PHILIPS

P855M/P860M System Description

Data
Systems

PHILIPS

P855M/P860M System Description

MAIN FEATURES

- CYCLE SPEED OF 0.84 MICROSECONDS
- MEMORY SIZE OF 4K UP TO 32K WORDS
- 16-BIT WORD ORIENTED
- 16 GENERAL PURPOSE REGISTERS
- 98 INSTRUCTIONS
- DIRECT, INDIRECT, INDEXED, INDEXED INDIRECT ADDRESSING
- UP TO 48 INTERRUPT LINES
- LOW AND HIGH SPEED DATA CHANNELS (INCL. DMA AND MIDB)
- INTERFACES FOR INDUSTRIAL EQUIPMENT
- DATA COMMUNICATION
- REAL TIME CLOCKS
- AUTOMATIC STACK HANDLING
- MEMORY PROTECTION
- HARDWARE MULTIPLY/DIVIDE
- POWER FAILURE DETECTION WITH AUTOMATIC RESTART
- POSSIBILITIES TO CONNECT ALL STANDARD PERIPHERALS, INCLUDING CASSETTE TAPE
- EXTENSIVE SOFTWARE PACKAGE INCLUDES:

  BASIC AND BASIC REAL TIME MONITORS

  DISC AND DISC REAL TIME MONITORS

  ASSEMBLERS, FORTRAN COMPILERS, LINKAGE EDITORS

  UPDATE PACKAGE, TEXT EDITOR, DEBUGGING PACKAGE

Th. C. Ensel

P855M/P860M

System Description

IMPORTANT

From November 1st, 1973 onwards, the performance of the P855M minicomputer has been improved to that of the P860M model. In this and all companion manuals, performance figures shown for the P860M model therefore also apply to the improved P855M. The P800M Minicomputer Equipment Catalogue has also been updated to reflect this improvement, at the same time bringing the product offering more in line with market demand. Thus a number of standard CPU configurations, including a range of memory module sizes, control panels, slides and other options may be ordered by a single type number.
Although the P860M has been removed from this new catalogue, it is still available on an 'on request' basis.

A publication of

Philips-Electrologica B.V.
Main Marketing Group OEM
Apeldoorn, Netherlands

Publication number 5122 991 11372

October 1973

Printed in The Netherlands

Preface

This handbook contains general information on both the hardware and software aspects of the P855M and P860M computers. It is intended to serve as a guide to the most important possibilities and features available with these computers. That includes short descriptions of the standard peripherals which may be connected.

For more detailed information, a set of handbooks is available: a System Introduction, giving a more detailed analysis of the features described in this book; an Interface Manual describing the interfacing possibilities; an Installation Manual, providing directions and details for installing a system; a set of service manuals for CPU and control units; manuals providing the necessary information about Data Communication with a P855M or P860M; a set of software manuals with detailed information about the available Monitors, Software Processors and Instruction Set, about Basic and Full FORTRAN and about the FORTRAN Mathematical Library.
The appendix on page 95 gives a complete survey of the P800M manuals.

To understand the information contained in this book, readers have only to be familiar with the general principles of computers and programming.

Great care has been taken to ensure that the information in this handbook is accurate and complete. However, should any errors or omissions be discovered, or should any user wish to make a suggestion for improving this handbook, he is invited to write his comments on the sheet provided at the end of the book and send it to:

Manual Writing Small Computers,
at the address on the opposite page.

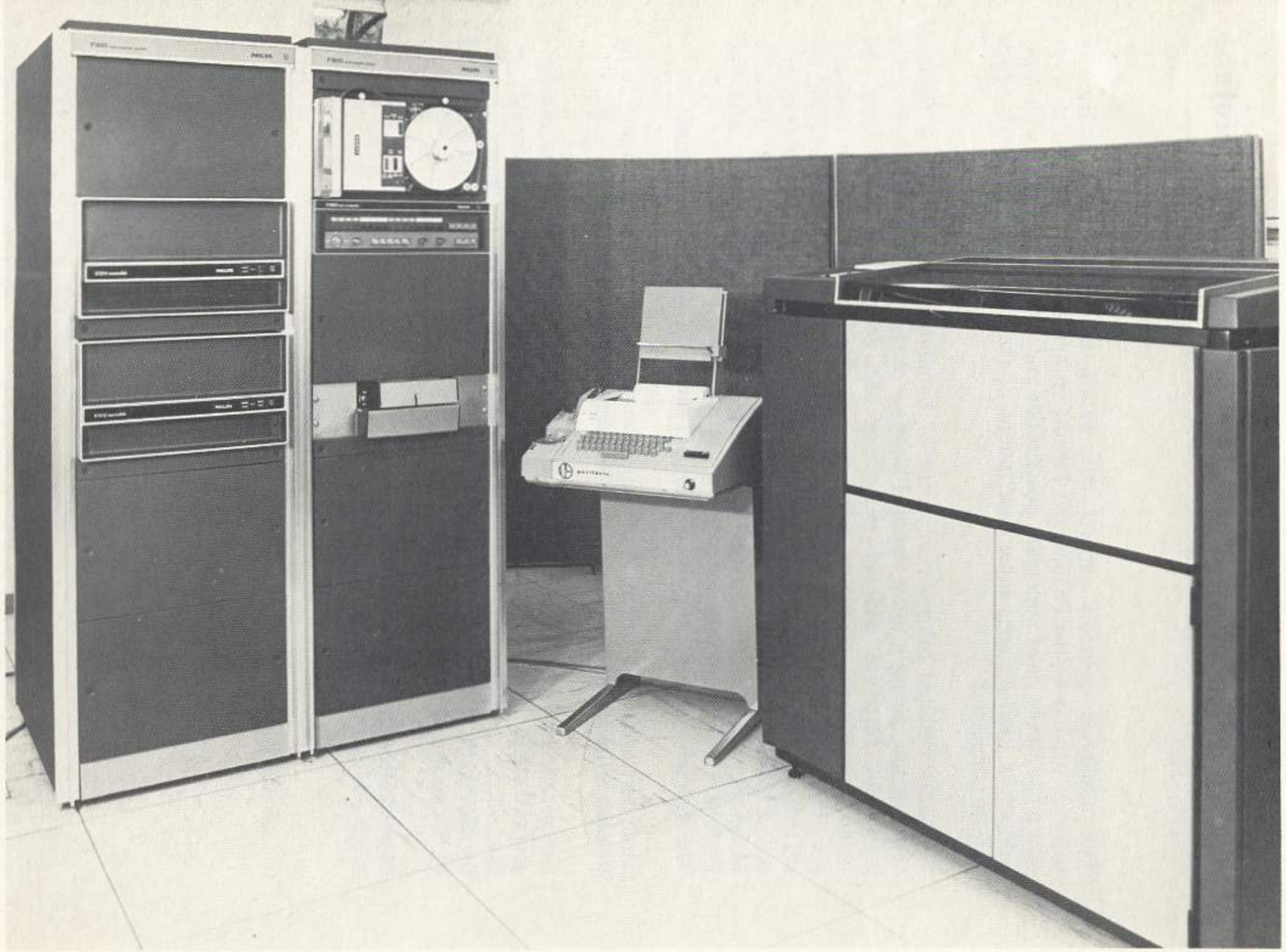


Table of contents

Introduction

The P855M and P860M are general-purpose digital mini-computers ideally suited for industrial and scientific applications, such as process control, test and measurement, data acquisition, numerical control and scientific problem-solving.
The basic central processor of both machines contains at least 4k of memory; it can be expanded up to 32k 16-bit words.

A number of I/O facilities are available: programmed channel is standard and, depending on the type of peripheral equipment connected, Multiplex Channel, Direct Memory Access (DMA) or Memory Increment Data Break (MIDB) can be added. For analog or digital processes MIOS or DIOS can be used to connect external devices to the P855M or P860M.

The basis of the interrupt system is made up of eight interrupt levels one of which is programmable by software and able to accept up to 16 signals (to be inhibited or enabled by means of a mask register). The others accept one signal each. The standard system can be expanded with groups of additional interrupt lines. The maximum interrupt system can accept 63 (47 + 16) interrupt signals hierarchized on 48 levels.

To increase the capability of the central processor, a number of extra hardware features are available, such as Real Time Clock, Memory Protection, Power Failure/Automatic Restart and Hardware Arithmetic.

The instruction set used for programming in Assembly language is based on the 16-register structure of the central processor. This makes it a very powerful and versatile set, providing the programmer with a wide range of possibilities and ensuring fast execution of his programs. Different monitors are available for various applications. For non-disc configurations a Basic and a Basic Real Time Monitor can be supplied. For disc systems there are also two different monitors: a Disc Real Time Monitor and a Monitor for non-real time disc systems, mainly intended as a program development tool.

Both machines accept the same Assembly language, with additional facilities for machines with 8k or more words of memory. FORTRAN is also available in different versions. Updating can be done on the statement level or, by means of the Text Editor, on the character level. Finally, a Linkage Editor and a Program debugging Package are supplied.

All these features are treated in more detail in the following sections of this book.

The following figure shows the standard and optional features of the P855M and P860M computers. The dotted lines indicate options.

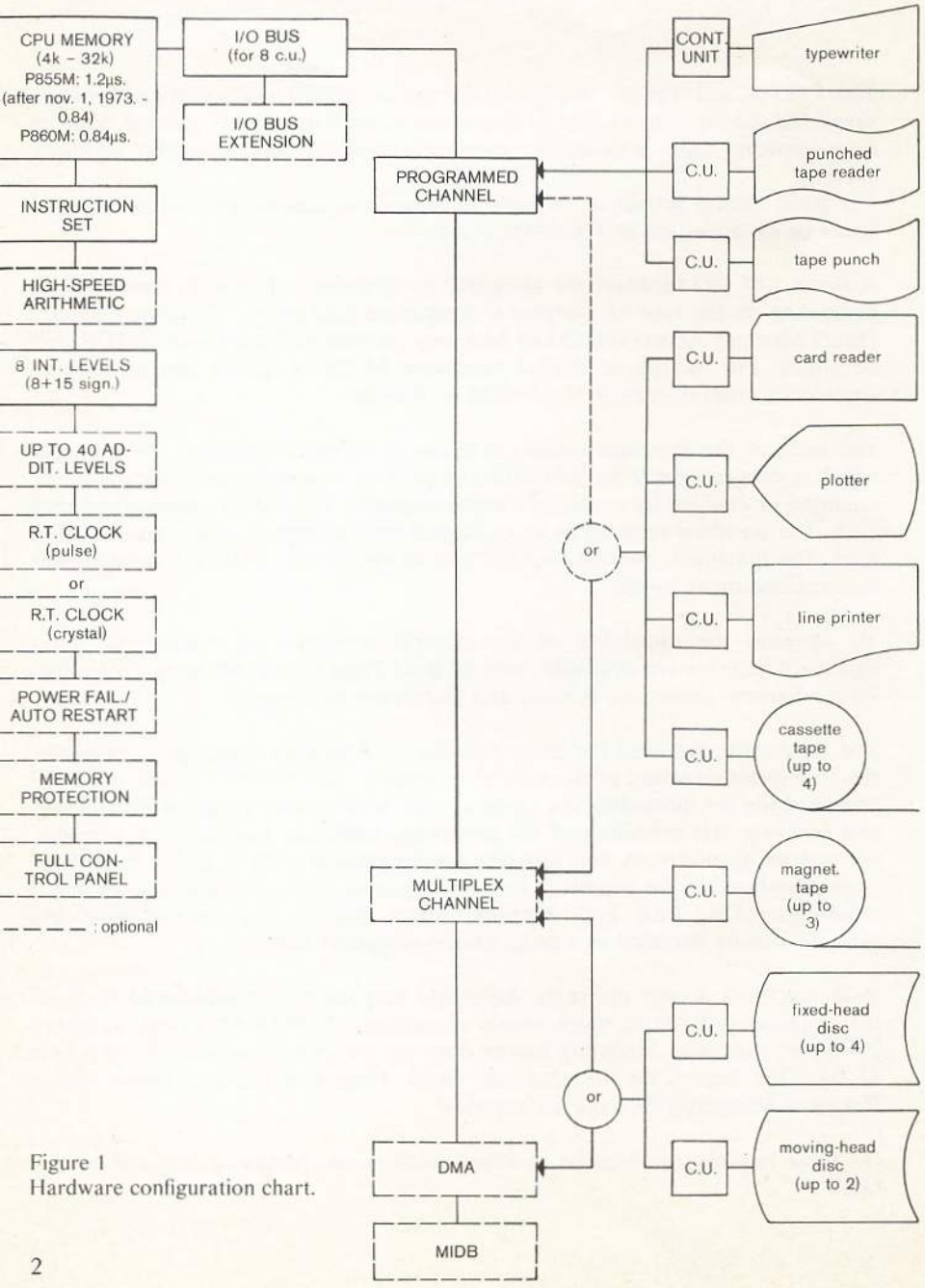


Figure 1
Hardware configuration chart.

CPU SPECIFICATIONS

Memory Size:	4k to 32k 16-bit words
Cycle Speed:	P855M: 1.2 microseconds (after Nov. 1, 1973: 0.84) P860M: 0.84 microseconds
General Purpose Registers:	16
Number of I/O slots:	8
Instruction Set:	98 instructions (86 standard, 12 optional)
Addressing Mode:	direct, indirect, indexed, indexed indirect
Central Processor Format:	two's complement binary
Word Format:	16-bit
Memory Type:	coincident current ferrite core read/write
Standard Hardware:	system stack stack overflow interrupt
Electrical Requirements:	110 or 220 Volts/50 or 60 Hz
Power Consumption:	minimum 500 VA maximum 700 VA
Technology:	TTL/MSI
Size (incl. power supply):	width: 19 inches (483 mm) height: 400 mm minimum 489 mm maximum depth: 540 mm
Weight (incl. power supply):	65 kg minimum 80 kg maximum
Environment:	0-45 °C ambient temperature
Relative Humidity:	0-85% without condensation

MEMORY UNIT

The standard memory unit uses ferrite cores to store data. The data path is 16 bits wide and data are transferred in parallel. The internal temperature and humidity of the memory unit are constantly monitored and if they exceed acceptable limits, the unit is made inoperable without disturbing existing data. Switching on and off the power supply to the memory unit has been designed so that stored data are not affected.

Capacity:	minimum 4k words, which can be expanded up to 32k in 4k modules.
Cycle Time:	The time taken for one complete read/write cycle is: P855M: 1.2 microseconds (after Nov. 1, 1973: 0.84), P860M: 0.84 microseconds

Register Structure
Figure 2 shows a simplified block diagram of the main hardware units and registers of the central processor.

The following of these registers are accessible by software:

P-register:	A 15-bit register (register 0) used as an instruction counter to contain the address of the next program instruction to be executed. The contents of this register are incremented by 2 each time a one-word instruction is executed, by 4 each time a two-word instruction is executed.
Registers 1 to 14:	Fourteen 16-bit general-purpose registers which can be used as accumulators (to contain the intermediate results of computation), as address or index-registers or as I/O registers.
Register 15:	A 16-bit register used as a stack pointer for the interrupt system.
CR register:	A 2-bit register used to indicate a condition after the execution of some instructions.
GC register:	A general indicator, 4 to 8 bits long, used to indicate control or interrupt states. The bits can be individually set or reset by program.
IM register:	A 16-bit register which can be used as an interrupt mask to inhibit or enable interrupt signal lines on the common interrupt line.
MK register:	One or two optional 16-bit registers which can be used as memory protect keys if the memory protect option is included in the system.
PL register:	A 6-bit register used as a priority level register.

Data Flow
The basic data flow paths between hardware units is shown simplified by figure 2, and the flow for particular functions is shown with reference to this diagram, following the explanations of the units.
BAD: Bus Address Lines
BIN: Bus Input Lines
BOF: Bus Function Lines
BOU: Bus Output Lines

DMA: Direct Memory Access
MIDB: Memory Increment Data Break
BR: Break Request Line (multiplex)
IR: Interrupt Request Line
EOR: End-Of-Range (multiplex)

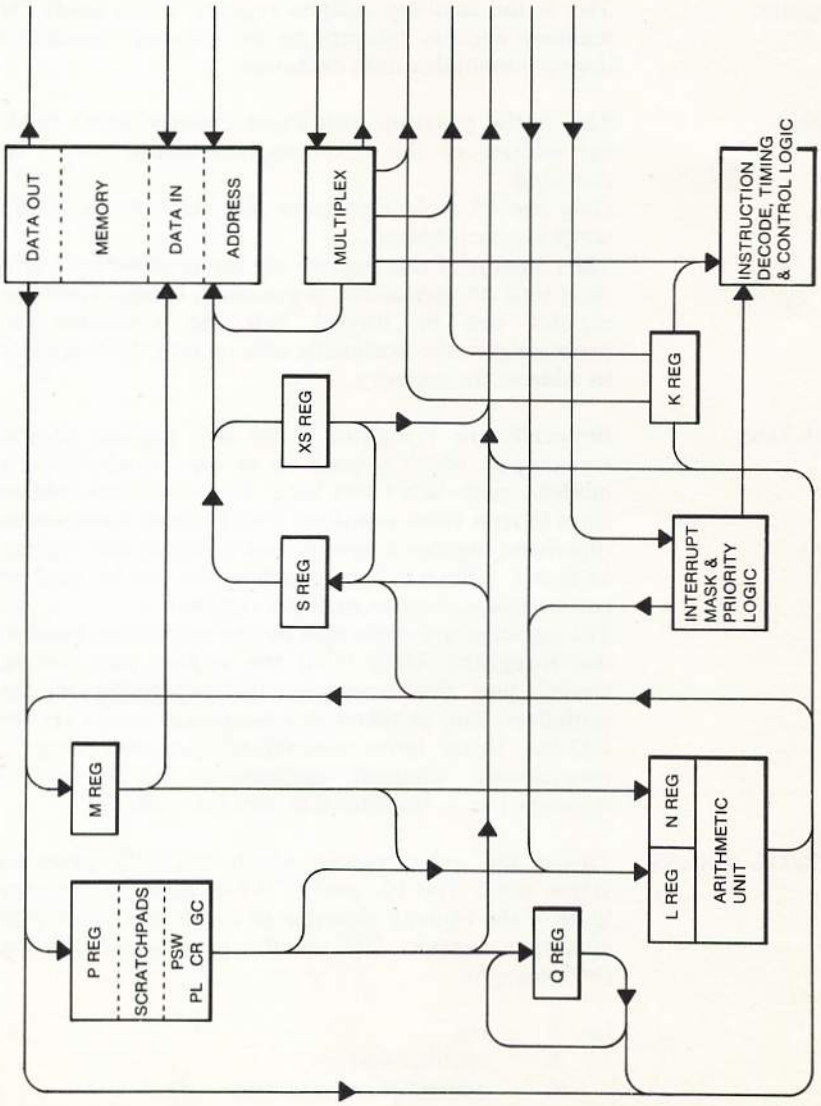


Figure 2 Simplified block diagram of CPU and data flow

Arithmetic Unit: The arithmetic unit data path is 16 bits wide. All bits are processed in parallel. It performs all arithmetic and logical functions in true binary, using two's complement notation.

S-register: This is a register which holds the memory ddress.

XS-register: This is the auxiliary address register which holds the memory address relevant to the running instruction during a multiplex data exchange.

P-register: This is the program instruction counter which holds the address of the next program instruction to be executed.
Only the 15 most significant bits refer to the 16-bit word memory address.
The contents of this register are incremented by 2 or 4 each time an instruction is executed. Output from the register can be loaded into the L-register for processing in the arithmetic unit or into the S-register to address the memory.

Scratch Pad: Physically the P-register is the first register of this scratch pad, which is made up of four random-access modules each 16×4 bits long. They are hardwired to form sixteen 16-bit registers. Two of these have unique functions, register 0 (instruction counter) and register 15 (stack pointer). The remaining 14 can be used as accumulators, address or index registers.
The registers are addressed by the instruction word in the K-register. Data from the scratch pad can be loaded into the L-register for processing in the arithmetic unit, or taken to a peripheral device via the I/O bus. In the latter case scratch pad addressing by programmed channel instructions is limited to registers 1 to 7. Register 0 is used for multiplex.

PL/CR/GC-registers: This is the 16-bit register which holds the program status word. The PL part of 6 bits gives the priority level of the running program (0 to 63). CR is the 2-bit condition register. GC are the general control bits (maximum 8):

bit 8 : run
9 : enable interrupt
10 : control panel interrupt

11 : power failure
12 : real time clock
13 : program interrupt (LKM, stack overflow, illegal instruction)
14 : memory protect error
15 : user/system mode indicator

Q-register: This is a 16-bit register used during double-length shift operations, and multiply/divide.

L-register: Input register for the arithmetic unit (first operand register).

N-register: Input register for the arithmetic unit (second operand register).

K-register: This register holds the instruction read from memory. Output from the register is to the instruction set logic and, in the case of I/O instructions, to the address and control lines (BAD and BOF) of the I/O bus.

M-register: This is a 16-bit memory buffer register to contain data transferred to and from the memory.

Examples of data flow in relation to figure 2.

Figures 2a to 2d show diagramatically the data flow involved in accessing an instruction and certain operands, carrying out an arithmetic action, and placing the final result.

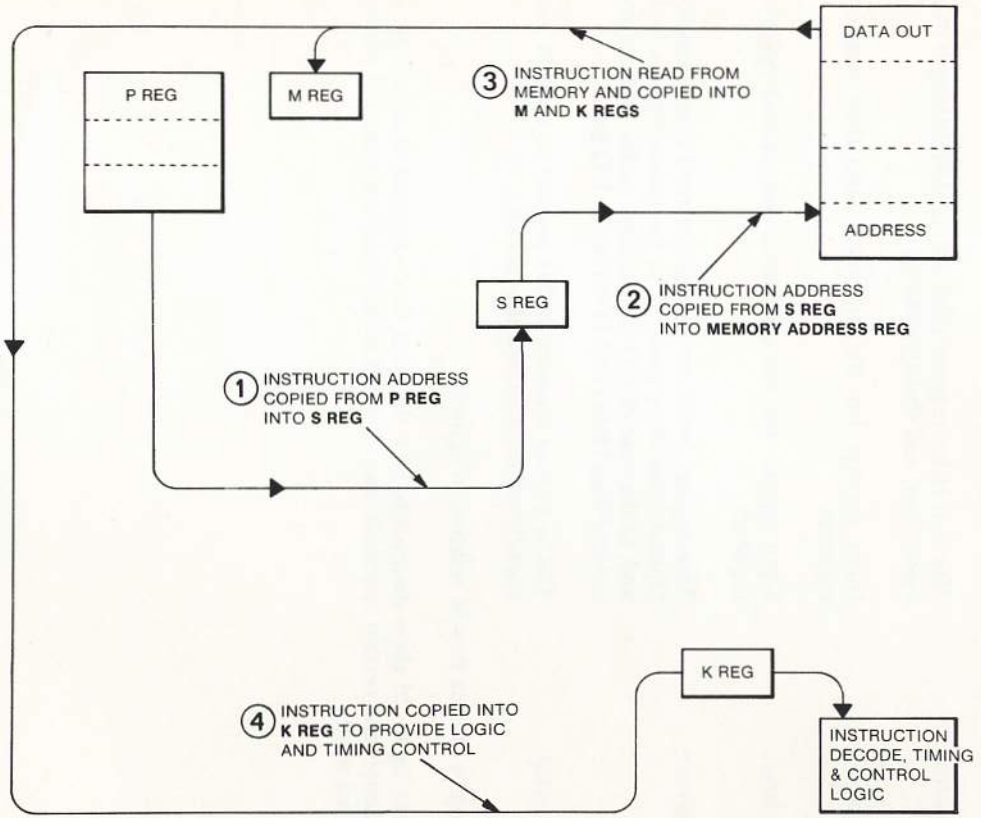


Fig.2a Access an instruction and place it into K. reg. ready for execution

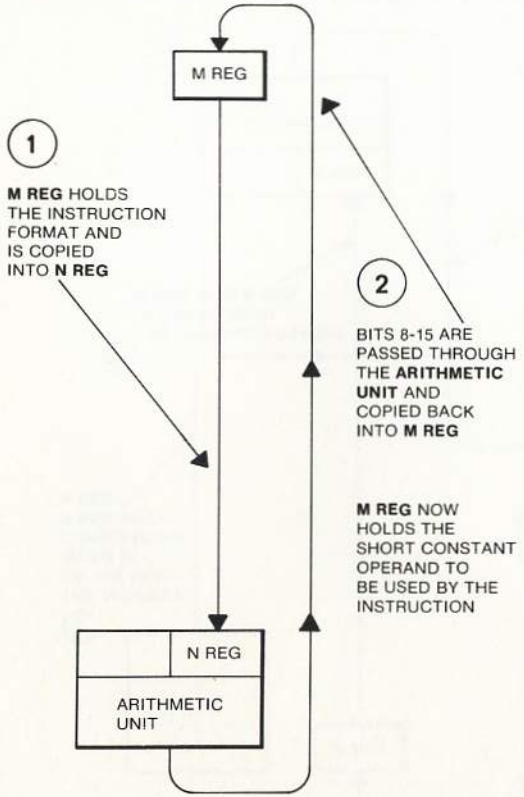


Fig.2b Access a short constant operand and place it into M. reg. ready for use

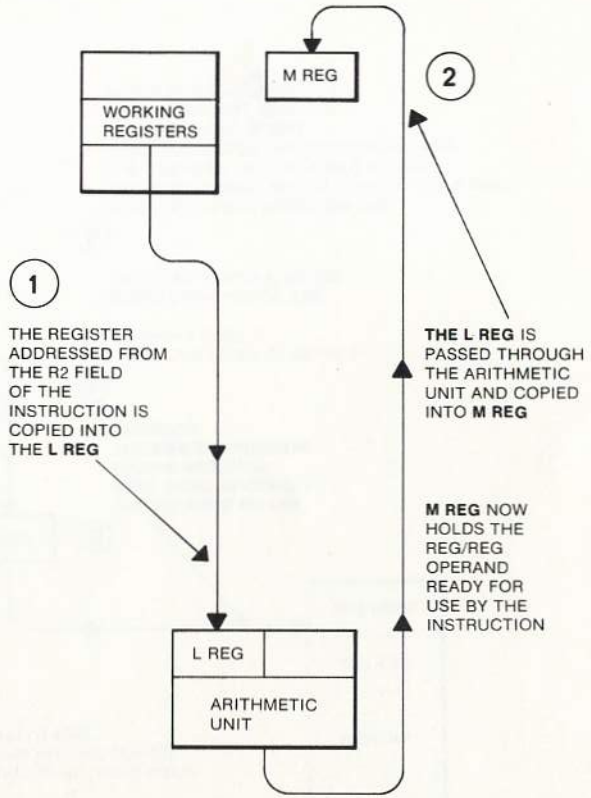


Fig.2b Action a reg/reg operand and place it into M. reg. ready for use

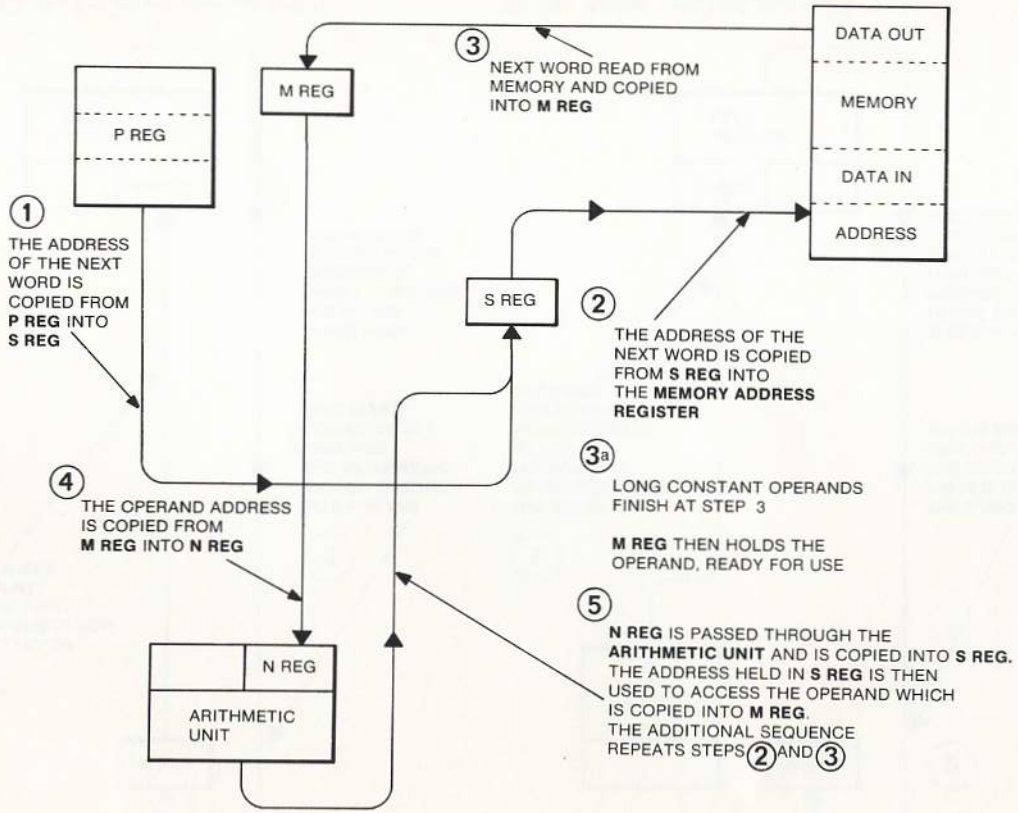


Fig. 2c Accessing long constant and directly addressed operands

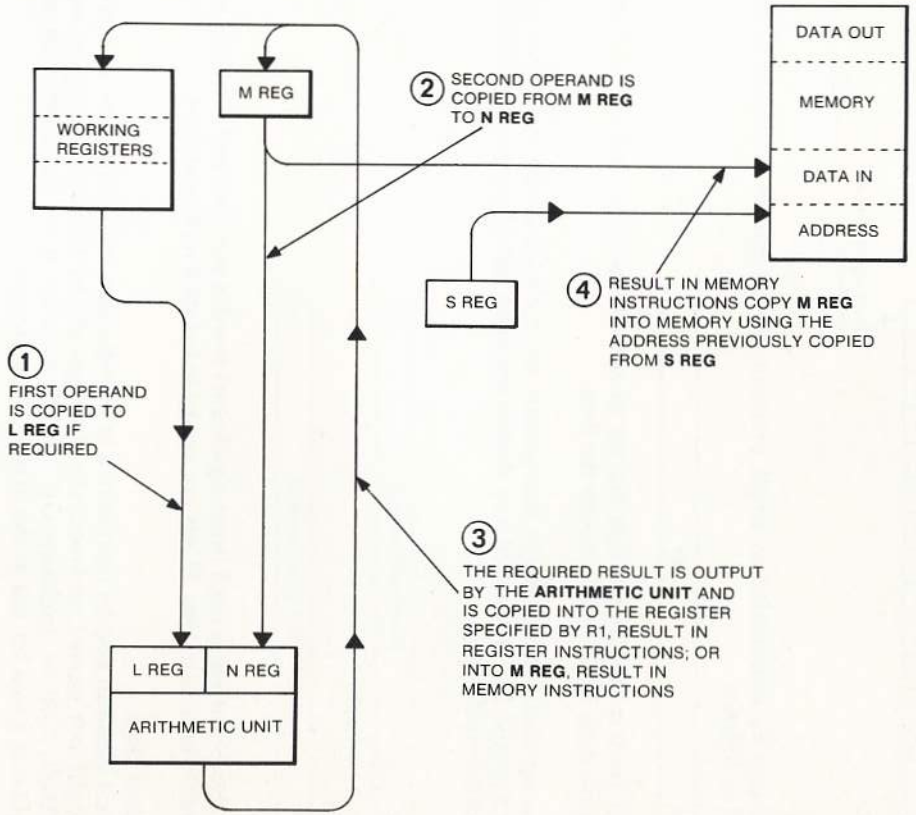
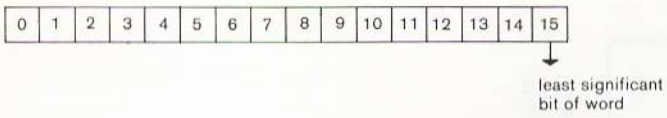


Fig. 2d Arithmetic action placing results in a register and in memory

DATA FORMAT

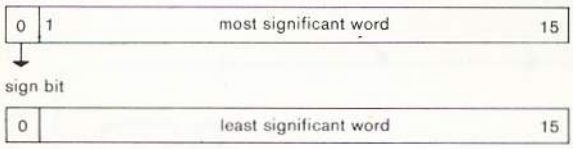
The basic format used to transfer data into or out of the central processor is the 16-bit word. Bit positions are numbered 0 to 15 as follows:



Data may be represented as single precision signed integer contained in one word as follows:

where bit 0 is used as the sign bit, set to zero for positive data and to 1 for negative data. Bits 1 to 15 contain the data.

If the optional double length instructions are included in the instruction set, data may also be represented as a double precision signed integer, contained in two words as follows:



The sign bit of the second (least significant) word is not used and is usually set to zero. This means that 30 bits are available for data representation.

Logical Data
Logical data can also be represented in a single data word. This type of data is generally not treated arithmetically, but logically by boolean operations such as 'AND', 'OR' or 'exclusive OR'. In this case bit 0 of the word does not function as a sign bit, but as the first of 16 conditions.

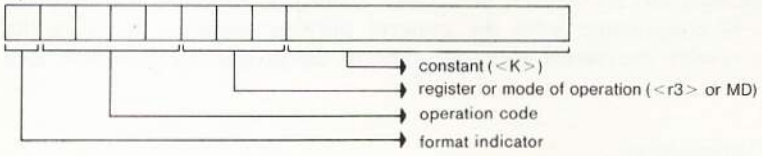
INSTRUCTION SET

The instruction set includes load, store, branch, arithmetic, shift, logical and I/O instructions (86 total). On option, the set may be extended with 12 instructions for double length arithmetic, multiply and divide.

Instruction Formats
There are two basic instruction formats which are indicated by the value of bit 0 in the instruction word.

Format 0:

These instructions are contained in one word. Bit 0 is set to zero to indicate this type of instruction.

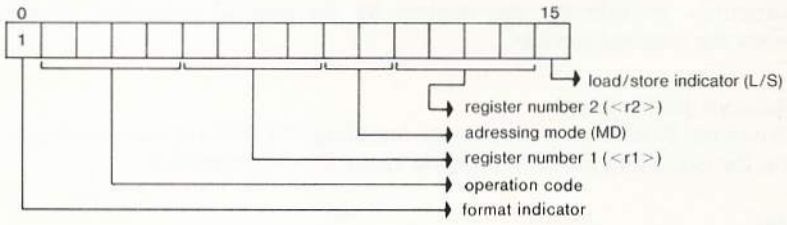


The types of instruction which can use this format are:

- constant instructions (short format)
- shift instructions
- branch instructions
- control instructions
- I/O instructions
- miscellaneous instructions

Format 1:

These instructions are contained in one or two words. Bit 0 is set to 1 to indicate this type of instruction.



The types of instruction which can use this format are:

- register-to-register instructions
- memory reference instructions
- constant instructions (long format)
- branch instructions
- miscellaneous instructions

Instruction set
The instruction set can be divided into six main groups:

– Memory Reference Instructions
These instructions are all written in format 1, with the exception of the relative branch instructions. The result can be stored either in memory or in a register.

– Register-to-Register Instructions
These instructions are all written in format 1 and by optimum use of these instructions, in conjunction with the general purpose registers to store the intermediate results, the overal execution time of the program can be held to a minimum.

– Constant Instructions
These instructions can be written in either format 0 or format 1. In format 0 instructions the constant is contained in the least significant 8 bits of the instruction word, in format 1 instructions all sixteen bits of the second word of the instruction can be used to contain the constant.

– Shift Instructions
These instructions allow the contents of the registers to be shifted a number of positions to the right or left. For logical data all 16 bits are shifted.
Shifting can be circular or not, i.e. it is possible to reintroduce bits shifted out at one end, into the other end of the register. When arithmetic data are shifted, the sign bit is left unchanged and the instruction is the same as multiply or divide by 2 times the number of positions shifted.

– Input/Output Instructions
These instructions provide for the control by the central processor of the operations on the external devices.

– Miscellaneous Instructions
These instructions include possibilities for handling the interrupts, executing a halt, setting the operating mode and giving control to the monitor.

Addressing
In addition to direct addressing, for memory reference instructions, the effective address may be specified by using indexed addressing, indirect addressing or a combination of both.

The figure of the layout of a format 1 instruction in the previous paragraph shows the use of the different fields of the instruction. Depending on the use of the mode indicator and register number 2 fields, the effective operand address may be one of the following:

	Register addressing	Direct addressing	Indexed addressing	Indirect addressing	Indexed indirect addressing
mode (bits 8-9)	00	10	10	11	11
register <r2>	≠0	=0	≠0	=0	≠0
effective memory address	<p>	<q>	<q> + <p>	contents of location addressed by <q>	contents of location addressed by <q> + <p>

<p>: address part of the instruction
<q>: contents of a register (A0-A15)
<r2>: register number 2 (see Format 1)

Execution Times

The execution time of an instruction depends on a number of variables. For instance, format 1 instructions can be contained in either one or two words, the result of the instruction can be stored either in a register or in memory and the mode of addressing may also affect the execution time of an instruction. Individual instructions all need a certain amount of time in which to perform their function and when calculating the execution time of a program, the extra time needed by the variables must be taken into account.

It will be seen from the table on the following pages that indexing does not increase the execution time, although the use of indirect addressing does increase execution time by 0.84 microseconds.

List of symbols used in Instruction Set:

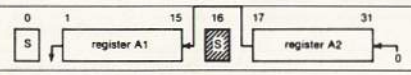
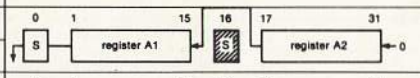
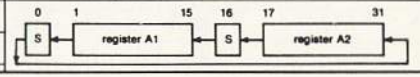
L/S bit	:	load/store bit: 0 = load; 1 = store	l.c.	:	logic cycle
mode	:	addressing mode	r	:	right
n.s.	:	not significant	l	:	left
M	:	memory	÷	:	comparison
P	:	P-register	A1	:	register A1
R2	:	bits 11 to 14 of instruction word (register 0 - 15)	A2	:	register A2
			s	:	sign bit
q	:	contents of P (i.e. the word after the instruction)	∀	:	exclusive or
			∧	:	logical and
K	:	constant	∨	:	logical or
KL	:	long constant	$\overline{M}$	:	one's complement
R3	:	registers 0 - 7	D.A.	:	device address
R1	:	registers 0 - 15	DCU	:	device control unit
PSW	:	Program Status Word	MP	:	memory protection
CR	:	condition register	( )	:	contents of
m.c.	:	memory cycle	[ ]	:	indirect addressing

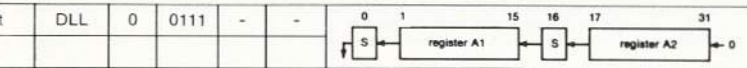
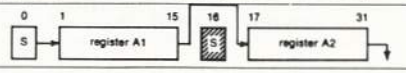
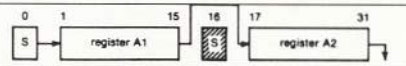
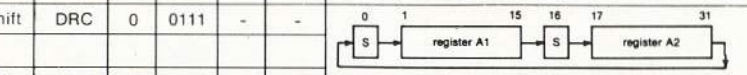
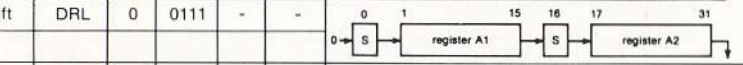
name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
Absolute branch	ABI	1	0001	10	0	(M) → P			3	bits 5-7: condition
				10	0	(M + (R2)) → P			3	bit 8: n.s.
				11	0	((M)) → P			4	
				11	0	((M + (R2))) → P			4	
					0	no branch: (P) + 4 → P		1	1	
Absolute conditional	AB	0	0001	-	n.s.	K → P		1	1	short format
branch (with constant)						(q) → P	3)			
						no branch: (P) + 2 → P		1	1	
Absolute conditional	ABL	1	0001	01	0	KL → P		1	2	long format
branch (with constant)						no branch: (P) + 2 → P		1	1	
Absolute conditional	ABR	1	0001	00	n.s.	(R2) → P		1	1	bits 5-7: condition
branch to register				01	0	((R2)) → P			2	bit 8: n.s.
						no branch: (P) + 2 → P		1	1	
Add constant	ADK	0	0010	-	-	(R3) + K → R3		1	1	short; R1 = 1111:
	ADKL	1	0010	01	0	(R1) + KL → R1			2	long system mode
Addition	AD	1	0010	10	0	(R1) + (M) → R1			3	R1 = 1111: system mode
				10	1	(R1) + (M) → M	2)		4	when l/s bit = 1,
				10	0	(R1) + (M + (R2)) → R1			3	R1 must be ≠ 0

				10	1	(R1) + (M + (R2)) → M			4	
				11	0	(R1) + ((M)) → R1			4	
				11	1	(R1) + ((M)) → M			5	
				11	0	(R1) + ((M + (R2))) → R1			4	
				11	1	(R1) + ((M + (R2))) → M	2)		5	
Addition/register	ADR	1	0010	00	n.s.	(R1) + (R2) → R1		1	1	when l/s bit = 1,
				01	0	(R1) + ((R2)) → R1			2	R1 must be ≠ 0;
				01	1	(R1) + ((R2)) → (R2)			3	R1 = 1111: system mode
Call function	CFI	1	1110			(P) → (R1), (R1) − 2 ⟶ R1				R1 = 1111: system mode
						(PSW) → (R1), (R1) − 2 → R1				
						then:				
				10	1	(M) → P		2	5	if reg. 15 = stack
				10	1	(M + (R2)) → P		2	5	pointer,
				11	1	((M)) → P		2	6	$>128_{10}$ → overflow
				11	1	((M + (R2))) → P		2	6	
Call function/constant	CF	1	1110	01	1	(P) → (R1), (R1) − 2 ⟶ R1	3)	2	4	if reg. 15 = stack
						(PSW) → R1, (R1) − 2 ⟶ R1,				pointer
						KL → P				$>128_{10}$ → overflow;

name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
Call function/register	CFR	1	1110			(P) → (R1), (R1) - 2 ⟶ R1				if reg. 15 = stack
						(PSW) → (R1), (R1) - 2 → R1				pointer, $>128_{10}$: overflow
						then:	3)			R1 = 1111: system mode
				00	1	(R2) → P		3	3	
				01	1	((R2)) → P		2	4	
Compare characters	CC	1	1101	10	1	(R1)r ÷ (M) l/r ⟶ CR			4	R1 = 1111: system mode
				10	1	(R1)r ÷ (M + (R2)) l/r → CR			4	
				11	1	(R1)r ÷ ((M)) l/r ⟶ CR			5	
				11	1	(R1)r ÷ ((M + (R2))) l/r → CR			5	
Compare characters	CCR	1	1101	01	1	(R1)r ÷ ((R2)) l/r ⟶ CR			3	R1 = 1111: system mode
register/register							4)			
Compare character	CCK	1	1101	01	1	(R1)r ÷ KLI → CR			3	R1 = 1111: system mode
with constant										
Compare words	CW	1	1101	10	0	(R1) ÷ (M) ⟶ CR			3	R1 = 1111: system mode
				10	0	(R1) ÷ (M + (R2)) → CR			3	
				11	0	(R1) ÷ ((M)) ⟶ CR			4	

				11	0	(R1) ÷ ((M + (R2))) → CR					4	
Compare words	CWR	1	1101	00	n.s.	(R1) ÷ (R2) → CR				1	1	R1 = 1111: system mode
register/register				01	0	(R1) ÷ ((R2)) → CR					2	
Compare word	CWK	1	1101	01	0	(R1) ÷ KL → CR					2	R1 = 1111: system mode
with constant												
Control Input/Output	CIO	0	1000	-	-	Start (bit 9=1) or stop (bit 9=0)			5)		3	bit 8 = 1
						any I/O operation						system mode
							quotient	remainder				
Divide	DV	1	1001	10	0	(A1, A2) / (M) →	A2	A1		10.5	3	
				10	0	(A1, A2) / (M + (R2)) →	A2	A1		10.5	3	
				11	0	(A1, A2) / ((M)) →	A2	A1		10.5	4	
				11	0	(A1, A2) / ((M + (R2))) →	A2	A1		10.5	4	
							quotient	remainder				
Divide by constant	DVK	1	1001	01	0	(A1, A2) / KL →	A2	A1		10.5	2	
							quotient	remainder	2)			
Divide registers/registers	DVR	1	1001	00	0	(A1, A2) / (R2) →	A2	A1		11.5	1	
				01	0	(A1, A2) / ((R2)) →	A2	A1		10.5	2	
Double Add	DA	1	1010	10	0	(M, M + 1) + (A1, A2) →	A1, A2			1	4	
				10	0	(M + (R2), M + (R2) + 1) + (A1, A2) → A1, A2				1	4	
				11	0	((M), (M) + 1) + (A1, A2) → A1, A2				1	5	
				11	0	((M + (R2)), ((M + (R2) + 1) + (A1, A2) → A1, A2				1	5	

name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
Double add registers/	DAR	1	1010	00	0	(R2, R2 + 1) + (A1, A2) ⟶ A1, A2		3	1	
registers				01	0	((R2), (R2 + 1)) + (A1, A2) → A1, A2		1	3	
Double add with constant	DAK	1	1010	01	0	KL + (A1, A2) → A1, A2		1	3	
Double subtract	DS	1	1011	10	0	(A1, A2) − (M, M + 1) ⟶ A1, A2		1	4	
				10	0	(A1, A2) − (M + (R2), M + (R2) + 1) → A1, A2		1	4	
				11	0	(A1, A2) − ((M), (M) + 1) ⟶ A1, A2	2)	1	5	
				11	0	(A1, A2) − ((M + (R2)), ((M + (R2)) + 1) → A1, A2		1	5	
Double subtract	DSR	1	1011	00	0	(A1, A2) − (R2, R2 + 1) ⟶ A1, A2		3	1	
registers/registers				01	0	(A1, A2) − ((R2), (R2 + 1)) → A1, A2		1	3	
Double subtract	DSK	1	1011	01	0	(A1, A2) − KL → A1, A2		1	3	
with constant										
Double left and	DLN	1	0111	-	n.s.		3)	$(1.5+\frac{n}{2})$	1	bits 8-10: 100
normalize shift										bits 11-14: R2;
										bit 15: n.s.
Double left arithmetic shift	DLA	0	0111	-	-		2)	$(1.5+\frac{n}{2})$		bits 8-10: 000
Double left circular shift	DLC	0	0111	-	-		1)	$(1.5+\frac{n}{2})$	1	bits 8-10: 110

Double left logical shift	DLL	0	0111	-	-			$(1.5+\frac{n}{2})$	1	bits 8-10: 010
Double right and normalize shift	DRN	0	0111	-	-		3)	$(1.5+\frac{n}{2})$	1	bits 8-10: 101 bits 11-14: R2; bit 15: n.s.
Double right arithmetic shift	DRA	0	0111	-	-			$(1.5+\frac{n}{2})$	1	bits 8-10: 001
Double right circular shift	DRC	0	0111	-	-		1)	$(1.5+\frac{n}{2})$	1	bits 8-10: 111
Double right logical shift	DRL	0	0111	-	-			$(1.5+\frac{n}{2})$	1	bits 8-10: 011
Enable interrupt	ENB	0	0101	-	-	machine status = 'permit interrupt'		1	1	bits 8-15: 01000000
Exchange characters register/register	ECR	1	1100	00	n.s.	(R2) → (R1)r; (R2)r → (R1)	3)	1	1	R1 = 1111: system mode
Exclusive OR	XR		0110	10	0	(R1) ∨ (M) → R1			3	R1 = 1111: system mode
				10	1	(R1) ∨ (M) → M			4	
				10	0	(R1) ∨ (M + (R2)) → R1			3	
				10	1	(R1) ∨ (M + (R2)) → M + (R2)	1)		4	
				11	0	(R1) ∨ ((M))→R1			4	
				11	1	(R1) ∨ ((M))→(M)			5	
				11	0	(R1) ∨ ((M + (R2)))→R1			4	
				11	1	(R1) ∨ ((M + (R2)))→(M + (R2))			5	

name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
Exclusive OR	XRR	1	0110	00	0	(R1) ∀ (R2) ⟶ R1		1	2	R1 = 1111: system mode
register/register				01	0	(R1) ∀ ((R2)) → R1			2	
				01	1	(R1) ∀ ((R2)) → R2	1)		3	
Exclusive OR	XRK	0	0110	-	-	$(R3)_{8-15}$ ∀ K → $R3_{8-15}$		1	1	short; R1 = 1111: system
with constant										mode
Exclusive OR	XRKL	1	0110	01	0	(R1) ∀ KL ⟶ R1			2	long; R1 = 1111: system
with constant										mode
Halt	HLT	0	0100	-	-	machine → 'halt' mode	3)	1	1	bits 8-15: 01111111;
										system mode
Increment Memory	IM	1	0010	10	1	(M) + 1 → M			4	bits 5-8: 0000
				10	1	(M + (R2)) + 1 → M			4	
				11	1	((M +)) + 1 ⟶ M	2)		5	
				11	1	((M) + (R2))) + 1 → (M + (R2))			5	
Increment memory/	IMR	1	0010	01	1	((R2)) + 1 ⟶ R2			3	bits 5-8: 0000
register										
Inhibit interrupt	INH	0	0100	-	-	machine status = 'prohibit all interrupts	3)	1	1	bits 8-15: 10111111
										system mode
Input to register	INR	0	1001	-	-	word/character from device → R3	5)		3	bit 8 = 0; system mode

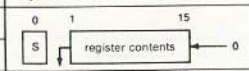
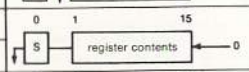
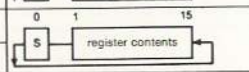
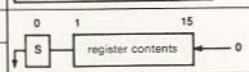
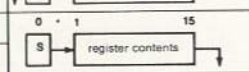
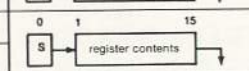
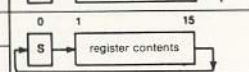
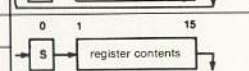
Link to monitor	LKM	0	0101	-	-	user mode ⟶ system mode		1	1	bits 8-15: 00000100
Load character	LC	1	1100	10	0	(M) l/r ⟶ R1r			3	R1 = 1111: system mode
				10	0	(M + (R2)) l/r → R1r	3)		3	R1 must be =/0
				11	0	((M)) l/r ⟶ R1			4	
				11	0	((M + (R2))) l/r → R1r			4	
Load character/constant	LCK	1	1100	01	0	KL l ⟶ R1r			2	R1 = 1111: system mode
Load character/register	LCR	1	1100	01	0	((R2)) l/r ⟶ R1r			2	R1 = 1111: system mode
Load constant	LDK	0	0000	-	-	K → $R3_{8-15}$, 0 → $R3_{0-7}$	3)	1	1	short; R1 = 1111: system mode
	LDKL	1	0000	01	0	KL ⟶ R1			2	long; R1 = 1111: system mode
Load register	LD	1	0000	10	0	(M) ⟶ R1			3	R1 = 111: system mode
				10	0	(M + (R2)) ⟶ R1			3	
				11	0	((M)) → R1			4	
				11	0	((M + (R2))) ⟶ R1	1)		4	
Load register/ register	LDR	1	0000	00	n.s.	(R2) ⟶ R1		1	1	R1 = 1111: system mode
				01	0	((R2)) → R1			2	
				01	0	(A15) + 2 → A15, ((A15)) → R1		1	2	system mode
Logical AND	AN	1	0100	10	0	(R1) ∧ (M) ⟶ R1			3	R1 = 1111: system mode
				10	1	(R1) ∧ (M) ⟶ M			4	
				10	0	(R1) ∧ (M + (R2)) → R1			3	
				10	1	(R1) ∧ (M + R2)) → M + (R2)			4	

name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
				11	0	(R1) ∧ ((M)) ⟶ R1			4	
				11	1	(R1) ∧ ((M)) ⟶ (M)			5	
				11	0	(R1) ∧ ((M + (R2))) ⟶ R1			4	
				11	1	(R1) ∧ ((M + (R2))) ⟶ (M + (R2))			5	
Logical AND	ANR	1	0100	00	0	(R1) ∧ (R2) ⟶ R1		1	1	R1 = 1111: system mode
register/register				01	0	(R1) ∧ ((R2)) ⟶ R1			2	
				01	1	(R1) ∧ ((R2)) ⟶ (R2)			3	
Logical AND	ANK	0	0100	-	-	$(R3)_{8-15}$ ∧ K ⟶ $R3_{8-15}$		1	1	short; R1 = 1111: system
with constant	ANKL	1	0100	01	0	(R1) ∧ KL ⟶ R1	1)		2	long; mode
Logical OR	OR	1	0101	10	0	(R1) ∨ (M) ⟶ R1			3	R1 = 1111: system mode
				10	1	(R1) ∨ (M) ⟶ M			4	R1 = 1111: system mode
				10	0	(R1) ∨ (M + (R2)) ⟶ R1			3	
				10	1	(R1) ∨ (M + (R2)) ⟶ M + (R2)			4	
				11	0	(R1) ∨ ((M)) ⟶ R1			4	
				11	1	(R1) ∨ ((M)) ⟶ (M)			5	
				11	0	(R1) ∨ ((M + (R2))) ⟶ R1			4	
				11	1	(R1) ∨ ((M + (R2))) ⟶ (M + (R2))			5	

Logical OR	ORR	1	0101	00	0	(R1) ∨ (R2) ⟶ R1		1	1	R1 = 1111: system mode
register/register				01	0	(R1) ∨ ((R2)) ⟶ R1			2	
				01	1	(R1) ∨ ((R2)) ⟶ (R2)			3	
Logical OR	ORK	0	0101	-	-	$(R3)_{8-15}$ ∨ K ⟶ $R3_{8-15}$		1	1	short; R1 = 1111: system
with constant										mode
Logical OR	ORKL	1	0101	01	0	(R1) ∨ KL ⟶ R1			2	long; R1 = 1111: system
with constant										mode
Multiple load	ML	1	0111	10	0	(M) ... (M + n) ⟶ A1 ... An			*	*3+(n-1)1 bits 5-8: n
				10	0	(M + (R2)) ... (M + (R2) + n) ⟶ A1 ... An				
				11	0	((M)) ... ((M) + n) ⟶ A1 ... An			**	**4+(n-1)1
				11	0	((M + (R2))) ... ((M + (R2)) + n) ⟶ A1 .. An				n = 1111: system mode
Multiple load/constant	MLK	1	0111	01	0	KL1, KL2, ... KLn ⟶ A1, A2, ... An			n+1	n = 1111: system mode
Multiple load/register	MLR	1	0111	01	0	((R2)) ⟶ A1; ((R2) + 2) ⟶ A2; ..... ;			n+1	n = 1111: system mode
						((R2) + 2n-2) ⟶ An	2)			bits 5-8; n.
					-	(A15) + 2n ⟶ A15; ((A15)) ⟶ A1;		1	n+1	system mode;
						((A15) - 2) ⟶ A2; .... ;				bits 5-8: n
						((A15) - 2n + 2) ⟶ An				
Multiple store	MS	1	0111	10	1	A1 ... An ⟶ M ... M + n		n	2+n	n = 1111: system mode;
				10	1	A1 ... An ⟶ M + (R2) ... M + (R2) + n	3)	n	2+n	bits 5-8: n
				11	1	A1 ... An ⟶ (M) ... (M) + n		n	2+n	
				11	1	A1 ... An ⟶ (M + (R2)) ... (M + (R2)) + n		n	2+n	

name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
Multiple store/register	MSR	1	0111	01	1	(A1) → (R2) ; (A2) → (R2) + 2 ;		n	n+1	n = 1111: system mode;
						....; (An) → (R2) + 2n - 2				bits 5-8: n
						(A1) → (A15) ; (A2) → (A15) - 2n + 2		n	n+1	pointer 128_{10}: stack overfl.
						.... ; (An, → (A15) - 2n + 2 ;				bits 5-8: n;
						(A15) - n → (A15)				system mode
Multiply	MU	1	1000	10	0	(A2) × (M) → A1, A2		8.5	3	
				10	0	(A2) × (M + (R2)) → A1, A2		8.5	3	
				11	0	(A2) × ((M)) → A1, A2		8.5	4	
				11	0	(A2) × ((M + (R2))) → A1, A2	2)	8.5	4	
Multiply registers/registers	MUR	1	1000	00	0	(A2) × (R2) → A1, A2		9.5	1	
				01	0	(A2) × ((R2)) → A1, A2		8.5	2	
Multiply with constant	MUK	1	1000	01	0	(A2) × KL → A1, A2		8.5	2	
One's complement	C1	1	1111	10	0	$(\overline{M})$ → R1			3	R1 = 1111: system mode
				10	1	$(\overline{M})$ → M			4	
				10	0	$\overline{(M + (R2))}$ → R1	1)		3	
				10	1	$\overline{(M + (R2))}$ → M + (R2)			4	
				11	0	$\overline{((M))}$ → R1			4	
				11	1	$\overline{((M))}$ → (M)			5	

				11	0	$\overline{((M + (R2)))} \rightarrow R1$			4	
				11	1	$\overline{((M + (R2)))} \rightarrow (M + (R2))$			5	
One's Complement	C1R	1	1111	00	n.s.	$\overline{(R2)} \rightarrow R1$		1	1	when l/s bit = 0,
register/register				01	0	$\overline{((R2))} \rightarrow R1$			2	R1 must be ≠ 0;
				01	1	$\overline{((R2))} \rightarrow (R2)$			3	R1 = 1111: system mode
Output from register	OTR	0	1000	-	-	word/character from R3 → device	5)		3	bit 8 = 0; system mode
Read channel address	RCA	0	1001	-	-	channel number causing MIDB interrupt			3	bits 8-15: 01000000;
						request → R3				system mode
Read interrupt lines	RIL	0	1001	-	-	State interrupt lines → R3		2	1	bits 8-15: 00000000;
										system mode
Relative backwards	RB	0	1011	-	0	(P) → displ. → P (branch effective)		1	1	bits 5-7: condition
conditional branch						(P) → P (no branch)		1	1	bits 8-14: displ.;
							3)			bit 15: n.s.
Relative forward	RF	0	1010	-	0	(P) + displ. → P (branch effective)		1	1	bits 5-7: condition
conditional branch					-	(P) → P (no branch)		1	1	bits 8-14: displ.;
										bit 15: n.s.
Reset internal	RIT	0	0100	-	1	to clear internal interrupt bits		1	1	bits 8, 9, 15: 1,
interrupt										bits 10-14: D.A.; system
										mode
Return from Function	RTN	1	1110	01	0	(R2) + 4 → R2 ; ((R2)) → P ;	*	1	3	*reloaded from stack;
						((R2) −2) → PSW				R2 = 1111: system mode
Send status	SST	0	1001	-	-	status character/word from device → R3	5)		3	bits 8-9:11; system mode

name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
Set mode	SMD	0	0101	-	-	system mode → user mode	3)	1	1	bits 8-15: 00000001
Single left and normalize shift	SLN	0	0111	-	-			$(1.5+\frac{n}{2})$	1	bits 8-10: 100; bits 11-14: R2; bit 15: n.s.
Single left arithmetic shift	SLA	0	0111	-	-		2)	$(1.5+\frac{n}{2})$	1	bits 8-10: 000
Single left circular shift	SLC	0	0111	-	-				*	bits 8-10: 110 *$(2+\frac{1}{2}n+\frac{1}{2})$
Single left logical shift	SLL	0	0111	-	-		1)	$(1.5+\frac{n}{2})$	1	bits 8-10: 010
Single right and normalize shift	SRN	0	0111	-	-		3)	$(1.5+\frac{n}{2})$	1	bits 8-10: 101 bits 11-14: R2; bit 15: n.s.
Single right arithmetic shift	SRA	0	0111	-	-			$(1.5+\frac{n}{2})$	1	bits 8-10: 001
Single right circular shift	SRC	0	0111	-	-		1)	$(1.5+\frac{n}{2})$	1	bits 8-10: 111
Single right logical shift	SRL	0	0111	-	-			$(1.5+\frac{n}{2})$	1	bits 8-10: 011

Store character	SC	1	1100	10	1	(R1)r → (M) r/l			4	R1 = 1111: system mode
				10	1	(R1)r → (M + (R2)) r/l			4	R1 must be ≠ 0
				11	1	(R1)r → ((M)) r/l			5	
				11	1	(R1)r → ((M + (R2))) r/l			5	
Store character/register	SCR	1	1100	01	1	(R1)r → (R2) r/l			3	R1 = 1111: system mode
Store register	ST	1	0000	10	1	(R1) → M	3)	1	3	R1 = 1111: system mode
				10	1	(R1) → M + (R2)		1	3	
				11	1	(R1) → ((M))		1	4	
				11	1	(R1) → ((M + (R2)))		1	4	
Store register/register	STR	1	0000	01	1	(R1) → (R2)		1	2	R1 = 1111: system mode
				01	1	(R1) → (A15) ; (A15) - 2 → A15		1	2	if pointer 128_{10}: stack
										overflow
Subtract constant	SUK	0	0011	-	-	(R3) - K → R3		1	1	short; R1 = 1111: system
										mode
Subtract constant	SUKL	1	0011	01	0	(R1) - KL → R1			2	long; R1 = 1111: system
										mode
Subtract register/register	SUR	1	0011	00	n.s.	(R1) - (R2) → R1	2)	1	1	when l/s bit = 1, R1 must
				01	0	(R1) - ((R2)) → R1			2	be ≠ 0; R1 = 1111:
				01	1	(R1) - ((R2)) → (R2)			3	system mode
Subtract word	SU	1	0011	10	0	(R1) - (M) → R1			3	R1 = 1111: system mode
				10	1	(R1) - (M) → M			4	when l/s bit = 1,
				10	0	(R1) - (M + (R2)) → R1			3	R3 must be ≠ 0

name (in alphabetical order)	mnemonic	for-mat	OP-code	mode	L/S (0/1) bit	function	condition register	execution time in cycles 6) l.c.	execution time in cycles 6) m.c.	remarks
				10	1	(R1) - (M + (R2)) → M + (R2)			4	
				11	0	(R1)-((M)) → R1			4	
				11	1	(R1) - ((M)) → (M)			5	
				11	0	(R1) - ((M + (R2))) → R1			4	
				11	1	(R1 - ((M + (R2))) → (M + (R2))			5	
Test mask	TM	1	0100	00	1	[(R1) ∧ (R2)] ÷ 0 → CR	1)	1	1	R1 = 1111: system mode
Test not mask	TNM	1	0110	00	1	[(R1) ∀ (R2)] ÷ 0 → CR		1	1	R1 = 1111: system mode
Test status	TST	0	1001	-	-	test DCU 'ready state'	5)		3	bits 8-9: 10; system mode
Two's complement	C2	1	0011	10	1	0 - (M) → M			4	bits 5-8: 0000
				10	1	0 - (M + (R2)) → M + (R2)	2)		4	
				11	1	0 - ((M)) → (M)			5	
				11	1	0 - ((M + (R2))) → (M + (R2))			5	
Two's complement/register	C2R	1	0011	01	1	0 - ((R2)) → (R2)			3	bits 5-8: 0000
Write interrupt mask	WIM	0	1000	-	-	(R3) → mask register		2	1	bits 8-15: 00000000
							3)			system mode
Write mask protection	WMP	0	1000	-	-	(R3) → M.P. key register		2	1	bits 8-15; 01000000;
										system mode
Write mask protection no. 2	WM2	0	1000	-	-	(R3) → M.P. key register		2	1	bits 8-15: 11000000; s.m.

NOTES FOR INSTRUCTION SET

Condition register

1) CR=0 if result = 0
   1 if result > 0
   2 if result < 0

2) CR=0 if result = 0
   1 if result > 0
   2 if result < 0
   3 if overflow

3) CR unchanged

4) CR=0 if a = b
   1 if a > b
   2 if a < b

5) CR=0 if command accepted
   1 if command not accepted
   3 if device address unknown

6) Execution times in microseconds

	P855M	P860M
mc (memory cycle)	1.2	0.84
lc (logic cycle)	0.72	0.72

INTERFACES

It is very ;important that a computer can be easily integrated into a user's system. Therefore, in the design of the P855M and P860M, complicated device control unit busing, unwieldy device addressing and restricted interfacing have been avoided. Instead, the clean, straightforward design of the CPU is easy to understand.
The simple I/O bus, used for programmed and for multiplex data transfers, is not subject to any critical timing or addressing restrictions — a great advantage when setting up a system either with standard or self-designed device control units.
Furthermore, memory increment data break, with direct access to the address lines, is available for quantitative analysis applications, and also a direct memory access channel for easy connection to high-speed devices.

Data are transferred into or out of the central processor by a standard I/O Bus which consists of 32 data lines (16 in, 16 out) and the signal and address lines necessary to effect data transfers. Transfers along the I/O bus are controlled by the Programmed Channel. The Programmed Channel is used for low speed applications such as operator's typewriter or punched tape or card devices. On option, Multiplex channel and Direct Memory Access can be connected for fast data transfers. The same functions used to start and stop data transfers on programmed channel, are also used on multiplex and direct memory access.
Figure 3 shows the I/O bus and its signal lines.

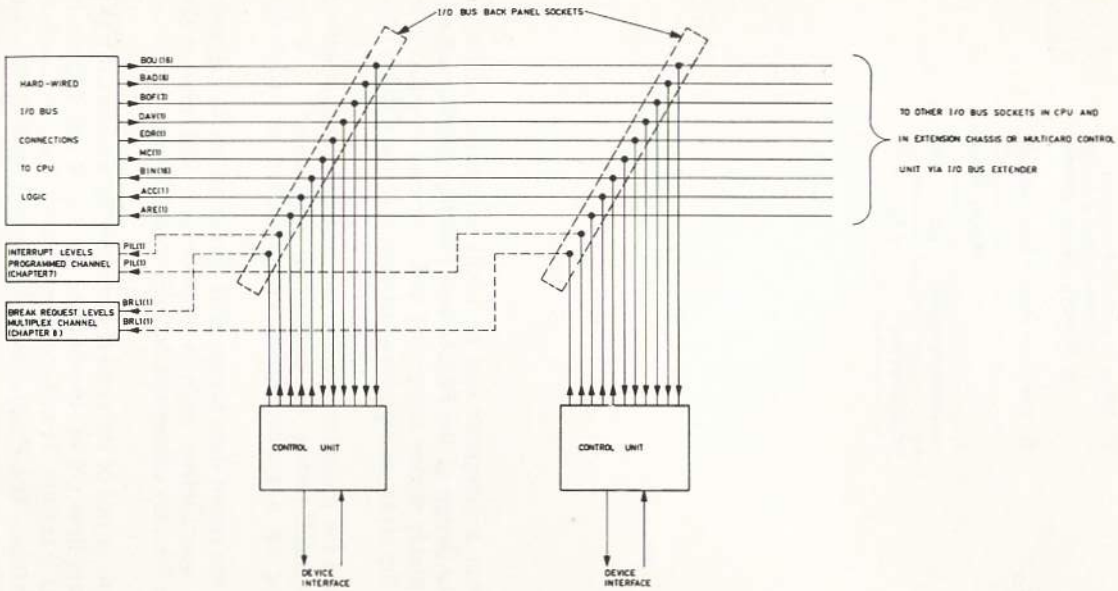


Figure 3 I/O Bus

CPU→CU	BOU:	Bus output: 16 lines
	BAD:	Bus address: 6 lines
	BOF:	Bus function: 3 lines; indicate the type of I/O operation
	DAV:	Device Address Validation
	MC:	Master clear (reset from control panel or power supply)
	EOR:	End-Of-Range signal (multiplex channel)
CU→CPU	BIN:	Bus input: 16 lines
	ACC:	Accept line; active only if BOF command has been recognized
	ARE:	Address recognized; response to DAV signal
	PIL:	Interrupt request line
	BR:	Break request line (multiplex channel)

Programmed Channel

I/O by programmed channel is controlled by a sequence of instructions regulating the the data transfer. Data are transferred one word or one character at a time and for each transfer an I/O instruction is necessary. The data transfers take place within or during program interrupts.

In addition to data transfer of single words or characters, the programmed channel functions used to start or stop a data transfer are also used on the multiplex or direct memory access channel.

The following illustration shows the relationship between an I/O instruction and a data transfer.

The abbreviations used in this drawing refer to the signal lines of the I/O bus, as described in Figure 3.

The instruction consists of an operation code, function bits (X,T,F), a register specification and a device address (DA). CR indicates the condition register.

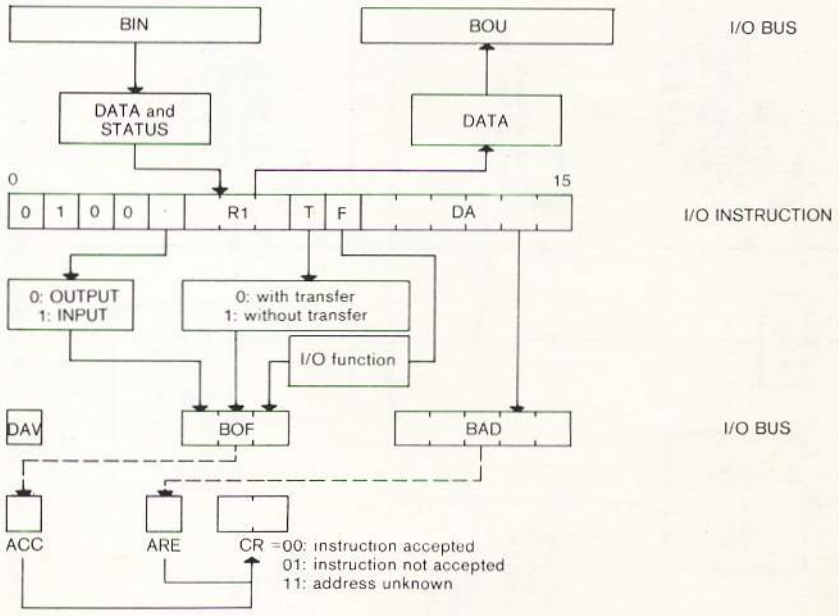


Figure 4 Relationship between instruction and data flow.

Figure 4a shows diagramatically the data flow involved during the two basic I/O instructions with reference to the flow diagram, figure 2.

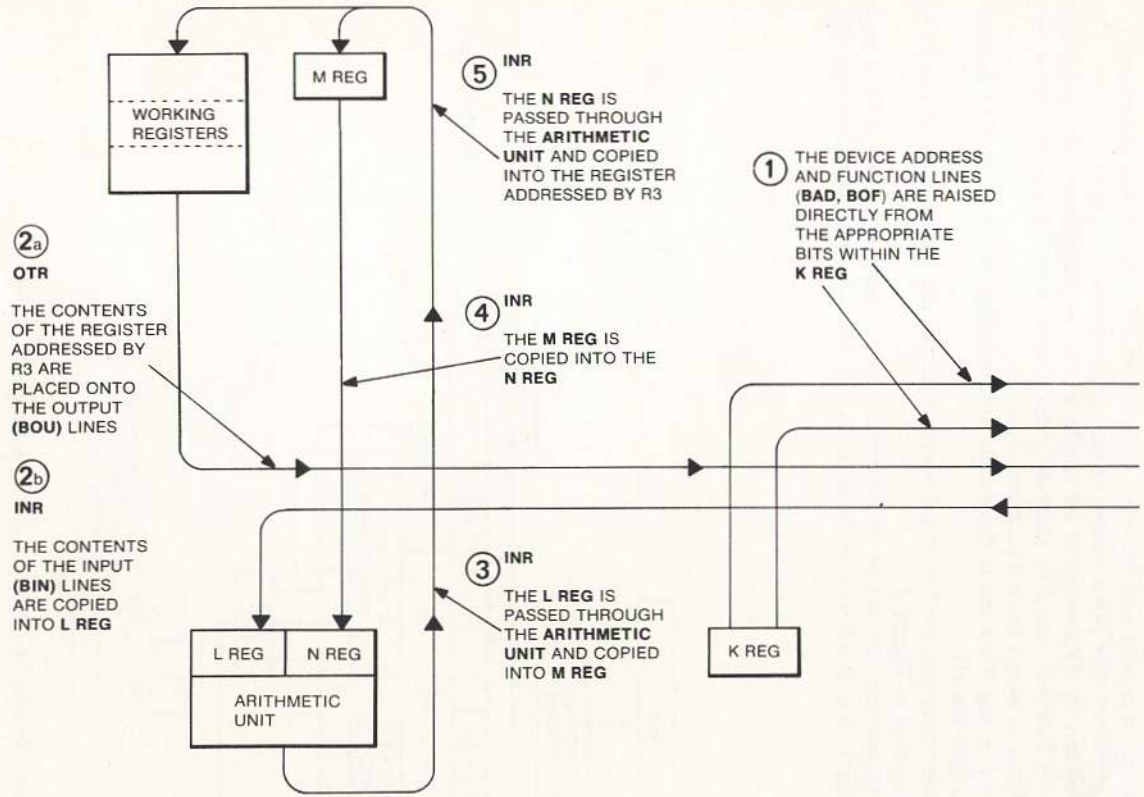


Fig. 4a Input/Output via a register using INR/OTR instructions

Multiplex Channel

With the multiplex channel data transfers are programmed in blocks, instead of one word or character at a time.

The multiplex channel has additional logic over the Programmed Channel, but still needs the central processor for arithmetic functions. The information needed by the multiplex channel is stored in multiplex control words in memory, two for each control unit connected to the multiplex, containing the length of the data transfer and the address of the buffer to or from which the data are transferred.

The transfer takes place via the I/O bus, each character or word transfer being initiated by a break request from the device control unit. Up to 15 device control units can be connected to the Multiplex Channel, so the break request must be checked for priority over other break requests. When a request is accepted, it breaks into the running program either between or during an instruction, depending on whether the instruction takes longer than 3 cycles. When the transfer is completed, the program is restarted.

Thus, a faster transfer rate is ensured, for the block length, location of the data in memory and whether transfer is to take place on a word- or character-basis are the only factors which need to be initialized by software.

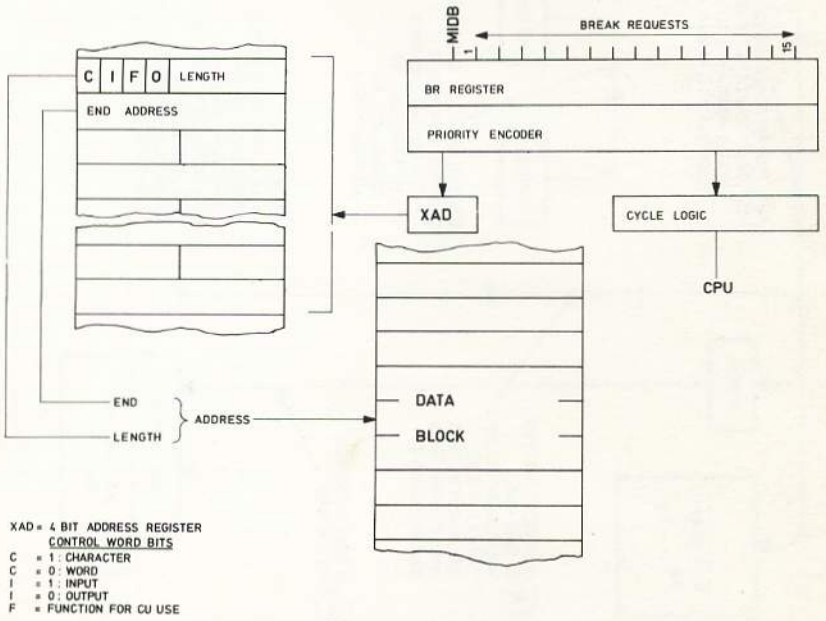


Fig. 5 Multiplex System

Figures 5a to 5e show diagramatically the data flow involved during the setting up and actioning of a multiplex transfer.

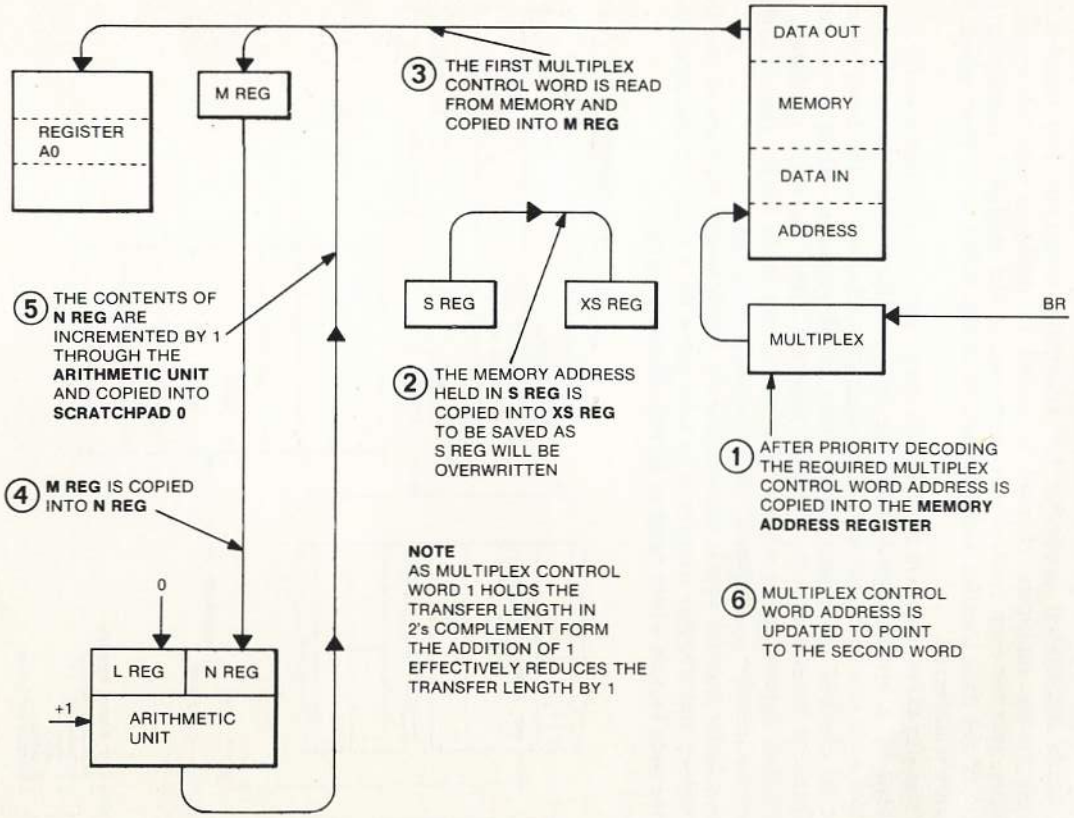


Fig. 5a Accessing multiplex control word 1

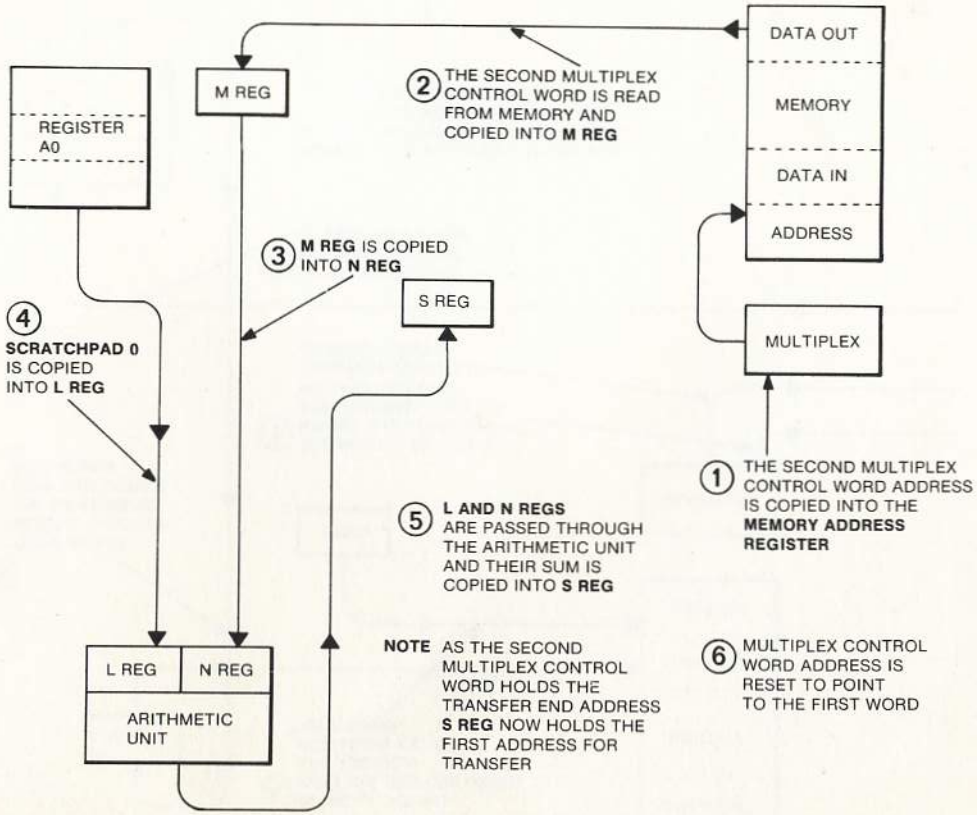


Fig. 5b Accessing multiplex control word 2 and setting the transfer address

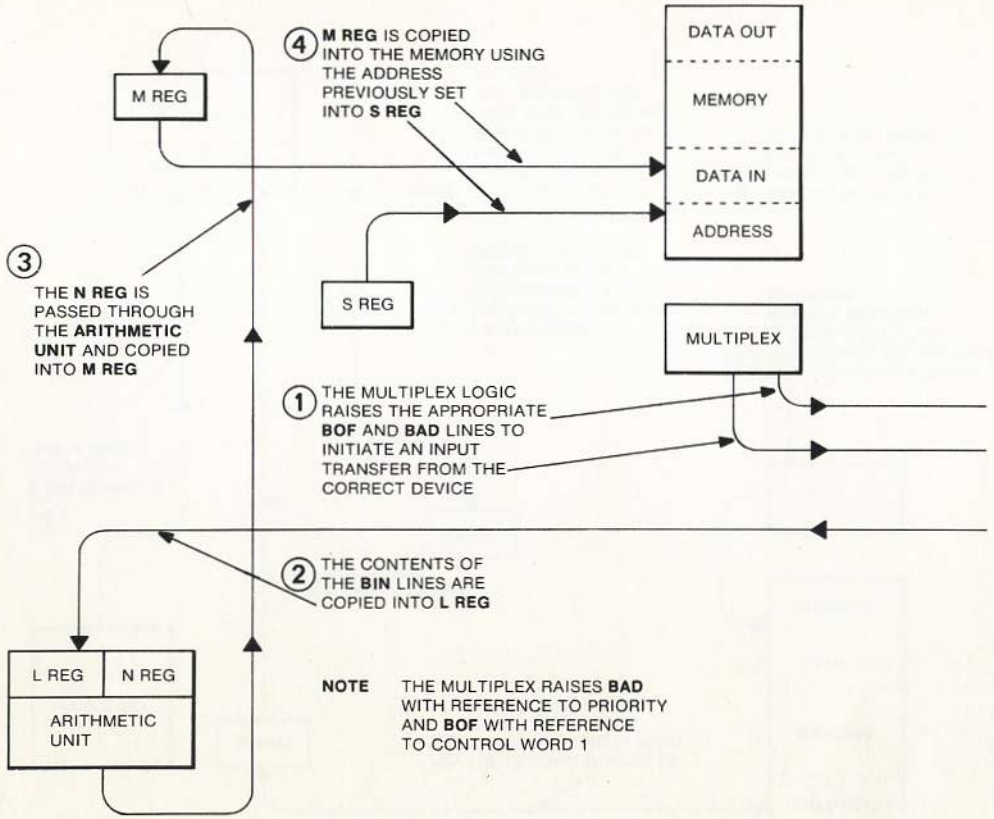


Fig. 5c Multiplex transfer (input)

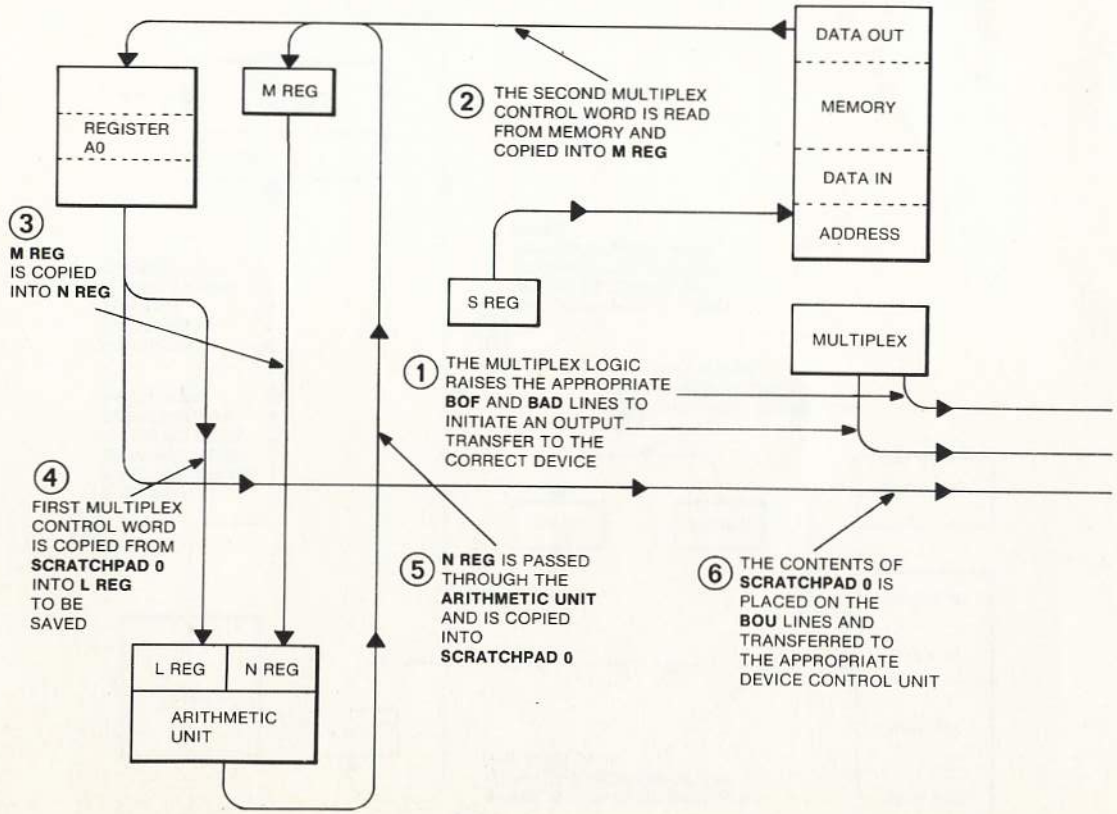


Fig. 5d Multiplex transfer (output)

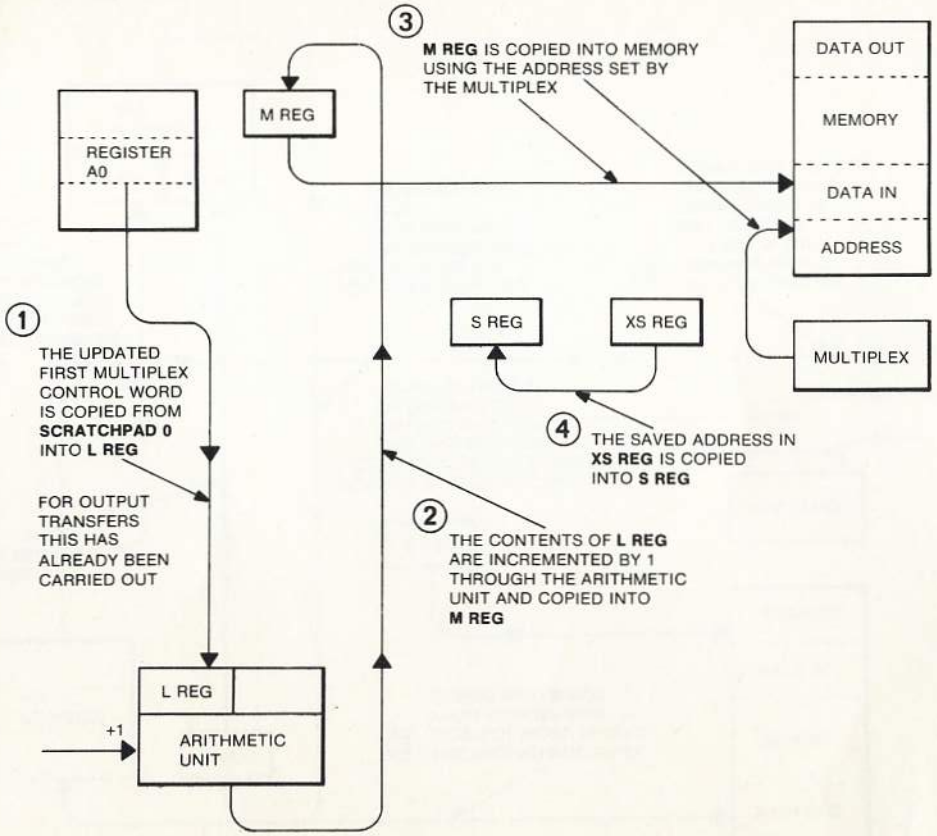


Fig. 5e Restoring the updated multiplex control word 1

Direct Memory Access

This channel enables the transfer of a block of words to be carried out without the use of the CPU once the transfer has been initiated, the data path used being a direct connection between the memory and the device control unit. The device control units are connected to a special I/O Bus (I/O Bus S) which can be switched within the DMA to either the normal I/O Bus or directly to the memory. The DMA also contains its own memory addressing and data control logic. Control of the DMA is initially carried out via the normal I/O Bus and this connection is never switched. The DMA is first ACTIVATED and then sends the transfer details in the form of 2 words defining the block START ADDRESS and LENGTH. A CIO START command is then sent to the device control unit and the transfer commences. Once started the control of the DMA transfer is by BREAK signals from the device control unit, the I/O connection between the CPU and the device being broken and replaced by the direct connection between memory and the device control unit on receipt of the first BREAK signal. During a transfer the DMA requests memory cycles as required and these requests take priority over the normal CPU accessing of memory, the CPU being stopped whilst the single memory cycle required to transfer 1 word is in progress. On completion of a block transfer the DMA must be deactivated to reconnect the device control unit to the normal I/O Bus and thus allow STATUS commands to be sent to the control unit.

Normally only high-speed peripheral devices will be connected to the DMA channel and where these contain their own control units, separate from the CPU, extender line drivers and receivers will be required to be used in conjunction with any connecting cables. Figure 6 shows the connection of a device control unit to the DMA and figure 6a to 6e show diagramatically the data flow paths involved during the setting up and actioning of a DMA transfer.

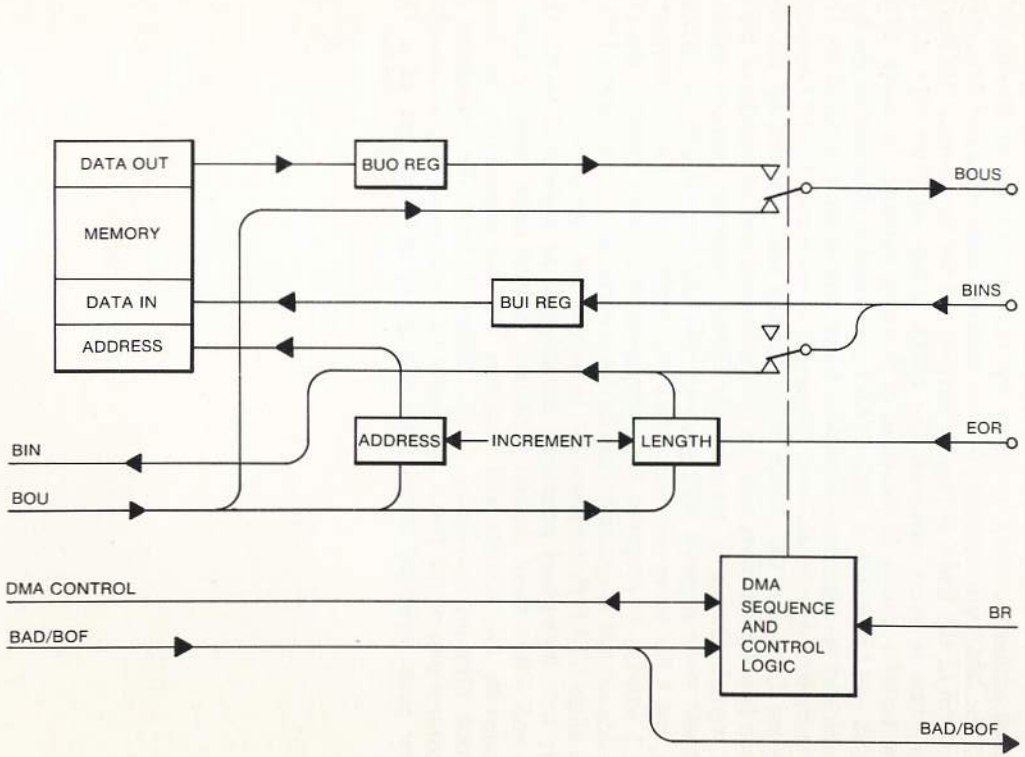


Fig. 6a Overall Data Paths of the DMA

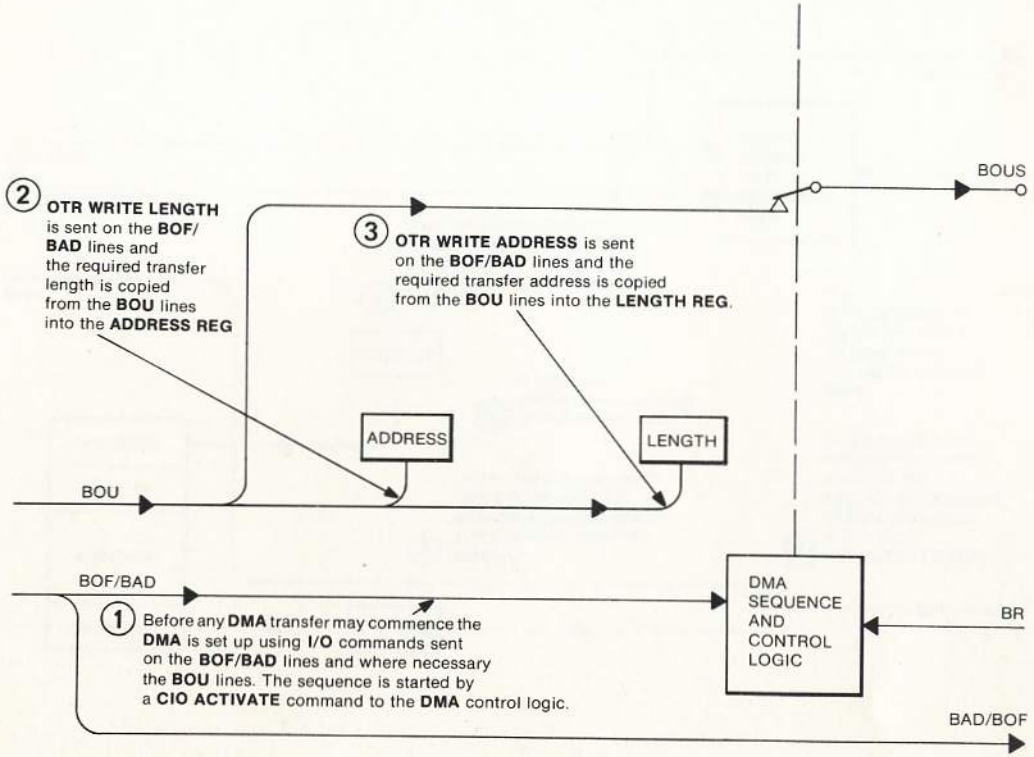


Fig. 6b DMA activation, set length, and set address

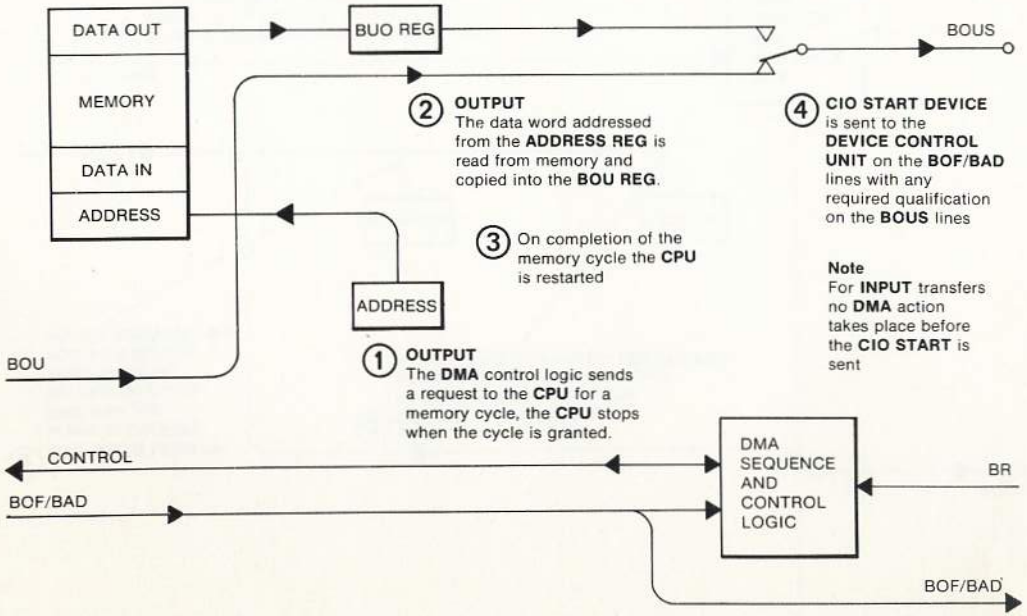


Fig. 6c DMA initial output action and CIO start device

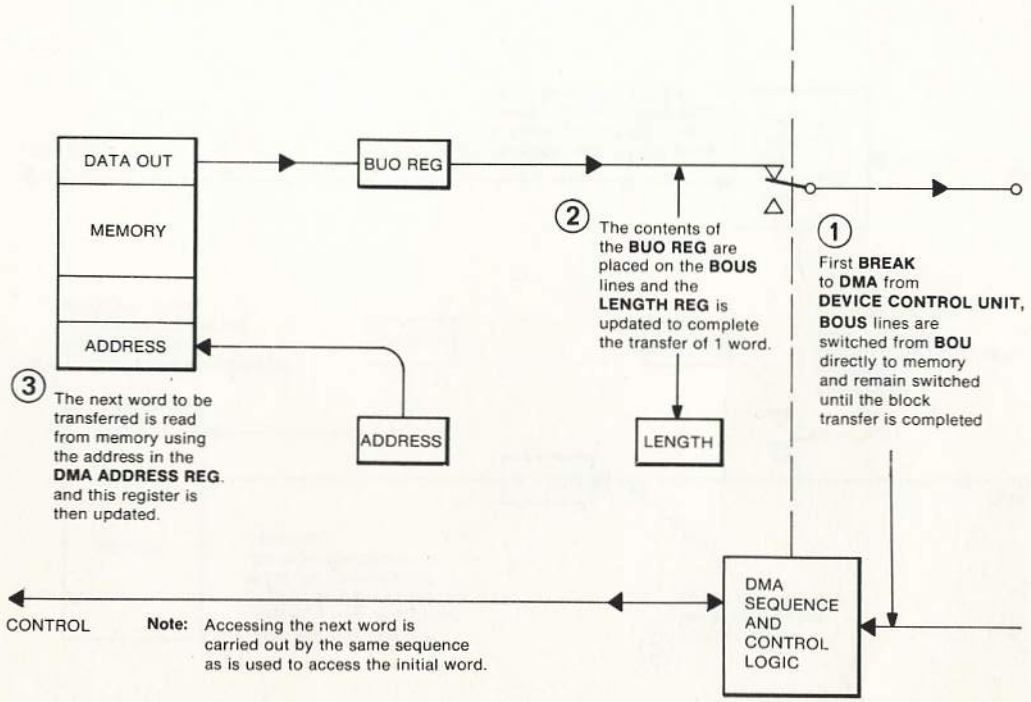


Fig. 6d DMA output action on receipt of 'break'

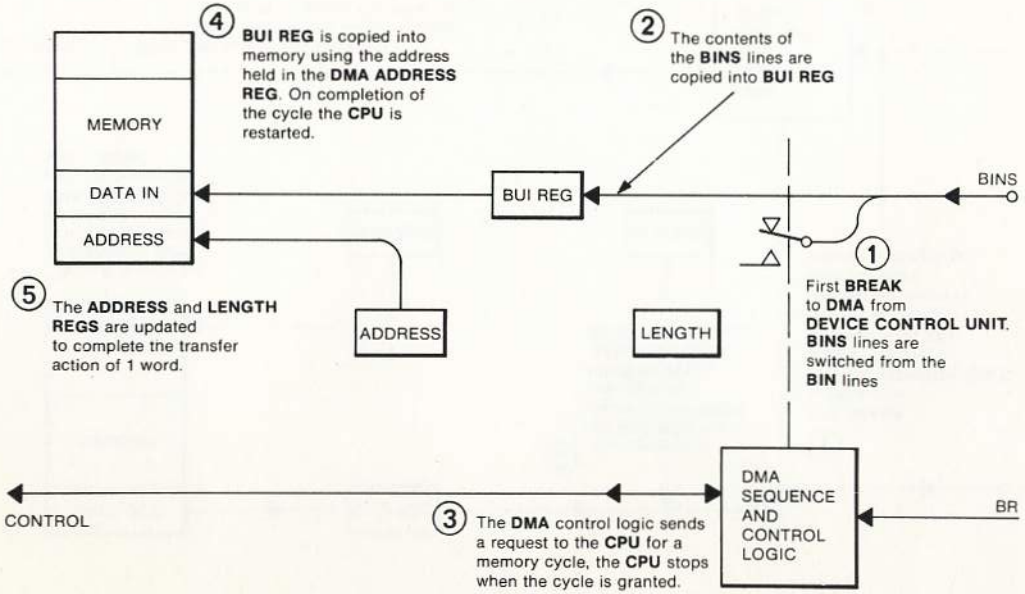


Fig. 6e DMA input transfer on receipt of 'break'

Maximum Transfer Rates

	Programmed Channel	Multiplex	DMA
P855M	28kc/s	160kc/s	810kc/s
P860M	40kc/s	240kc/s	1.2Mc/s

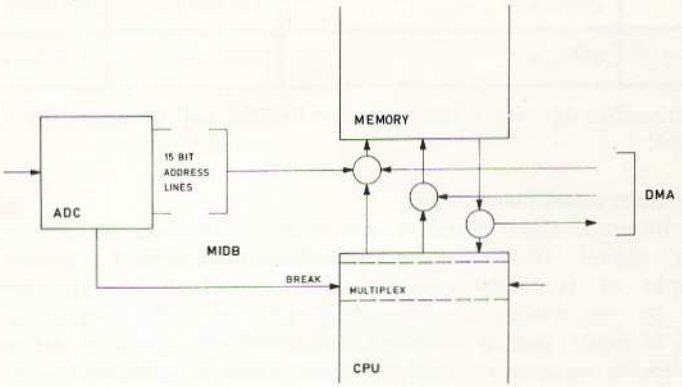
From November 1st, 1973, the rates for P855M will be upgraded to those of the P860M.

Memory Increment Data Break

Memory Increment Data Break is used in quantitive analysis applications, such as with digital thermometers, scintillation counters, quality control, radiography or statistical analysis in cases where average measurement readings are important: in an assigned area of CPU memory a statistical summary of digital data is compiled by periodically sampling the output from the monitoring equipment. Each digitized value is quantified as a particular memory address and during the sampling process the memory location representing the measured value is addressed and its contents incremented by one. Thus each location records the number of times the value represented by its address has been measured.

The MIDB channel has no data transfer lines, but addresses the memory directly. The addressed location is incremented by the multiplex logic each time a break signal is received from the connected device. This break line is connected to the highest priority break signal input of the multiplex break register.

A typical application is shown in the figure below where analog values from sensing and scanning equipment are fed into an analog-to-digital converter (ADC) and the result is applied to the computer memory address lines. At fixed intervals a break signal is given by the ADC unit and on each break the memory location addressed by the ADC is incremented by one. The graph shows quantities represented by memory locations plotted against the time that each value is obtained. Sensing, scanning and analog-to-digital equipment used with MIDB is not included in the P855/P860M systems, but the Philips MIOS system is available for this type of application (see Chapter 3).



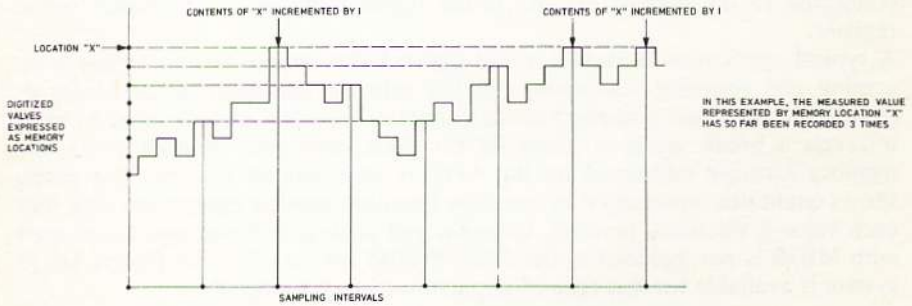
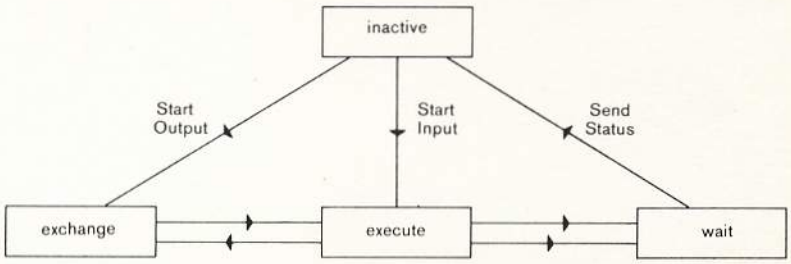


Fig. 7 Memory Increment Data Break

Control Units
All peripherals and external devices are connected to the I/O Channels via device control units. A control unit is principally based on four functional parts:

- an address decoder, to determine from the address lines of the I/O Channel whether it is being addressed.
- a function decoder to find out from the function lines of the I/O Channel which function it is to perform.
- a sequence control unit to provide for start/stop and other control signals after the address has been recognized and the function decoded. In accordance with this, a control unit may be in inactive, execute, exchange or wait state, depending on the input/output instructions given:



- a data buffer, for intermediate data storage, to effect a more efficient utilization of machine time.

The following table shows the number of devices which can be attached to a control unit; the last three columns indicate with an X the channels to which the control units can be connected.
The indicated connections are only the standard connections.

Control Unit for:	Devices per C.U.	Progr. Channel	Multiplex Channel	DMA
Operator's Typewriter	1	X		
Punched Tape Reader	1	X		
Tape Punch	1	X		
Line Printer	1		X	
Magnetic Tape	4		X	
Cassette Tape	3		X	
Moving Head Disc	2		X	X
Fixed Head Disc	4		X	X
Plotter	1		X	
Card Reader	1		X	

INTERRUPT AND STACKING SYSTEM

The interrupt system is used for all peripheral operations and for handling internally generated interrupts. The system will handle up to 63 interrupt request lines. Interrupts are handled according to their priority, which is established by pre-wiring; the highest priority interrupt request is accepted and compared with the priority level of the running program. If the priority level of the interrupt is higher than that of the running program, the program is interrupted, if it is running in enable mode, and the P-register contents (the address of the next program instruction) and the program status word (containing the priority level and other information) are stored in a memory stack. A new program is then started by the interrupt and this program runs until stopped by a higher priority interrupt or until it is completed.

Figure 8 shows the interrupt system in simplified form. The interrupt request lines are wired up to 48 memory locations forming the interrupt register. The priority level of an interrupt is determined by the memory location to which the interrupt line is wired. One of the lines is a maskable common interrupt line which is able to accept 15 interrupt signals from different sources. This line has a single priority level but individual priorities can be handled by software, with an interrupt mask register: a set bit in this mask inhibits the associated interrupt line. The mask register is loaded by program instruction, so that the acceptable interrupts can be changed at any time.
The state of the interrupt lines is sampled with each cycle, copied onto the interrupt register and transferred to the priority encoder. There it is converted to a binary number, transferred to the comparator and compared with the priority level (PL) register. If a new interrupt has become active, of a higher priority level than the current one, a program interrupt is given.

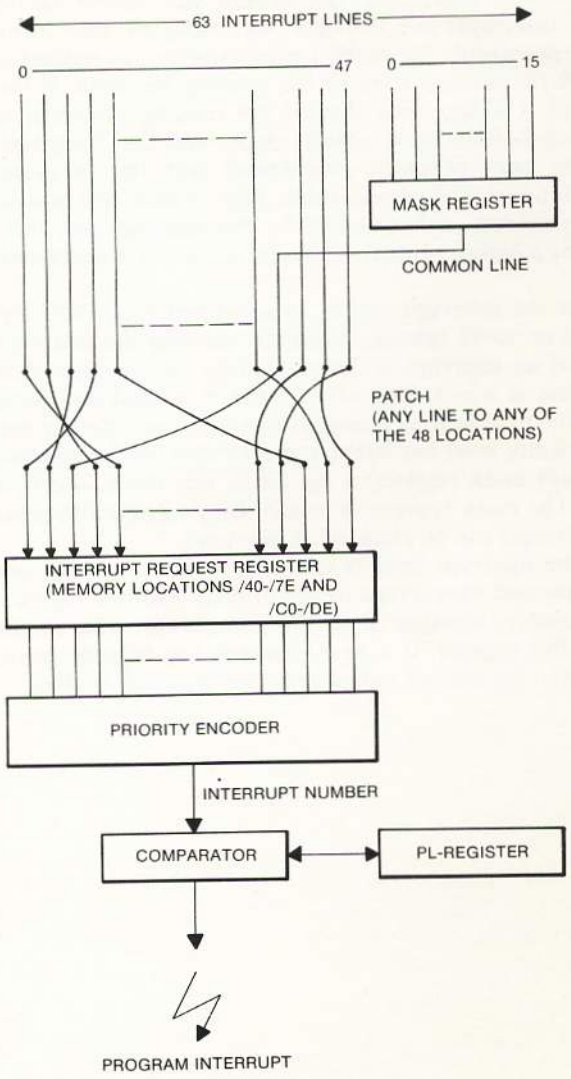
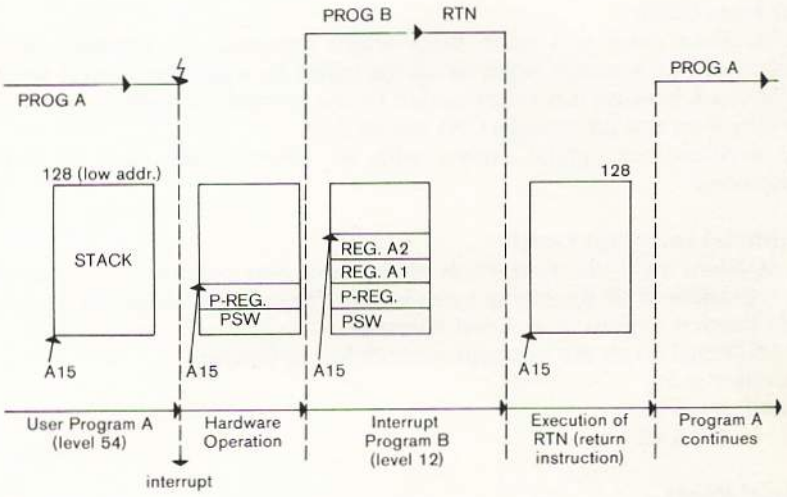


Fig. 8 Interrupt system

A hardware routine is started by the program interrupt signal from the comparator after the current program instruction is completed. During this routine the P-register contents and the program status word are stacked and the stack pointer (scratch pad register 15) is decremented to point to the next free location in the stack. The 6-bit number from the priority encoder, which is the priority level of the new program started by the interrupt, is loaded into the priority register. This number is also used to select a location in memory which contains the starting address of the new program.
The illustration below shows, in simplified form, the operation of the interrupt and stacking system:



The new program started by the interrupt will normally contain routines to save the contents of registers for the old program and may also include an instruction to enable the interrupt system to accept new interrupt requests. These routines are included in the executive monitors provided with the P855M/P860M systems.
Return to the interrupted program is initiated by a return instruction (RTN) to take the program status word and program address from the stack. The priority level of the program, contained in the status word, is loaded into the priority register and the program address is loaded into the P-register. The program then resumes and continues until completed or until interrupted by a new interrupt signal.

ADDITIONAL FEATURES

Power Failure/Automatic Restart

This feature provides the means to detect a power failure in time and automatically restart a program, without loss of information.
If the a.c. power fails completely or drops below the minimum level for error-free operation for longer than 20 milliseconds, an interrupt is generated 5 milliseconds before the d.c. operating voltages fall below normal. During this time all information relevant to the current program can be stored by software. Provided that the control panel key is in the 'LOCK' position, the program is restarted automatically when the power is restored, i.e. all the hardware is reset, the stored information is retrieved and the interrupted program is resumed.

Real Time Clock

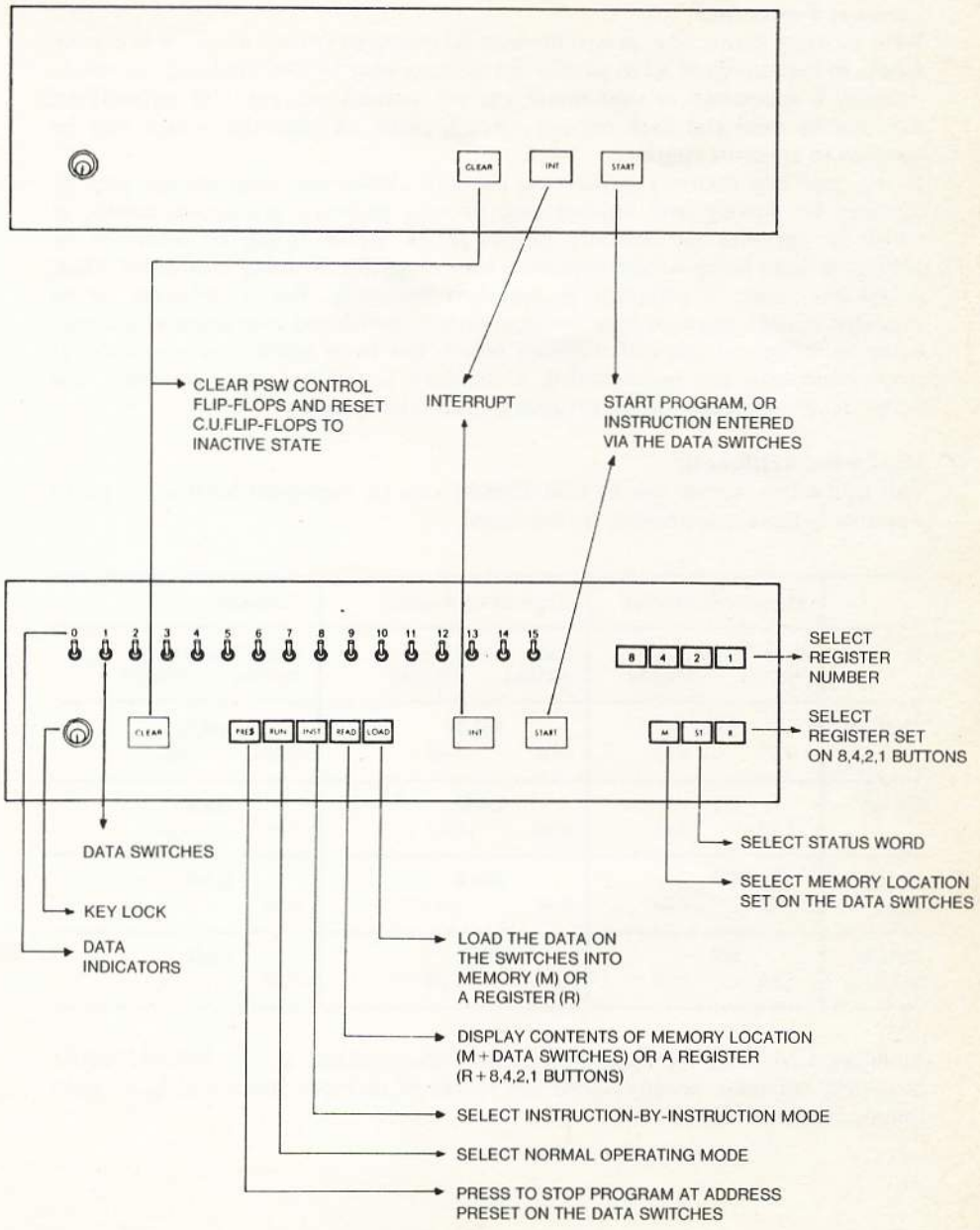
The standard clock is a pulse timer which generates an interrupt every 20 milliseconds. Its accuracy depends on the mains to which the central processor is connected, because it is synchronized by the normal a.c. voltage.
It is turned on and off through CIO instructions.
Also available are crystal timers, with an interval time of 1, 2, 5 or 10 milliseconds.

Additional Interrupt Levels

The standard eight interrupt levels of the basic processor can be increased up to a maximum of 48 by adding extra levels. These extra levels can be used for either internal options or external interrupts.
The additional levels are available in modules, as follows:
- levels 9 to 18
- levels 19 to 28
- levels 29 to 48.

Control Panel

The mini-panel is the standard control panel for all models, the full control panel is an optional extra.
The full panel contains switches and indicators which allow the operator or programmer to manually load or display the contents of registers or memory locations. It is possible to interrupt the running program and to make any necessary alterations to it.
A keylock is fitted which in one position permits all the switches on the panel to function, and in the other positions only the interrupt button.



Mini-panel and full control panel

Memory Protection

With memory protection, programs may be executed in user mode or in system mode. In system mode all available instructions may be executed and the whole memory is accessible. In user mode, certain instructions, e.g. I/O instructions, may not be used and their occurrence will cause an interrupt, which may be coupled to an abort routine.

If the user has memory protection, he can define the unprotected part of memory by writing and loading one or two memory protection masks, in which he specifies the memory blocks of 1k words which he wants to be protected from being overwritten by a user program. Reading is allowed. Thus, in system mode, a program is executed normally. For a program to be executed in user mode, it may not contain any privileged instructions, nor may it try to modify an area of memory which has been defined as protected. If these conditions are not satisfied, a memory protection error interrupt will occur; no modification of the protected area takes place.

Hardware Arithmetic

The instruction set of the P855M/P860M can be expanded with a set of 12 optional hardware arithmetic instructions:

	Memory Reference		Register-to-Register		Constant	
*)	Exec. Times (μs.) P855M	P860M	Exec. Times (μs.) P855M	P860M	Exec. Times (μs.) P855M	P860M
Multiply	**MU** 9.72	8.37	**MUR** 8.04	7.68	**MUK** 8.52	7.80
Divide	**DV** 11.12	9.81	**DVR** 9.48	9.12	**DVK** 9.96	9.24
Double Add	**DA** 5.50	4.08	**DAR** 3.40	3.00	**DAK** 4.30	3.24
Double Subtract	**DS** 5.50	4.08	**DSR** 3.40	3.00	**DSK** 4.30	3.24

Simulation routines for these instructions are available in the P855M/P860M monitors, but these occupy about 500 words of memory space and have much longer execution times.

*) From November 1st, 1973, the execution times for P855M will be equal to those for P860M.

Cabinetry

Most equipment, including the central processor, can be housed in standard 19-inch racks, 36 standard units (1 unit = 1.75 in. = 44.45 mm) high. Available are basic cabinets and extension cabinets (same as basic cabinet, but without side panels).

These can be complemented with air ventilation units, 10 Amp. power distribution panels, telescopic slides for CPU and peripherals and front panels of 1, 3, 5 or 7 standard units to fill up any space between the devices in the rack.

See also the photograph on page IV.

DIOS (Digital Input/Output System)

DIOS is a general-purpose system which acts as a TTL (transistor-to-transistor logic) interface between a P855M or P860M and any external equipment. The principal function of the DIOS is to control the exchange of 16-bit data words (in both directions) via the programmed channel. Three versions of the basic DIOS are available, consisting of logic circuitry, each handling four 16-bit words by means of standard I/O instructions. These basic systems are:

- DIC: attached to the I/O bus, controls four 16-bit gated input words;
- DOC: attached to the I/O bus, handles four 16-bit buffered output words;
- DIOC: attached to the I/O bus, controls two gated input and two buffered output words.

The basic system is intended for equipment whose voltage levels and currents are similar to those used by the processor interface. It may be extended with some additional features:

- for input lines, a sixteen-bit buffer register and change-of-state detection;
- for both input and output lines, level adaptation and galvanic isolation circuitry.

The input buffer consists of two 16-bit registers plus one flip-flop per word for the external CALL line, which are loaded by external equipment. The output of the 16-bit buffer is connected to the corresponding input lines of the basic DIOS. The change-of-state detection provides for an external exchange state.
Level adaptation allows the system to exchange TTL signals at the logical interface level of the particular external devices. Electrical isolation is provided where the ground voltage of the external device differs from that of the CPU, or where the power supply or transmission lines are noised at high level. Two methods of operation are possible:

- software control: data transfer is controlled by the program, using a single I/O instruction per word exchange;
- external control: a seventeenth line (CALL line) from the external system to each word permits a data exchange request on the corresponding data path. When one or more of these lines are activated, the DIOS generates an interrupt signal to the CPU.

The diagram on the following page shows the basic DIOS connected to the I/O Bus.

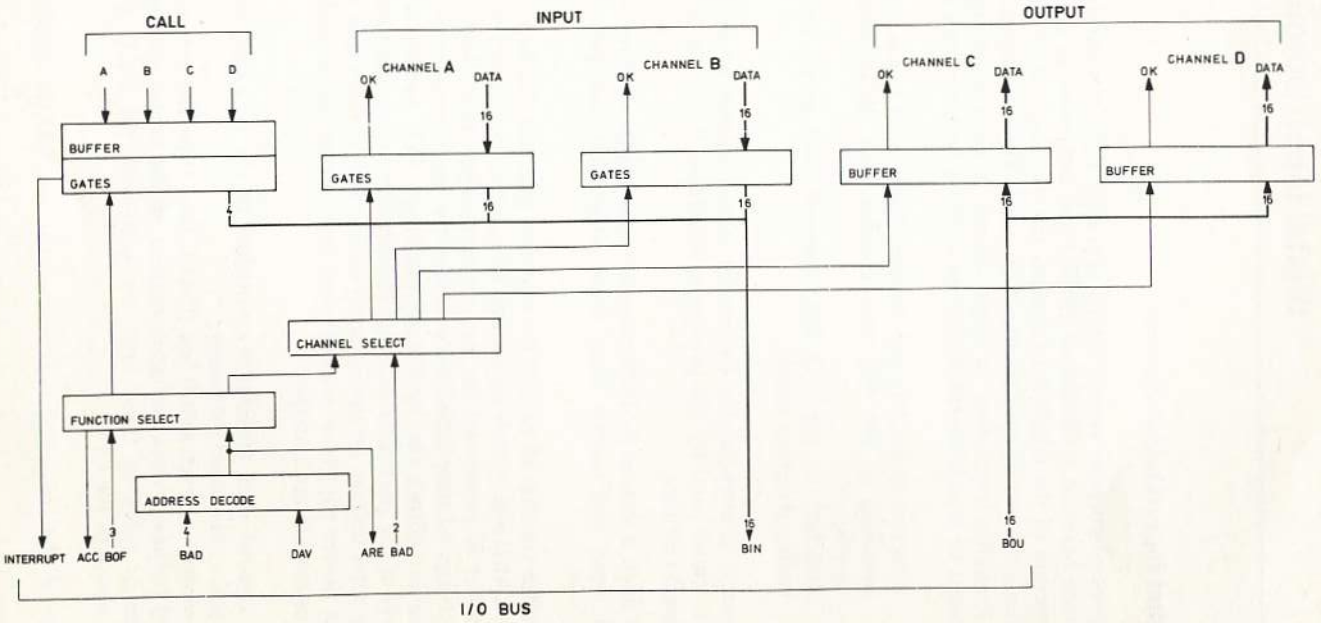


Fig. 9 Basic DIOS

MIOS (Modular Input/Output System)

MIOS is an interface system for application in process control to act as an interface between the computer and any external equipment. It performs such tasks as:

- signal acquisition and conditioning (including filtering and rationalisation of input levels);
- sequential interrogation of the various inputs;
- analog-to-digital conversion of inputs representing physical quantities, for acceptance by the computer as data words;
- provision of outputs for display, control or data acquisition purposes.

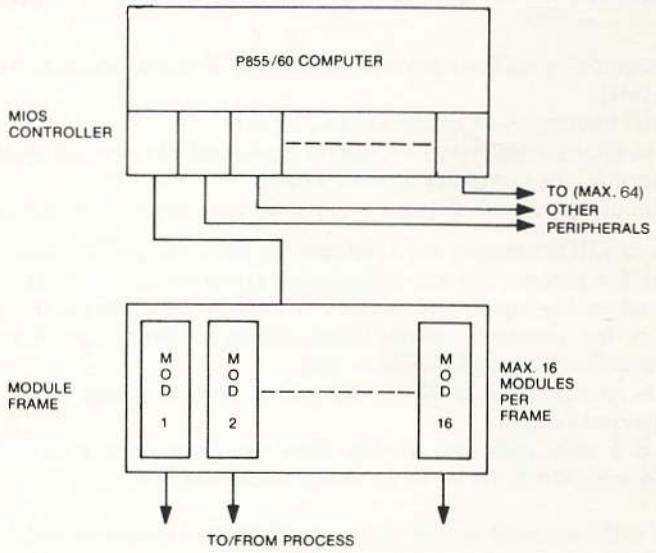
Four types of MIOS systems are available for different applications:
MIOS-4S is for process control and similar applications where large numbers of digital and analog inputs and outputs of various signal levels are involved.
MIOS-4D is for relatively simple applications involving small amounts of digital input and analog and digital output.
MIOS-4A is an extension of MIOS-4D and is able to handle both digital and analog input and output.
MIOS-4C is a unit with one analog input and one analog output, used in conjunction with MIOS-4A or as an independent channel.

MIOS is a self-contained system built up of printed circuit modules housed in standard frame assemblies, designed for mounting in 19-inch racks. The modules are connected to MIOS control unit cards which are inserted in the standard C.U. sockets of the I/O bus. In the case of MIOS-4D, each card may drive up to 16 MIOS modules. The modules, in turn, are connected to the external equipment.

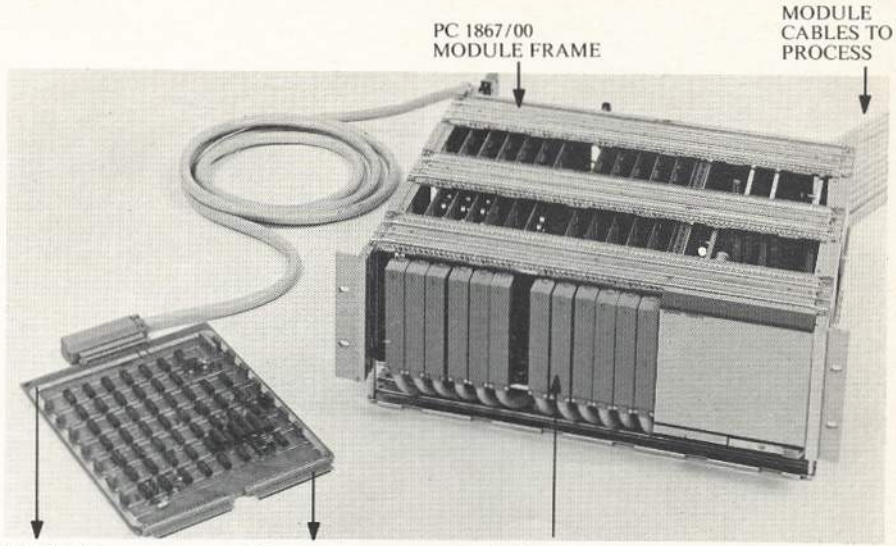
The following modules are available for MIOS-4D and MIOS-4A:
- Digital Input Isolation Module
- Digital Input Solid-state Module
- Digital Input Counter Module
- Digital Input Priority-interrupt Module
- Digital Output Solid-state Module
- Digital Output Control Module
- Analog Output Fast Module
- Analog Output Control Module
- Analog Input Low-level Module
- Analog Input High-level Module
- Analog Input Solid-state Module

Other MIOS items available are:
- Data Amplifiers
- Analog Input Data Concentrator
- Analog-to-Digital Converter.

The following figure shows the basic set-up of the system:



Per Module:
16 solid-state digital inputs
16 galvanically isolated digital inputs
16 program interrupt digital inputs
2 seven-bit counter inputs
16 medium-level solid-state digital outputs
1 pulse output
2 high-speed analog outputs
1 analog output for control purposes.


PC 1867/00
MODULE FRAME
MODULE
CABLES TO
PROCESS
PC 1203/00
MIOS CONTROL UNIT
TO P855/P860 I/O BUS
MIOS MODULES

Data communication may be defined as the technology of transmitting all kinds of information from one point to another, remote point.
To meet the ever increasing need of reliable data communication equipment Philips have designed a number of devices which handle the transfer of data from a terminal to a computer and vice versa, via the switched network, leased lines or private lines.
These devices are:

P845-050	MELCU	Multiple Equipment Line Control Unit
P845-150	MELCU	Repeater
P845-102	MIU	Modem Interface Unit
P846-950	ASYLCU	ASYnchronous Line Control Unit
P848-001	CPC	Clock Pulse Card
P848-002		Option for extending the range of frequencies produced by CPC
P847-950	SYLCU	SYnchronous Line Control Unit
P847-101	DERCO	DEtection Redundancy Check Option

All units are built on a standard card (20.8×31 cm) so that they fit in the standard rack equipment.
A short description of each unit and its function in a configuration is given below:

MELCU This line control is designed for low speed, asynchronous bit-by-bit transmission (up to 300 bits per second). Each unit controls 8 full duplex lines which must be of the same speed. It may be used inplant, connected to 8 ASRs with V24 interface as well as outplant. In that case it is connected to 8 V21 modems (one for each line) which are controlled by MIU (see below).

MELCU Repeater. This control unit has the same features as the MELCU described above and a T.T.L. interface.

MIU This unit is required when MELCU is to be connected to the switched networks. It controls the operation of a V21 modem.

ASYLCU A line control unit for asynchronous transmission over one line with speeds of 600, 1200 or 2400 bits per second. Connection to a V23 modem.

CPC The clock pulse card is required for the asynchronous control units

MELCU and ASYLCU. It provides the clock pulses necessary to shift the data bits. The frequencies delivered by this card are: 50, 75, 100, 110, 150, 200, 300, 600, 1200 and 2400 Hz as well as 15 times those frequencies. Other frequencies are available as an option.

SYLCU — This control unit handles the synchronous transmission and is connected to one V24 modem. The unit is half-duplex oriented and is transparent to any line procedure. SYLCU may be used in conjunction with DERCO. Speed is 600 to 50,000 bits/second.

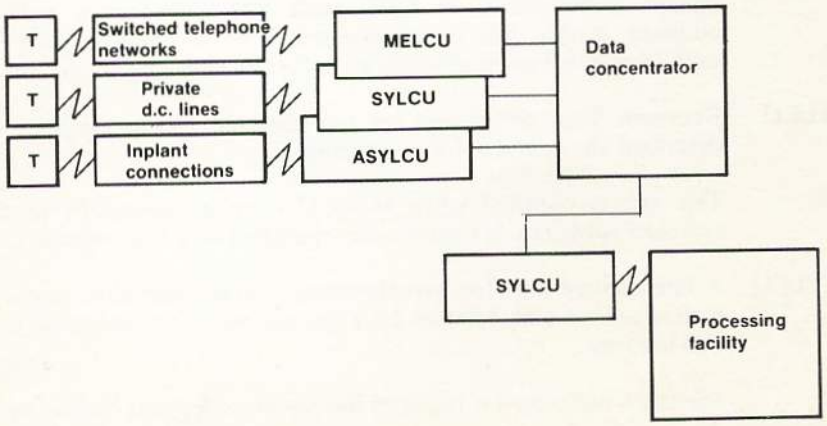
DERCO — This card has to be used with SYLCU and allows to have a hardware Cycle Redundancy Check and the detection of control characters in the Binary Synchronous Communication procedure.

The application of those devices in different configurations provides a variety of data communication systems such as:

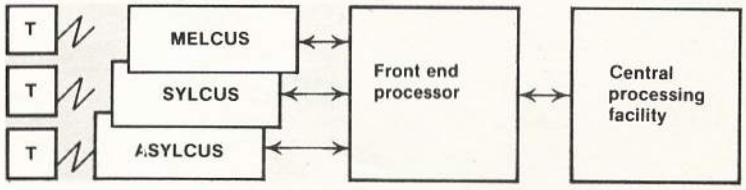
- data concentrators
- front-end processors
- terminal controllers.

Data concentrator

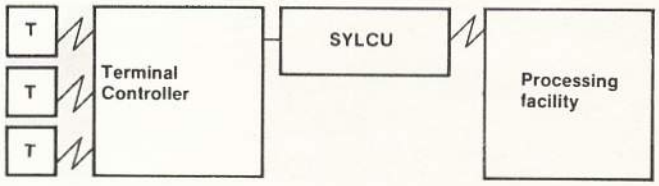
A data concentrator is a configuration in which data is sent from a cluster of terminals to a processor which performs certain functions to allow the data, coming from these terminals, to be sent to a central processor where the data is actually processed.
This feature avoids the use of many lines and the cost advantage may therefore be considerable as the use of the lines is charged on a time/distance basis.



Front-end processor
The set of data communication devices connected to the P800 series computer may be used to provide a low-cost alternative to hard-wired communication controllers on the front-end of large computers thus freeing the central processing facility to process information.



Remote Terminal Controller
To allow remote access to a batch processing facility the information to be processed is stored on punched tape, cards, magnetic tape or any other data carrier.
The output can be displayed on a CRT, stored on magnetic tape or disc or printed on a line printer.



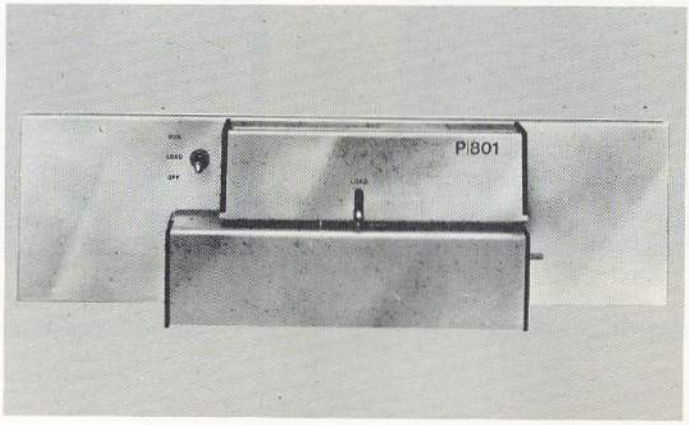
Software for data communication
Apart from the hardware designed for data communication Philips has also developed software for it as an extension of the existing monitors, drivers for the control units, a synchronous communication procedure and a Cycle Redundancy Check for the synchronous procedure.

The following range of peripheral equipment is available for use with the P855M and P860M computers:

PUNCHED TAPE EQUIPMENT

Punched Tape Readers

Two punched tape readers are available for punched tape input to the system. The readers are photoelectric types; model P801-001 with a reading speed of up to 333 characters per second, and model P802-001 with a speed of up to 600 characters per second.



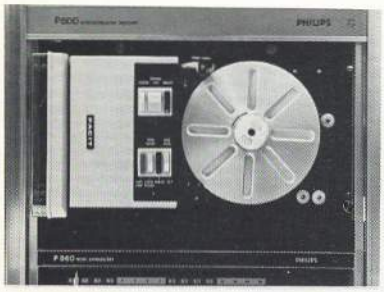
P801-001 Tape Reader

Model	P801-001 (Digitronics 2540 EP)	P802-001
Max. reading speed (characters/sec.)	333	600
Size (w × h × d) (mm)	483 × 133 × 203	483 × 133 × 203
Weight (kg)	15	15
Power consumption	150 VA	150 VA
Operating temp.	0-45 °C	0-45 °C
Relative humidity	20-85%	20-85%

Tape Punches

There are two tape punches available, both using standard one-inch tape. Model P803-001 operates at a speed of up to 75 characters per second, and the other, P804-001, is capable of punching up to 150 characters per second.

Once a punch has been loaded with tape and power switched on, it operates under computer control.



P803-001 Tape Punch

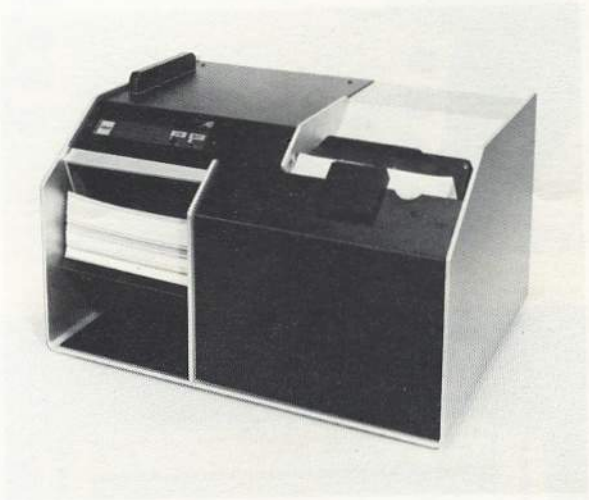
P804-001 Tape Punch

Model	P803-001 (Facit 4070)	P804-001 (Facit 5107)
Maximum speed (characters/sec.)	75	150
Size (w × h × d) (mm)	220 × 190 × 432	483 × 311 × 265
Weight (kg)	13	26
Power consumption	200 VA	220 VA
Operating temp.	0-45 °C	10-40 °C
Relative humidity	20-85%	20-85%

PUNCHED CARD EQUIPMENT

Card Reader

One card reader is available, the model P806-101. It reads 285 80-column cards per minute. Operation of the reader is automatic under computer control, and operator attention is required only for loading and starting, and clearing defective cards.
The (input)hopper and (output)stacker each hold up to 550 cards.

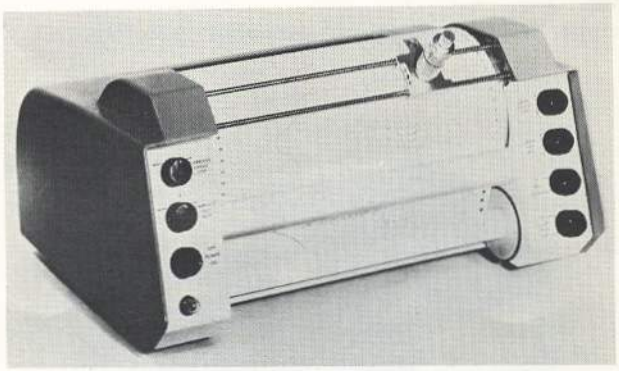


P806 Card Reader

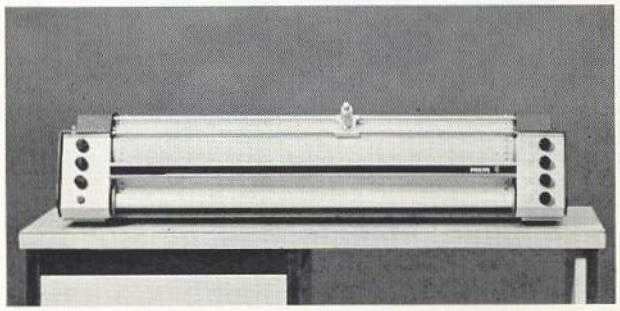
Model	P806-101 Documation M200
Reading speed for 80-col. cards per min.)	285
Card capacity hopper	550
Card capacity stacker	550
Size (w × h × d) (mm)	489 × 279 × 355
Weight (kg)	27
Power consumption	525 VA
Operating temp.	15-25 °C } Limits imposed by the cards
Relative humidity	50-70% }

PLOTTER EQUIPMENT

Two plotters are available. They each operate at a speed of 300 steps per second, with a step size of 0.1 mm. The difference is in the plotting width, which is 11 inches for the model P813-001 and 29.5 inches for the model P813-002.



P813-001 Plotter



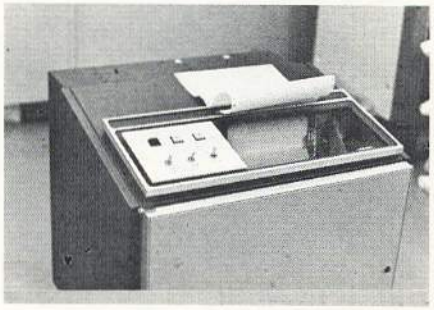
P813-002 Plotter

Model	P813-001 (CalComp 565,563)	P813-002
Steps per second (0.1 mm steps)	300	300
Plotting width	11 in	29.5 in
Size (w × h × d) (mm)	457 × 250 × 375	1000 × 250 × 375
Weight (kg)	20	65
Power consumption	175 VA	175 VA
Operating temp.	0-55 °C	0-55 °C
Relative humidity	20-85%	20-85%

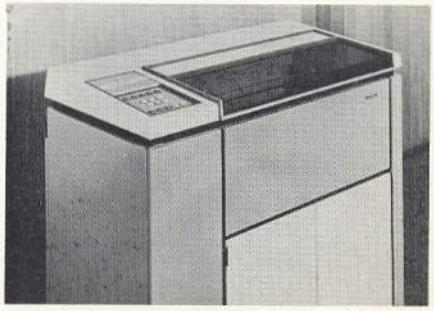
LINE PRINTER EQUIPMENT

Three types of line printer are available:

- model P810-001, providing printout from the computer at a speed of up to 356 lines per minute. Each line is 80 characters long and the character drum contains 64 characters.
- model P811-001 has a printing speed of up to 245 lines per minute. The line length is 132 characters and the character drum contains 64 characters.
- model P812-001 is the same as model P811-001, but has a printing speed of up to 670 lines per minute. The line length is 132 characters and the character drum contains 64 characters.



P810-001
Line Printer



P811-001
Line Printer

Model	P810-001	P811-001	P812-001
	(Data Products 2310, 2420, 2440)		
Printing speed (lines per minute)	356	245	670
Line length	80 col.	132 col.	132 col.
Size (w × h × d) (mm)	610 × 584 × 559	1232 × 1168 × 622	1232 × 1168 × 622
Weight (kg)	84	272	362
Power consumption	330 VA	500 VA	1.40 kVA
Operating temp.	10-43 °C	10-43 °C	10-43 °C
Relative humidity	30-80%	30-80%	30-90%

DISC EQUIPMENT

Two types of disc unit are available:

- model P821-001, a fixed-head disc with a capacity of 210k words and an average access time of less than 8.5 milliseconds.
- model P822-001, a moving-head disc with a capacity of 2.7 million 8-bit characters. The average access time is less than 125 milliseconds. The disc cartridge is top-loaded onto the drawer-type disc drive which can be mounted in a standard 19-inch rack.

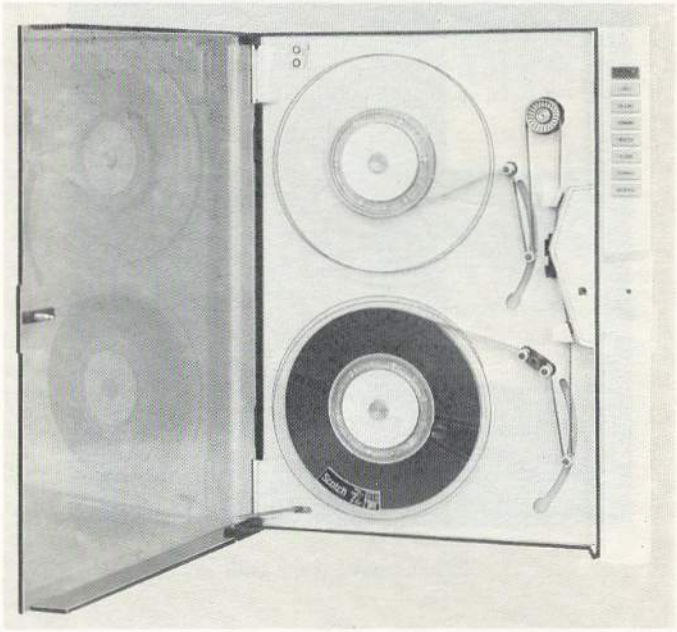
P821-001 Disc Unit

P822-001 Disc Unit

Model	P821-001 (SAGEM MS 300)	P822-001 (Philips X1210)
Head type	fixed	moving
Capacity	210k words	2.7M characters
Average access time	8.5 msec.	125 msec.
Transfer rate	200.000 ch./sec.	100.000 ch./sec.
Format	Sectors of 200 words of 16 bits	
Size (w × h × d) (mm)	483 × 311 × 462	483 × 260 × 725
Weight (kg)	40	45
Power consumption	270 VA	150 VA
Operating temp.	0-50 °C	10-32 °C
Relative humidity	0-95%	20-80%

MAGNETIC TAPE EQUIPMENT

Two magnetic tape units are available, differing only in speed: 25 inches per second for the model P831-002, and 45 inches per second for the model P831-004. Both units use 9-track tape, have a density of 800 bits per inch, 10.5 inch reels and operate on a read-after-write basis. Data are recorded by means of the NRZ1 method. The format is IBM compatible.

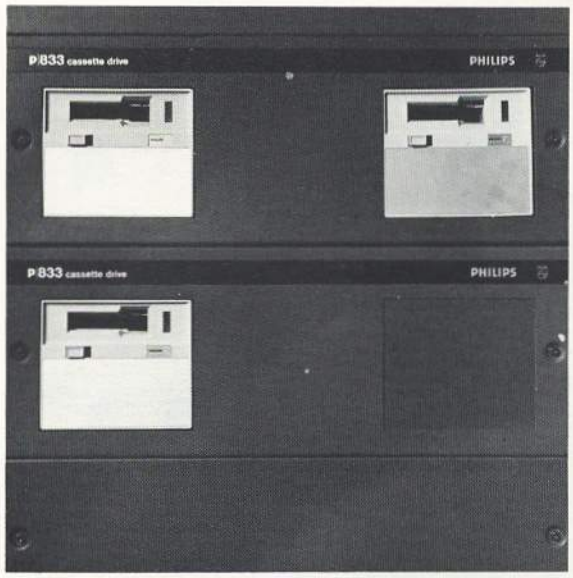


P831 Magnetic Tape Unit

Model	P831-002 (PERTEC 6840-9)	P831-004
Tape speed (inches per second)	25	45
Recording method	NRZ1	NRZ1
Density (bpi)	800	800
Transfer speed (characters/sec.)	20,000	36,000
Size (w × h × d) (mm)	483 × 622 × 318	483 × 622 × 318
Weight (kg)	38	38
Power consumption	300 VA	300 VA
Operating temp (for the tape)	15-32 °C	15-32 °C
Relative humidity	15-95%	15-95%

CASSETTE TAPE EQUIPMENT

One cassette tape unit is available, the model P833-001. This is a two-track serial recorder with a capacity of 338,000 characters per track, operating at a speed of 7.5 inches per second, with a density of 800 bits per inch.



P833 Cassette Tape Unit

Model	P833-001 (Philips ELA)
Capacity (per track)	338,000 characters
Tape speed (inches per second)	7.5
Density (bpi)	800
Transfer speed (characters/sec.)	750
Size	2 units + power mounted in 19-in. shelf
Weight (kg)	3.5
Operating temp.	0-50 °C
Relative humidity	5-95%

TYPEWRITER EQUIPMENT

Four typewriters and a character printer with keyboard are available:

- model P841-001, a light duty typewriter with a speed of 10 characters per second. With tape reader and punch.
- model P841-002, a heavy duty typewriter with a speed of 10 characters per second. With tape reader and punch.
- model P841-003, a light duty typewriter with a speed of 10 characters per second.
- model P841-004, a heavy duty typewriter with a speed of 10 characters per second.
- model P842-001, a matrix printer with keyboard; printing speed is 50 characters per second (see next page). It can be used without keyboard, as a line printer.

P841 Typewriter

Model	P841-001 (ASR 33) light duty	P841-002 (ASR 35) heavy duty	P841-003 (KSR 33) light duty	P841-004 (KSR 35) heavy duty
Printing speed (characters/sec.)	10	10	10	10
Punched tape	yes	yes	no	no
Paper width	216 mm	up to 241 mm	216 mm	up to 241 mm
Size (w × h × d)	560 × 1140 × 470	1016 × 978 × 610	473 × 835 × 470	510 × 978 × 610
Weight (kg)	25	100	23	61
Power consumpt.	300 VA	300 VA	250 VA	300 VA
Operating temp.	0-45 °C	0-45 °C	0-45 °C	0-45 °C
Relative humidity	20-85%	20-85%	20-85%	20-85%

Model	P842-001 (Philips PTS 3100)
Printing speed (characters/sec.)	50
Platen width (peg-distance)	231 mm 303 mm 314 mm
Paper width: friction-fed peg-fed	148-299 mm 216 mm 288 mm 299 mm
Size (w × h × d) (mm)	500 (knob excl.) × 166 × 455
Weight (kg)	20
Power consumption	100 VA
Operating temp.	10-40 °C
Relative humidity	20-80%

The user presents his application problem to the P855M/P860M computers in the form of a source program written either in the machine oriented language, Assembly Language, or in the problem oriented language, FORTRAN. They are converted into machine code by either the Assembler or the FORTRAN compiler. The resulting object program can be linked to other programs to which it makes external reference by the Linkage Editor.
Execution of the user's program takes place under supervision of one of the four monitors available. If the user so wishes, he can test his object program with a Debugging package and update his source programs, either at the statement or the character level, with the packages Update and Text Editor, respectively. These system programs and packages comprise the software available to users of P855M/P860M computers. They can be divided into three categories:

- Processing programs
- Control programs
- Service programs.

Moreover, a Mathematical Library is available, including arithmetic routines to perform basic operations on single or double precision floating-point and complex numbers, as well as operations for standard mathematical functions on these numbers.
The library is used mainly by FORTRAN programs, but any of the library routines may also be called directly by any program in assembly language.
For loading programs, several loaders are available. First a bootstrap, a hardware loader must be entered manually by means of the data switches on the control panel. This bootstrap loader will load absolute programs, e.g. stand-alone systems, absolute user programs or other, relocatable loaders. These other loaders may be a so-called mini-loader or an IPL (Initial Program Loader). Of these two the IPL is the more sophisticated one, being able for example, to output messages. The IPL is used to load the monitor, which in turn will load the user program, if a loader is included in the monitor modules. Otherwise, the miniloader can be used to load the user programs.

PROCESSING PROGRAMS

These are the Assemblers, FORTRAN compilers and Linkage Editor.

Assembler

This program converts Assembly Language instructions into machine language instructions.
Each machine instruction corresponds to a single Assembly Language instruction and possibly extended by a data word. The Assembler also prints an output listing and error diagnostics.
There are four versions of the Assembler available. A stand-alone version that will operate with the minimum configuration of 4k words of memory. A basic, monitor-controlled version that requires a minimum of 8k words of memory. With this assembler, it is possible to recover source errors directly. The Extended Assembler (monitor-controlled) offers some additional directives and is available in a disc and in a non-disc version. Both operate in a minimum system of 12k. The P850M 2k Assembler can be used on a P855M or P860M as well.

FORTRAN compilers

The Basic FORTRAN compiler translates Basic Fortran source programs into interpretive code object modules to be link-loaded before execution. The compiler is self-initializing and does not require reloading between successive compilations.
The minimum memory requirements are 5k words.
There are two versions of the Basic FORTRAN compiler: a Stand Alone version and one designed to run under monitor control. The compilers are identical, as are the language elements. The differences are in the operating procedures and the system libraries (the system library for the Stand Alone version contains I/O routines which are part of the monitor). The monitor controlled system accepts source programs from, and outputs object modules to, any of the peripheral devices that may be connected to an 8k configuration. The Stand Alone version is restricted to a punched tape system.
Also available is a Full FORTRAN IV compiler with a minimum memory requirement of 8k, so it is usable in a 12k configuration.

Linkage Editor

The user should avoid writing large, complex programs; instead, the problem should be broken down into several stages and each stage written as a separate module.
If external references are included in the modules it will be possible to link them together. This facility is provided by the Linkage Editor. The advantages of modular programming are:

— each module can be written in the most suitable source language;

- several programmers could be employed on the problem to advance the completion date;
- frequently used routines can be stored in an object library and linked to whatever program requires them;
- the possibility of errors is reduced and testing and correction is simplified;
- future updating of the source text is made easier.

The minimum configuration required by the Linkage Editor is:

- 4k words of memory;
- operator's typewriter;
- high speed tape reader;
- high speed tape punch (for link edit operations).

The Linkage Editor will operate in a link edit mode or in a link load mode.
In link edit mode the Linkage Editor joins a number of object programs (output by the Assembler, FORTRAN compiler or the Linkage Editor itself) and/or object libraries to form one larger object program. The resulting program is output onto a suitable medium and may contain unsatisfied external references. If all external references are satisfied, the program may be loaded by the program loader and executed. Otherwise, it may be stored in an object module library or input to a further linkage process.
In link load mode the Linkage Editor loads a number of object programs and/or object libraries into memory where linkage will take place. All external references must be satisfiable.

CONTROL PROGRAMS

Loading, execution and supervision of a user program is handled by one of the monitor programs. There are four monitors available:

- Basic Operating Monitor
- Basic Real Time Monitor
- Disc Operating Monitor
- Disc Real Time Monitor

Although the Basic Monitors are punched tape oriented they can be extended to handle other peripheral devices.
The monitors have a modular structure to enable the user to select only those modules necessary to fulfil his system requirements. At system generation time, the selected features can be merged into a single module. As the user environment becomes more demanding, so the appropriate modules can be added to the monitor.

The Real Time Monitors are capable of monitoring several programs

concurrently. The Basic Operating Monitor and the Disc Operating Monitor handle one program at a time; however, a feature known as 'scheduled labels' enables a form of multi-tasking to be achieved.

Basic Operating Monitor
The minimum configuration requirements of the BOM are 8k words of memory plus an operator's typewriter with ASR tape equipment.
The BOM, which is intended mainly for program development, can occupy between 1k and 3k words of memory depending on what optional features have been selected by the user. The smallest monitor would contain the dispatcher module and a limited range of monitor requests, but no operator communication modules. The largest monitor would also feature the dynamic allocation module (which gives the user access to the higher end of memory beyond the user program area to obtain temporary memory space), operator communication with the system via the typewriter and line printer and punched card handling modules.

The BOM is loaded (by the program loader) into the system area which is the lower part of memory. This area is optionally protected against user access. The monitor loads user programs into the user area beyond the system area.

Disc Operating Monitor
The minimum configuration requirements of the DOM are 12k words of memory, an operator's typewriter and one fixed head or moving head disc.
The system and user programs are disc resident. The user program is executed during a session. A session is opened by the user presenting his identification to the system and closed by a control command from the user. The user identification is a security feature which has previously been declared to the system and stored in a catalogue on the disc. For each user, the catalogue points to a directory, on the same disc, containing the names and locations of the user files. The user files are contained within one user library on this disc. The system is handled as any other user is, with its identification entered in the catalogue and with its own library, i.e. the system software components.
If the user program requires to create a file during a session, space is allocated dynamically on the same disc as his library resides, i.e. the user program cannot write on other discs.
However, access can be gained to the System Library through control commands and to libraries of other users in read-only mode, which need not be on the same disc as his library.
Files created in a session are considered temporary, though the user can make them permanent by issuing a control command to store them in his library.
Communication with the system is through control commands entered on the typewriter, but a card reader or punched tape reader may also be used, making the system batch-processing oriented.
Via these commands, the user can create and delete files, call the processors, handle his library and execute his programs.

It is also possible to keep programs that have been prepared for execution and have them executed under control of the Disc Real Time Monitor, as the two systems are compatible.

Basic Real Time Monitor

The minimum configuration required for the Basic Real Time Monitor is 8k words of memory and an operator's typewriter with ASR punched tape equipment.

The BRTM supervises the execution of up to 14 user levels in a real time environment. The monitor is very modular in structure so that the user can minimize its memory occupation by selecting only those modules which he needs for his application.

The real time features are based on a system of priority levels, to schedule the hardware interrupts and the user programs:

- levels 0 to 47 are hardware interrupt levels
- levels 48 to 63 are software priority levels, of which 48 and 49 are the levels on which the monitor modules operate.

The highest level is 0, so hardware interrupts always overrule user software level programs.

A dispatcher which runs on level 48, schedules all activities, passing control to the highest level program and handling incoming interrupts. Upon interrupt, registers are saved in a system stack, addressed by register A15, and the routine servicing the interrupt is started. User-written interrupt routines can easily be included in the system.

Temporary memory space, which may be needed by user programs and (reentrant) subroutines, may be obtained in the dynamic memory allocation area, i.e. the area of memory remaining after all user programs have been loaded. This is done by means of requests to the monitor for allocation and deallocation of memory space.

Monitor requests enable the user programs to perform a number of other functions as well, such as different types of I/O and I/O control, activation, level connection and disconnection, time-slicing, waiting for an event, exit, timer handling, reserving exclusive use of a certain peripheral device, etc.

In conjunction with monitor requests, the 'scheduled label' feature can be applied, as with all monitors, to enhance the system's multiprogramming aspects.

A scheduled label is a subroutine which is started when the monitor request function has been completed, although it is specified at the same time as the monitor request, i.e. the main program can continue while the requested function is performed. For example, combined with an I/O request, the main program continues while the requested I/O function is performed and the branch to the scheduled label routine is not made until that operation has been terminated. This can be very useful, for example to analyze the results of a monitor request.

Disc Real Time Monitor

The DRTM requires a configuration of at least 8k words of memory, an operator's typewriter and one fixed head disc.
The features of the system are similar to those of the BRTM. In addition, programs may be disc resident and the system software is also stored on disc. Disc-resident programs are loaded into a special memory-partition as read-only programs.

SERVICE PROGRAMS

Two packages are available for updating user programs, namely Update and the Text Editor. A program debugging package is also available.

Update

The Update program (available as a stand-alone or monitor-controlled program) enables the user to update his Assembly and FORTRAN source programs by inserting and deleting lines and modules. Object programs may be updated by inserting and deleting modules. The user programs may reside on any acceptable medium. The minimum configuration in which Update will work is 4k words of memory, an operator's typewriter, a high speed punched tape reader and high speed tape punch.
A cassette tape unit may be substituted for either the tape reader or the tape punch, but not for both reader and punch.
Update consists of two modules. The first module accepts and executes the control messages, introduced by the operator, which contain the correction parameters. It then transfers the updated text, line by line, to the second module. For updating object modules, the control messages contain the names of modules to be inserted or deleted. The second module outputs the updated program onto the file specified in an option control message.
The first module is also included in the Assemblers and FORTRAN compilers to facilitate executing temporary corrections without changing the actual program on the data carrier.

Text Editor

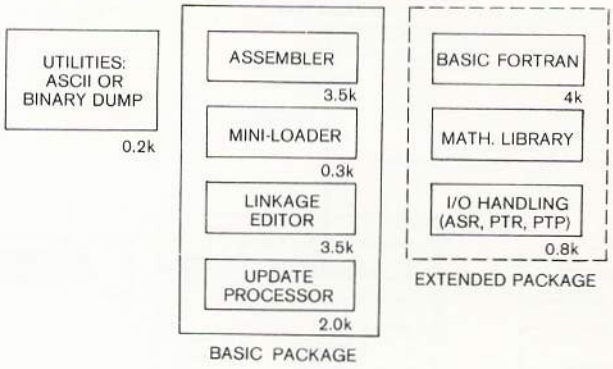
The Text Editor updates punched tape input programs on a character or line level. It operates under monitor control.
The minimum configuration requirements are 8k words of memory and an operator's typewriter. When all control messages have been processed, the updated program is output on the specified device.

Debugging Package

The Debugging Package allows the user to change certain parts of his program during execution. The program is made to stop at certain addresses (break-points), defined in Debug control messages and give control to the user. He may then ask for memory or register dumps, modify the contents of memory locations, etc. in order to produce a correct program.

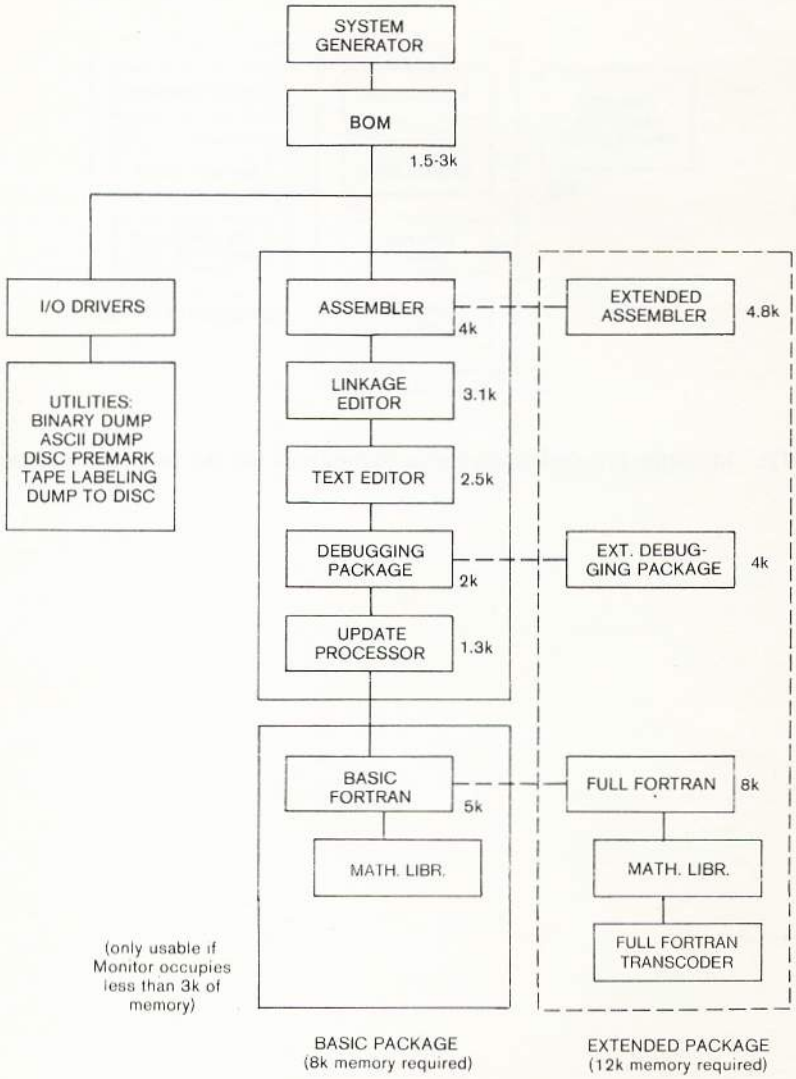
SOFTWARE CONFIGURATORS

P855M/P860M Stand-alone Software



NOTE: Multiply/Divide (hardware) is mandatory for the extended package.

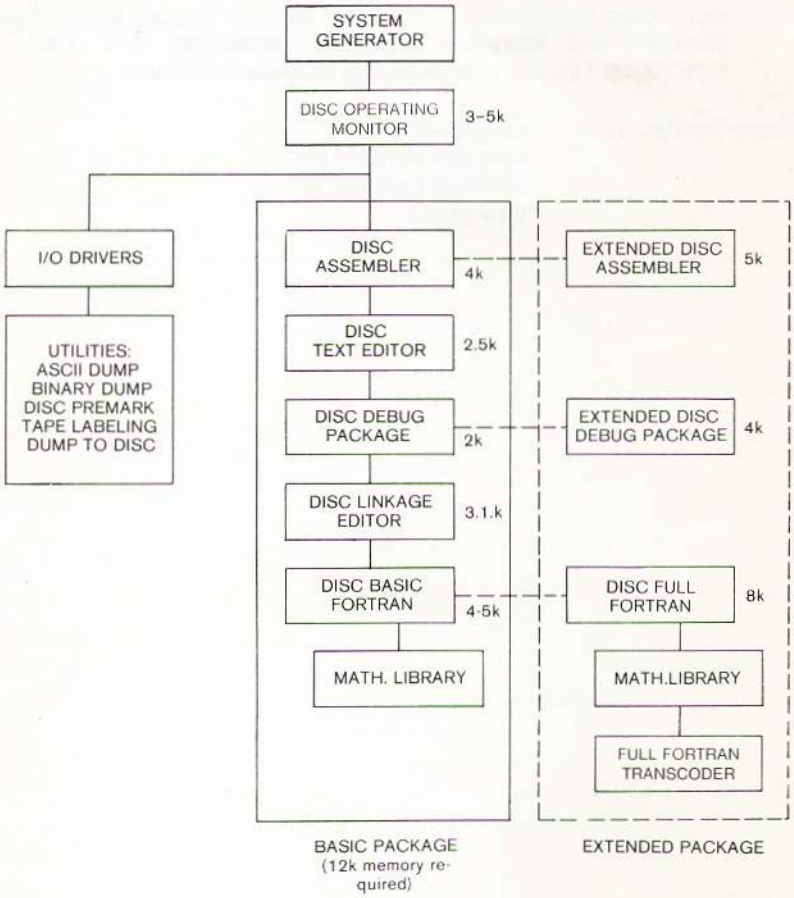
P855M/P860M software attachable to BOM (Basic Operating Monitor) in an 8k or larger system.



NOTE: The software is contained on paper tape.
Multiply/Divide (hardware or simulated) is mandatory.
This software can also be used under BRTM and DRTM, but for the user's own responsibility, as these are not foreground/background monitors and therefore offer no protection. The Real Time FORTRAN Library is to be used with these monitors.

Minimum configuration:
- 8k memory
- operator's typewriter
- punched tape reader
- tape punch

P855M/P860M software attachable to the Disc Operating Monitor (DOM).



NOTE: Multiply/Divide (hardware or simulated) is mandatory. The software is contained on disc.

Minimum configuration:
12k memory
operator's typewriter
punched tape reader
tape punch
one disc unit

Besides whichever is relevant of the mini-computer system manuals listed below, reference should also be made to the peripheral equipment manuals issued by the various suppliers and described in the P800M Documentation Catalogue (publication number 5122 991 2335X).
The last digit of a publication number, shown here as an 'X', denotes the issue level and is therefore subject to change.

title	publication number	general	programming	operating	servicing	training
P850M User Manual	5122 991 1453X	X	X	X		X
P850M Reference Data	5122 991 1457Xp		X	X		
P850M System Introduction	5122 991 1458X				X	X
P855M System Description	5122 991 1137X	X				X
P850M/P855M Reference Data	5122 991 1690X		X	X	X	
P855M Basic Operating Monitor	5122 991 1135X		X	X		X
P855M Disc Operating Monitor	5122 991 1136X		X	X		X
P855M Basic Real Time Monitor	5122 991 1158X		X	X		X
P855M Disc Real Time Monitor	5122 991 1160X		X	X		X
P855M Small Real Time Monitor	5122 991 1232X		X	X		X
P855M Timer Function SRTM	5122 991 1233X		X	X		X
P855M MIOS Drivers	5122 991 1234X		X	X		X
P855M Basic FORTRAN Reference Manual	5122 991 1143X		X	X		
P855M Basic FORTRAN Reference Data	5122 991 1159X		X	X		
P855M Full FORTRAN Reference Manual	5122 991 1140X		X	X		
P855M Full FORTRAN Reference Data	5122 991 1167X		X	X		
P855M System Software Manual — Paper Tape	5122 991 1155X		X	X		X
P855M System Software Manual — Disc	5122 991 1162X		X	X		X
P855M Software Training Manual	5122 991 1149X	X				X
P850M Data Communication	5122 991 1129X	X	X	X		X
P855M Operator's Guide	5122 991 1138X			X		
P855M Interface Manual	5122 991 1147X		X		X	X
P850M/P855M Installation Manual	5122 991 1148X	X			X	
P850M CPU Service Manual	5122 991 1688X				X	X
P855M CPU Service Manual	5122 991 1231X				X	X
P855M CPU Logic Diagrams	5122 991 1169X				X	X
P855M Appendix C 'TTL DATA'	5122 991 1210X				X	
P850M/P855M Control Units — Slow Service Manual	5122 991 1230X				X	X
P850M/P855M Control Units — Fast Service Manual	5122 991 1691X				X	X
P800 Data Communication Service Manual	5122 991 1689X				X	X

PHILIPS data systems

Problem EXAMPLE Programmer A. V.D. LINDEN Date 11-11-1971 Page 1 of 3

LABEL	OPERATION	OPERAND	IDENTIFICATION
*THIS	IS A PR	ØGRAM TØ TRANSFER 20 CHARACTERS FRØM AN I/Ø DEVICE	
*(WITH	ADDRES	S=1), INTØ A BUFFER, ADD THEM TØGETHER IN PAIRS AND STØRE	
*THE	RESULT	IN THE SAME BUFFER.	
	IDENT	PRØGEX	
S	EQU	1	
H	EQU	0	
ASR	EQU	1	
BUF	RES	10 RESERVE BUFFER	
CØUN1	DATA	-20 INITIALIZE CØUNTERS	
CØUN2	DATA	-10	
START	LDK	A2,0 REGISTER A2 IS USED AS INDEX REGISTER FØR BUFFER	
*DURING INPUT.			
	LDK	A5,1 LØAD 1 FØR INPUT BEFØRE CIØ START.	
TSTST	TST	A1,1 TEST DEVICE STATUS.	
	ABL(3)	DEVUN BRANCH IF DEVICE UNKNØWN.	
	SRC	A1,1	
	ABL(2)	TSTST BRANCH BACK IF DEVICE IS BUSY.	
STASR	CIØ	A5,S,ASR START ASR.	
INPUT	INR	A3,0,ASR INPUT ØNE CHARACTER.	
	ABL(1)	SENST IF NØT ACCEPTED, GET STATUS WØRD.	
	SC	A3,BUF,A2 STØRE CHARACTER IN BUFFER.	
	IM	CØUN1 INCREMENT CØUNTER1.	
	ABL(0)	STDEV BRANCH IF ZERØ.	
	ADK	A2,1 INCREMENT INDEX REGISTER.	

PHILIPS data systems

Problem EXAMPLE Programmer A. V.D. LINDEN Date 11-11-1971 Page 2 of 3

LABEL	OPERATION	OPERAND	IDENTIFICATION
	ABL(7)	INPUT INPUT NEXT CHARACTER.	
STDEV	CIØ	A5,4,,ASR HALT I/Ø.	
	ABL(7)	INPUT BACK TØ INPUT.	
SENST	SST	A6,1 CØMMAND ACCEPTED? IF NØT BRANCH TØ INPUT	
*TEST	CONTENT	6, ØF STATUS WØRD IN A6.	
	SRC	A6,1	
	ABL(0)	ADDIT INPUT CØMPLETE, GØ TØ NEXT ØPERATIØN.	
	ABL(2)	NØTØP BIT15=1, DEVICE NØT ØPERABLE.	
	SRC	A6,1	
	ABL(2)	THERR BIT14=1, THRØUGHPUT ERRØR.	
	SRC	A6,1	
	ABL(2)	DATAF BIT13=1, DATA FAULT.	
	SRC	A6,1	
	ABL(2)	INCLE BIT12=1, INCØRRECT LENGTH.	
THERR	LDK	A4,1	
	ABL(7)	HALT	
INCLE	LDK	A4,2	
	ABL(7)	HALT	
NØTØP	LDK	A4,3	
	ABL(7)	HALT	
DATAF	LDK	A4,4	
	ABL(7)	HALT	
DEVUN	LDK	A4,5	
	ABL(7)	HALT	
ADDIT	LDK	A1,,0 SET INDEX REGISTER FØR BUFFER.	

PHILIPS

data systems

Problem EXAMPLE Programmer A. V.D. LINDEN Date 11-11-1971 Page 3 of 3

LABEL	OPERATION	OPERAND	IDENTIFICATION
	LDKL	A3,=/00FF LOGICAL CONSTANT INTO A3.	
SUM	LC	A2,BUF,A1 LOAD LEFT CHARACTER INTO A2.	
	ANS	A3,BUF,A1 CLEAR BITS 0 TO 7 OF BUFFER CURRENT WORD.	
	ADS	A2,BUF,A1 ADD RIGHT CHARACTER TO LEFT ONE AND STORE	
*INTO	BUFFER.		
	ADK	A1,2 INCREMENT INDEX REGISTER A1	
	IM	COUN2 INCREMENT COUNTER2.	
	ABL(2)	SUM LOOP ON SUM UNTIL COUNT2=0.	
HALT	HLT		
	END	START	
*WHEN	THE TABLE IS COMPLETE (I.E. COUN2=0) ITS CONTENTS CAN BE CONVERTED		
*TO	DECIMAL NOTATION AND EXTERNAL ASCII FORMAT, USING A UTILITY PACKAGE,		
*AND	THEN PUT ON THE OPERATOR'S TYPEWRITER.		