if there is room for it). The information in the header is updated.

Register destroyed: F
Timing (T-cycles): 274 to 328

BQAJFI

Entry parameters:	DE reg. -	Buffer header address
	BC reg. -	Number of bytes to insert
Exit parameters:	HL reg. -	First data address
	BC reg. -	Byte count
	A reg. -	Offset

Description:

This adjusts the pointer to the beginning of data without moving any data. It is used to create space at the front of the buffer (positive BC register when calling) or for throwing away byte(s) at the beginning of data (negative BC register). If space is inserted (positive BC) that space is not initialized to nulls or spaces. The routine adjusts Offset, Pointer and Byte count.

Register destroyed: F
Timing (T-cycles): 243

BQLDBC

Entry parameters:	DE reg. -	Buffer header address
	BC reg. -	Byte count
Exit parameter:	HL reg. -	First data address

Description:

This is analogous to the previous routine BQAJFI, except that space is added to or subtracted from the end of the data buffer. Note also that the BC value is the overall byte count, not the increment to it as in BQAJFI.

Register destroyed: None
Timing (T-cycles): 102

BQLDRQ

Entry parameters:	DE reg. - Buffer header address
	HL reg. - Return queue address
Exit parameters:	None
Description:	This routine is used for writing to the 'Return Semaphore' field at the very top of the data buffer format.
Registers destroyed:	None
Timing (T-cycles):	56

BQRDRQ

This routine is identical to the previous one BQLDRQ except that the Return Queue address is returned to the caller, rather than set by the caller.

BQRDNA

Entry parameters:	DE reg. - Buffer address
	HL reg. - Process name - 2 bytes in ASCII
Description:	This reads the 'Sending Process Name' field in the buffer header.
Registers destroyed:	None
Timing (T-cycles):	82

BQLDNA

This is identical to BQRDNA (above) except that the name is written and is a calling parameter, rather than being read and returned to the caller.

SETSTA

Entry parameters:	DE reg. -	Buffer head address
	A reg. -	Status byte to be written in
Exit parameter:	HL reg. -	Pointer to status byte

Description: This is used to insert a number into the Status Byte slot in a buffer header.

Registers destroyed: F

Timing (T-cycles): 38

SRESTA

This is identical to SETSTA (above) except that the Status byte is read and returned, instead of being supplied by the user and written.

5.9.2 DEVICE DRIVERS UNDER SCRIMP/M

An example of a device driver for the CR8 is given below - it is the printer driver. Remember that device driving is done by SCRIMP/M in dual processor applications, or by CP/M or MP/M (i.e. the only operating system) in Z80-only systems.

```
        .Z80
        PAGE 50
;***********************************************************************
;
;
;   XXXXXXXXXXXXXXXXXX          X   X  X   X   XXX    XXXX
;   XXXXXXXXXXXXXXXXXX           X X   XX XX  X   X  X
;   XXXX    XX    XXXX            X    X X X  X   X   XXX
;   XX XX   XX   XX XX           X X   X   X  X   X      X
;   XX  XX  XX  XX  XX          X   X  X   X   XXX   XXXX
;   XX   XX XX XX   XX
;   XX    XXXXXX    XX
;   XXXXXXXXXXXXXXXXXX          CHRISTIAN ROVSING A/S
;   XXXXXXXXXXXXXXXXXX
;   XX    XXXXXX    XX          PROGRAMMER:     PNJ
;   XX   XX XX XX   XX
;   XX  XX  XX  XX  XX          DATE:           821216
;   XX XX   XX   XX XX
;   XXXX    XX    XXXX
;   XXXXXXXXXXXXXXXXXX
;   XXXXXXXXXXXXXXXXXX
;
;
;______________________________________________________________________
;
;               ********************************************************
;               *
;               *  MODULE NAME: CHROUT/CHRIN PROCESS
;               *
;               ********************************************************
;
;-----------------------------------------------------------------------
;
;DESCRIPTION: !
;
;-----------------------------------------------------------------------
        PAGE
;***********************************************************************
;
;CHANGE RECORD:
;-----------------------------------------------------------------------
;    DATE      PROGRAMMER       DESCRIPTION
;-----------------------------------------------------------------------
;       !               !
;       !               !
;***********************************************************************
        PAGE
        INCLUDE SYSEQU
        INCLUDE DEVEQU
;       PUBLIC  CHRIO1,TX1INT,EXT1INT
        PUBLIC  CHRIO2,TX2INT,EXT2INT
;       PUBLIC  CHRIO3,TX3INT,EXT3INT
;       PUBLIC  CHRIO4,TX4INT,EXT4INT
        EXTRN   DEQTXF,ENQTXE,DEQRXE,ENQRXF
        EXTRN   BQSET,BQADLA,RELEASE,CURRENT,OFIFO

                               1
```

EXAMPLE DEVICE DRIVER

```
        EXTRN   BQFIFI,CRTMS,BAUDTAB,IFIFO,BQLDBC

;CHRIO1:
CHRIO2:
;CHRIO3:
;CHRIO4:
        CALL    CURRENT
        LD      L,(IX+PDATA+0)
        LD      H,(IX+PDATA+1)          ;GET HL TO POINT ON DATA AREA
        PUSH    HL
        POP     IY
CHROUT:
        LD      HL,FLAG1
        BIT     4,(HL)
IF00:
        JP      NZ,FI00                 ;IF XONOFF = OFF
        BIT     0,(HL)
IF01:                                   ;   IF TX IN PROGRESS = OFF
        JP      NZ,FI01
        LD      HL,FLAG2
        BIT     0,(HL)
IF02:                                   ;      IF BUFFER IN USE = OFF
        JP      NZ,ELSE02
        CALL    NEWBUF                  ;         CALL  NEWBUF
        JP      FI02
ELSE02:                                 ;      ELSE
        CALL    RETBUF                  ;         CALL  RETBUF
FI02:                                   ;      FI
FI01:                                   ;   FI
FI00:                                   ;FI
CHRIN:
        LD      L,(IX+PDATA+0)          ;HL = FIFO START ADDRESS
        LD      H,(IX+PDATA+1)
        LD      BC,0020H                ;BC = FIFO LENGTH
        CALL    OFIFO                   ;GET A CHR FROM FIFO
IF03:
        JP      NC,FI03                 ;IF A CHR AVARIABLE
        PUSH    BC                      ;   SAVE CHR
        LD      A,C
        CP      13H
IF04:                                   ;   IF CHR = XOFF (CTRL S)
        JP      NZ,ELSE04
        LD      HL,FLAG1
        SET     4,(HL)                  ;      SET XONOFF = ON
        JP      FI04
ELSE04:
        CP      11H
IF05:                                   ;      IF CHR = XON (CTRL Q)
        JP      NZ,FI05
        DI
        LD      HL,FLAG1
        RES     4,(HL)                  ;         SET XONOFF = OFF
        BIT     0,(HL)
IF06:                                   ;         IF TX IN PROGRESS = ON
        JP      Z,FI06
```

2

EXAMPLE DEVICE DRIVER (CONTINUED)

```
        LD      BC,(COUNTER)
        LD      HL,(POINTER)
        LD      A,(HL)                  ;           GET NEXT CHR IN BUFFER
        DEC     BC
        INC     HL
        LD      (COUNTER),BC
        LD      (POINTER),HL            ;           RESTORE POINTERS
        OUT     (SIO+SDATA),A           ;           SEND CHR TO LINE
FI06:                                   ;        FI
        EI
FI05:                                   ;     FI
FI04:                                   ;  FI
        LD      A,CHANNEL                  ;   GET CHANNEL NUMBER
        CALL    DEQRXE
IF07:
        JP      NC,ELSE07               ;  IF BUFFER IN IFRXE
        LD      BC,0000H                ;     RESET BYTE COUNT
        CALL    BQLDBC
        LD      A,03H                   ;     SET BUFFER TYPE = 03H
        CALL    BQADLA                  ;     INSERT
        LD      A,0FFH                  ;     SET OPCODE TYPE = 0FFH
        CALL    BQADLA                  ;     INSERT
        POP     BC                      ;     GET SAVED CHR
        LD      A,C                     ;     LOAD A-REG. WITH DATA BYTE
        CALL    BQADLA                  ;     INSERT DATA IN BUFFER
        LD      A,CHANNEL               ;     GET CHANNEL NUMBER
        LD      HL,0000H                ;     SET RETURN QUE = 0000H
        CALL    ENQRXF                  ;     SEND BUFFER TO RX FULL QUE
        JP      FI07
ELSE07:                                 ;  ELSE
        POP     BC                      ;     GET SAVED CHR
        LD      E,(IX+PDATA+0)          ;     RESET FIFO
        LD      D,(IX+PDATA+1)          ;     DE = LOWER ADDRESS
        LD      HL,0020H
        ADD     HL,DE                   ;     HL = LOWER ADDRESS + 20H
        EX      DE,HL                   ;     EXCHANGE DE-HL
LOOP:
        LD      (HL),00H                ;     INSERT 00H UNTIL
        INC     HL                      ;     FIFO IS RESET
        LD      A,D
        CP      H
        JR      NZ,LOOP
        LD      A,E
        CP      L
        JP      NZ,LOOP
FI07:                                   ;  FI
FI03:                                   ;FI
        CALL    IN                      ;TRY TO GET A CHR. FROM CONSOL
        CALL    RELEASE                 ;SCHEDULE
        JP      CHROUT
;
;***************************************************************************
;
NEWBUF:
        LD      A,CHANNEL               ;GET CHANNEL NUMBER
```

3

EXAMPLE DEVICE DRIVER (CONTINUED)

```
        CALL    DEQTXF
IF08:
        JP      NC,FI08               ;IF BUFFER IN IFTXF
        LD      (BUFFER),DE           ;   SAVE BUFFER ADDRESS
        CALL    BQFIFI                ;   FIND FIRST DATA BYTE
        LD      D,(HL)                ;   GET BUFFER TYPE
        LD      A,03H
        CP      D                     ;   COMPARE BUFFER TYPE WITH 03H
IF09:
        JP      NZ,ELSE09             ;   IF BUFFER TYPE = 03H
        CALL    OUT                   ;      CALL OUT SUBROUTINE
        JP      FI09
ELSE09:                               ;   ELSE
        LD      A,07H
        CP      D                     ;      COMPARE BUFFER TYPE WITH 07H
IF10:
        JP      NZ,FI10               ;      IF BUFFER TYPE = 07H
        CALL    INIT                  ;         CALL INIT SUBROUTINE
FI10:                                 ;      FI
        LD      DE,(BUFFER)           ;      GET BUFFER ADDRESS
        LD      A,00H                 ;      OFFSET = 00H
        CALL    BQSET                 ;      CALL BUFFER RESET
        LD      A,CHANNEL             ;      CHANNEL NUMBER
        LD      HL,0000H              ;      SET RETURN QUE = 0000H
        CALL    ENQTXE                ;      SEND BUFFER TO TX EMPTY QUE
FI09:                                 ;   FI
FI08:                                 ;FI
        RET
;
;***************************************************************************
;
RETBUF:
        LD      DE,(BUFFER)           ;GET BUFFER ADDRESS
        LD      A,00H                 ;OFFSET = 00H
        CALL    BQSET                 ;CALL BUFFER RESET
        LD      A,CHANNEL             ;CHANNEL NUMBER
        LD      HL,0000H              ;SET RETURN QUE = 0000H
        CALL    ENQTXE                ;SEND BUFFER TO TX EMPTY QUE
        LD      HL,FLAG2
        RES     0,(HL)                ;SET BUFFER IN USE = OFF
        RET
;
;***************************************************************************
;
INIT:
        INC     HL
        LD      B,(HL)                ;GET CONTROL TYPE
        LD      A,04H                 ;COMPARE CONTROL TYPE = 04H
        CP      B
IF11:
        JP      NZ,ELSE11             ;IF CONTROL = 04H
        INC     HL
        LD      B,(HL)                ;   GET FORMAT NUMBER
        LD      A,10H
        CP      B                     ;   COMPARE FORMAT NUMBER = 10H
```

4

EXAMPLE DEVICE DRIVER (CONTINUED)

```
IF12:
        JP      NZ,ELSE12               ;   IF FORMAT = 10H
        INC     HL
        LD      C,(HL)                  ;      GET SIO PARM 1 TO REG.C
        INC     HL
        LD      B,(HL)                  ;      GET SIO PARM 2 TO REG.B
        PUSH    BC                      ;      STORE PARM
        INC     HL                      ;      *** SET CONSOLE BAUDRATE ***
        LD      A,TCW                   ;      GET CONSOLE TIMER MODEWORD
        OUT     (TC),A                  ;      SEND IT TO TIMER
        LD      A,(HL)
        CALL    BAUDTAB                 ;      GET THE RIGHT BAUD RATE VALUE
        LD      A,L                     ;      GET LSB OF BAUDRATE CONSTANT
        OUT     (TR),A                  ;      SEND
        LD      A,H                     ;      GET MSB
        OUT     (TR),A                  ;      SEND
        POP     BC
        LD      A,30H                   ;      MASK NOT USED BIT IN PARM 1
        AND     C                       ;      XX00XXXX  0=NOT USED,  X=USED
IF14:
        JP      NZ,ELSE14
        LD      A,1FH                   ;      MASK NOT USEN BIT IN PARN 2
        AND     B                       ;      XXX00000
        JP      NZ,ELSE14               ;      IF SIO PARM 1-2 IS VALID
        DI
;
        LD      A,03H
        OUT     (SIO+SCONT),A           ;         OUTPUT SIO CONTROL REG.
        LD      D,C                     ;         MAKE WORK COPY OF REG.-C
        LD      A,0FH                   ;         MASK OFF NOT USED BIT
        AND     D
        LD      D,(IY+MIRROR+4)         ;         GET MIRROR
        OR      D                       ;         INSERT NEW BIT IN MIRROR
        LD      (IY+MIRROR+4),A         ;         STORE MIRROR
        OUT     (SIO+SCONT),A           ;         OUT TO SIO
;
        LD      A,05H
        OUT     (SIO+SCONT),A           ;         OUTPUT SIO CONTROL REG.
        LD      D,B                     ;         MAKE WORK COPY OF REG.-B
        LD      A,60H                   ;         MASK OFF NOT USED BIT
        AND     D
        LD      D,(IY+MIRROR+5)         ;         GET MIRROR
        OR      D                       ;         INSERT NEW BIT IN MIRROR
        LD      (IY+MIRROR+5),A         ;         STORE MIRROR
        OUT     (SIO+SCONT),A           ;         OUT TO SIO
;
        LD      A,03H
        OUT     (SIO+SCONT),A           ;         OUTPUT SIO CONTROL REG.
        LD      A,0C0H                  ;         MASK OFF NOT USED BIT IN REG.-C
        AND     C                       ;
        LD      C,A                     ;         STORE USED BIT IN REG.-C
        LD      A,80H                   ;         MASK OFF NOT USED BIT IN REG.-B
        AND     B
        SRL     A
        SRL     A                       ;         ROTATE THIS BIT TO POS. 5
```

5

EXAMPLE DEVICE DRIVER (CONTINUED)

```
        OR      C                       ;          INSERT BIT FROM REG.-C
        LD      D,(IY+MIRROR+3)         ;          GET MIRROR
        OR      D                       ;          INSERT NEW BIT IN MIRROR
        LD      (IY+MIRROR+3),A         ;          STORE MIRROR
        OUT     (SIO+SCONT),A           ;          OUT TO SIO
;
        EI
        LD      C,00H                   ;          INIT OK
        JP      FI14
ELSE14:                                 ;       ELSE
        LD      C,03H                   ;          SIO PARM 1-2 NOT VALID
FI14:                                   ;       FI
        JP      FI12
ELSE12:                                 ;    ELSE
        LD      C,02H                   ;       WRONG INIT FORMAT
FI12:                                   ;    FI
        JP      FI11
ELSE11:                                 ;ELSE
        LD      C,01H                   ;    WRONG CONTROL OPCODE
FI11:                                   ;FI
        LD      A,CHANNEL               ;GET CHANNEL NUMBER
        CALL    DEQRXE
IF15:
        JP      NC,FI15                 ;IF BUFFER IN IFRXE
        LD      A,07H
        CALL    BQADLA                  ;    INSERT BUFFER TYPE = 07H
        LD      A,04H
        CALL    BQADLA                  ;    INSERT OPCODE = STATUS
        LD      A,C
        CALL    BQADLA                  ;    INSERT STATUS BYTE
        LD      A,CHANNEL               ;    GET CHANNEL NUMBER
        LD      HL,0000H                ;    SET RETURN QUE = 0000H
        CALL    ENQRXF                  ;    SEND BUFFER TO RX FULL QUE
FI15:                                   ;FI
        RET
;
;*************************************************************************
;
OUT:
        DEC     BC                      ;DEC BYTECOUNT
        INC     HL                      ;OPCODE BYTE
        DEC     BC
        INC     HL                      ;DATA BYTE
        LD      D,(HL)                  ;GET DATA BYTE
OUT1:
        IN      A,(SIO+SCONT)
        AND     CSTRDY
        CP      CSTVAL
IF20:
        JP      NZ,ELSE20
        DEC     BC                      ;DEC DATACOUNT
        INC     HL                      ;INC DATAPOINTER
        LD      (POINTER),HL            ;STORE DATAPOINTER
        LD      (COUNTER),BC            ;STORE DATACOUNTER
        LD      HL,FLAG2
```

6

EXAMPLE DEVICE DRIVER (CONTINUED)

```
        SET     0,(HL)                  ;SET BUFFER IN USE = ON
        LD      HL,FLAG1
        SET     0,(HL)                  ;SET TX IN PROGRESS = ON
        LD      A,D
        OUT     (SIO+SDATA),A           ;SEND CHR TO CONSOLE
        JP      FI20
ELSE20:
        CALL    RELEASE
        JP      OUT1
FI20:
        RET
;
;*******************************************************************
;
IN:
        IN      A,(SIO+SCONT)           ;GET STATUS FROM SIOB
        AND     CSRRDY                  ;MASK RECEIVER READY
        CP      CSRVAL                  ;COMPARE WITH READY VALUE
        RET     NZ                      ;RETURN IF NO CHR.
        IN      A,(SIO+SDATA)           ;GET INPUT
        AND     07FH                    ;STRIP PARITY
        LD      BC,0020H                ;FIFO LENGTH
        LD      L,(IX+PDATA+0)
        LD      H,(IX+PDATA+1)          ;GET FIFO START ADDRESS
        CALL    IFIFO
        RET
;
;*******************************************************************
;
;TX1INT:
TX2INT:
;TX3INT:
;TX4INT:
        EX      AF,AF'
        EXX
        LD      HL,FLAG1
        BIT     0,(HL)
IF16:                                   ;IF TX IN PROGRESS = ON
        JP      Z,ELSE16
        BIT     4,(HL)
IF17:                                   ;   IF XONOFF = OFF
        JP      NZ,ELSE17
        LD      BC,(COUNTER)            ;      GET REG.BC = COUNTER
        LD      DE,(POINTER)            ;      GET REG.DE = POINTER
        LD      A,B
        OR      C
IF18:
        JP      Z,FI18                  ;      IF COUNTER <> 00H
        LD      A,(DE)                  ;         GET DATA
        DEC     BC
        INC     DE                      ;         INC DATA POINTER
        LD      (COUNTER),BC            ;         STORE COUNTER
        LD      (POINTER),DE            ;         STORE POINTER
        OUT     (SIO+SDATA),A           ;         OUT TO SIO
FI18:                                   ;      FI
```

7

EXAMPLE DEVICE DRIVER (CONTINUED)

```
        LD      A,B
        OR      C
IF19:
        JP      NZ,FI19                 ;       IF COUNTER = 00H
        RES     0,(HL)                  ;          SET TX IN PROGRESS = OFF
        LD      A,28H
        OUT     (SIO+SCONT),A
FI19:                                   ;       FI
        JP      FI17
ELSE17:                                 ;    ELSE
        LD      A,28H
        OUT     (SIO+SCONT),A
FI17:
        JP      FI16                    ;    FI
ELSE16:
        LD      A,28H
        OUT     (SIO+SCONT),A
FI16:                                   ;FI
        EXX
        EX      AF,AF'
        EI
        RETI
;
;*************************************************************************
;
;EXT1INT:
EXT2INT:
;EXT3INT:
;EXT4INT:
        EX      AF,AF'
        LD      A,10H
        OUT     (SIO+SCONT),A
        EX      AF,AF'
        EI
        RETI
;
;*************************************************************************
;
POINTER:
        DW      0
COUNTER:
        DW      0
BUFFER:
        DW      0
FLAG1:
        DB      0                       ;BIT 0 = TX IN PROGRESS
                                        ;BIT 4 = XONOFF
FLAG2:
        DB      0                       ;BIT 0 = BUFFER IN USE
;
                                        ;SCI             MP2
;
;CHANNEL EQU    2H                      ;CH1             CON
CHANNEL EQU     3H                      ;CH2             LPT
;CHANNEL EQU    4H                      ;CH3
```

8

EXAMPLE DEVICE DRIVER (CONTINUED)

```
;CHANNEL  EQU     5H                         ;CH4
;
SIO       EQU     SIO1                       ;CH1           LPT
;SIO      EQU     SIO1+1                     ;CH2           CON
;SIO      EQU     SIO2                       ;CH3
;SIO      EQU     SIO2+1                     ;CH4
;
;TCW      EQU     0B6H
;TC       EQU     CTC1+03H                   ;CH3/4         CON
;TR       EQU     CTC1+02H
;
TCW       EQU     076H
TC        EQU     CTC1+03H                   ;CH2           LPT
TR        EQU     CTC1+01H
;
;TCW      EQU     036H
;TC       EQU     CTC1+03H                   ;CH1           SYS CLOCK
;TR       EQU     CTC1+00H
          END
```

9

EXAMPLE DEVICE DRIVER (CONTINUED)

5.9.3. HOW TO INCORPORATE A NEW PROCESS INTO SCRIMP/M

5.9.3.1 TESTING THE NEW PROCESS

1. Write the source code! Some guidance was given in earlier parts of section 5.9. Assemble it using M80 or similar.

2. Link the modules, using the L80 linker, in the following order:

   KERNEL, ASMIF, BQM, TQM, START, SUB, BOOT, CONFIG, SYSRAM, USERRAM, CON, LPT, FLP, Your new Modules.

3. Attempt to run the whole thing as a job under CP/M on the Z80A. The executable file can be tested with DDT like any other user program, since the linker will output a file whose code's starting address is 100H.

5.9.3.2 INTEGRATING THE NEW SYSTEM ONTO THE SYSTEM TRACKS

When the new system appears to be working correctly, it must be located on the system tracks of drive B, C or D (the 8088 processor's system resides on drive A). The sequence of operations to do this is as follows:

1. Bring in DDT again by typing:
   ```
   DDT
   ```

2. Read in the executable file, with an offset of 3580H. Type:
   ```
   I Filename.COM
   R3580
   ```

   DDT will then read in the file and place it at starting location 3680H since the file itself starts at 0100H. Note down the output of DDT when it displays the address of the next location. Let us call this NL.

3. Reset the area from 100H to 367FH by typing:
   ```
   F100, 367F, 00
   ```

4. Next, some system information has to be written to the locations starting at 3480H. These are:

```
3480:     00     Load Address
          01
          XX     System Length
          XX
          08     Console Baud Rate
          00
          08     Line printer Baud Rate
          00
          24
          24
          01
          01
          00     Start Address
          10
```

   These parameters can be set up using the S command in DDT.

5. Terminate DDT by pressing <CTRL-C>

6. The load file is now ready for placing on the two system tracks; the old system should, however, be saved in case the new system does not work. The value NL from step 2 must be used. This value must be rounded up to the next multiple of 100 Hex, divided by 100 Hex, then translated into decimal to give the number X. For example, if NL was 5378H.

   5378H -> 5400 -> 54H -> 84 = X.

7. Type:

   SAVE X Filename.BIN

   using the appropriate value of X and a suitable file name.

8. Finally, to move the new system to the system tracks, type:

   GETPUT

   PY

   where Y is the drive name (B, C or D).

5.9.4 COMMUNICATION BETWEEN SCRIMP/M AND MP/M

It can be seen from the SCRIMP/M 'Software Structure' diagram back at the beginning of section 5.9 that the two processors on the MP^2 card communicate via a shared RAM area belonging to the Z80A. The addresses involved are shown in the following table. They start at the symbolic address SHARAM: (8000H)

8088/Z80 Interface

Z80A Addresses (Hex)	Function
8000 to 8020	Reserved
8020 to 805F	Channel 0. Device Testing, initializing system, system commands
8060 to 809F	Channel 1. Spare
80A0 to 80DF	Channel 2. Console Driver
80E0 to 811F	Channel 3. Line Printer Driver
8120 to 815F	Channel 4. Floppy Driver
8160 to 829F	Channels 5 to 9. Spare

The interface in use is the LTU (Line Terminal Interface) as used in the CR80 minicomputer.

5.10 SETTING BAUD RATE AND OTHER CHARACTERISTICS OF TERMINALS

5.10.1 CR7 COMFORT

1. Switch off and unplug the terminal from the mains.

2. Remove the narrow panel at the lower rear of the main part of the terminal just above the mains switch.

3. This will reveal the edge of the video processor printed circuit board. Three rows of DIL switches (small blocks of 8 switches each) may be seen on the lower side of the board at the edge. Not all of these switches are concerned with Baud Rate; only one of the 9 Baud rate switches must be closed (i.e. 'ON'). The following list describes all the switches reading from left to right (when looking at the edge of the board in its normal position).

CR7 and CR5 Switch Settings

First Block (marked DIL 3)

Switch No.	Function
1 (left)	Off: DTR communication. On: XON/XOFF communication.
2	Off: normal. On: Self test executes (Off-line)
3	Off: steady cursor. On: Blinking cursor
4	Off: cursor is an underline. On: Block cursor.
5	Off: Typed-in characters not displayed. On: Normal echo.
6	Should be set to On.
7	Should be set to Off.
8 (right)	Should be set to On.

CR7 and CR5 Switch Settings (continued)

Second Block (DIL 2)

Switch No.	Function	
1 (left)	Off: local	On: On-line
2		On: 75 Baud
3		On: 110 Baud
4		On: 150 Baud
5		On: 300 Baud
6		On: 600 Baud
7		On: 1200 Baud
8 (Right)		On: 2400 Baud

Third Block Right hand side (DIL 1)

Switch No.	Function	
1 (left)		On: 4800 Baud
2		On: 9600 Baud
3	Not used	
4	Off: 2 stop bits On: 1 stop bit	
5	Off: Even parity On: Odd parity	
6	Off: Enable parity On: Disable parity	
7 and 8 (right)	Both Off: 8-bit data. 7 Off & 8 On: 7-bit data 7 On & 8 Off: 6-bit data. Both On: 5-bit data.	

If there is any doubt as to which direction on the switches is 'ON', observe the 3rd switch going from left to right. If it is ON, the cursor on the screen will blink.

5.10.2 CR5 COMPASS

The procedure for setting the Baud Rate, etc. is virtually the same since the same video card is used. The only differences are:

1. Access is from the front of the terminal (by pulling off the ventilation grille below the screen).
2. In some examples jumpers are used rather than switches.
3. The jumpers or switches are on the upper (component) side of the board.

5.10.3 TANDBERG TDV

Baud rate selection on this terminal is effected very simply from the keybaord. Hold down SHIFT and press MODE twice. A menu appears .which is self explanatory. The Baud rate is selected using the cursor control keys. Press ENTER when the selection is made. Full documentation is supplied with the terminal.

5.10.4 INTEGRATED WORKSTATION

This was described in detail at the end of section'5.2.2. Switches 4 to 1 in the lower bank of DIP switches inside the workstation are used for Baud rate selection. If all four are in the right hand position, Baud rate is 9600.

6. LIST OF ABBREVIATIONS

BCD	Binary Coded Decimal
BSC	Binary Synchronous Communications
CCITT	International Telegraph & Telephone Consultative Committee
COBOL	COmmon Business Oriented Language
CP/M	Control Program/Microcomputer. A single task, single user operating system used (in the CR8) in the Z80 or 8088 microprocessors.
CPU	Central Processor Unit
CR80	Minicomputer manufactured by Christian Rovsing A/S, often used as front-end processor.
CR	Christian Rovsing A/S, manufacturer of the CR8
CR8	CR8/16 Microcomputer
DML	Data Manipulation Language (part of DBMS System)
DTE	Data Terminal Equipment
FIFO	First In, First Out
HDLC	High Level Data Link Control
I/F	Interface
JCL	Job Control Language used with IBM mainframes
LED	Light Emitting Diode
LIFO	Last In, First Out
LTU	Line Terminating Unit - a microprocessor controlled serial I/O device for the CR80.
MDBS	Micro Data Base System - an application software package for the CR8.
MFM	Modified Frequency Modulation
MP^2	Multi Purpose Multi Processor Board
MP/M	Multi-user, Multi-tasking version of CP/M
NMI	Non-Maskable Interrupt
OEM	Original Equipment Manufacturer
PAM	Process Addressable Monitor
PCB	Printed Circuit Board
QRS	Query Retrieval System (The query language that forms part of DBMS data management system).
RAM	Random Access Memory
SASI	Shugart Associates System Interface

SCRIMP/M	Subset Comparable Real-time Implementation of MP/M
SCI	System Communication Interface
SDLC	Synchronous Data Link Control
TDX	X-Net Adaptor
VDU	Visual Display Unit (a type of terminal)

7. INDEX

Some Notes on the Use of this Index.

1. The first page number(s) mentioned for each topic refers to the main entry (if any) for that topic.

2. The notation (for example) 5-66..68 means that the reference starts on page 5-66 and continues through to page 5-68. On the other hand, 5-66, 5-67, 5-68 means that there are one or more separate references to the topic on each of these three pages.

3. Please note the different uses of the semicolon and comma. When listing references to other subject names in the index, whole entries are separated by semicolons, while subsections are specified by commas. For example, the entry:
    scheduling - see SCRIMP/M, CPU management
   means 'refer to the CPU management subsection of SCRIMP/M', whereas
    scheduling - see SCRIMP/M; CPU management
   means 'there are two references, one under 'SCRIMP/M' and the other under 'CPU management'. Similarly, 'see X,Y,Z' means 'see subsection Y and Z of X'.

READERS' COMMENTS

If you find any 'bugs' in this manual or have any comments of a more general nature, we would be pleased to hear from you.

INACCURACIES FOUND:

SECTIONS THAT WERE DIFFICULT TO UNDERSTAND (please give page numbers):

OMISSIONS:

ANYTHING ELSE:

Your name and address:

Please send this form to: CHRISTIAN ROVSING A/S, Development Division, Ref.: NIM, Lautrupvang 2, DK 2750 Ballerup, Denmark.