FERRANTI

F100-L

Hardware Data Book

COMPUTER SYSTEMS

Ferranti Limited

Microprocessor Marketing Unit
Western Road, Bracknell,
Berkshire RG12 1RA
Telephone: Bracknell (0344) 3232
Telex: 848117

This replica of the Ferranti F100-L Hardware Data Book has been created using Google Docs' optical character recognition (OCR) feature to turn photographs of an original copy into text. Tables and the simpler diagrams have been recreated as necessary while other figures have been retained as bitmap images. The replica does not attempt to duplicate the exact layout and typography of the original but the look and feel of the document has been followed very closely.

An original copy of the F100-L Hardware Data Book is held in the British Library:

Title: F100-L hardware data book.

Author: Ferranti Ltd.

Subjects: Microprocessors -- Handbooks, manuals, etc; Microcomputers -- Handbooks, manuals, etc; Dewey: 621.39

Publication Details: Bracknell : Ferranti Ltd., c1977.

Language: English

Identifier: System number 015900910

Physical Description: 109 p. : ill. ; 21 cm.

Shelfmark(s): General Reference Collection YD.2011.a.8067

UIN: BLL01015900910

A second copy of the book is held in the Ferranti Archive at the Manchester Museum of Science and Industry.

Release History

Version	Date	Changes/Errata
0.1	20th May 2019	First cut with all pages present. Some pin names were illegible on F111-L pin-out diagram. Many diagrams present as bitmap images.
0.2	7th June 2019	Minor formatting changes. Correct units for line driver current in Table 2.1
0.3	16th July 2019	Corrected illegible pin names on F111-L pin-out diagram.
0.4	12th Dec 2019	Fixed table errors in 1.4.3, 1.5.3.1.1.2, 1.5.3.1.3 and mis-OCR'd character in shift description
0.5	25th April 2020	Fixed Subtract with Carry Eqn in section 1.5.17, typo in section 1.9
0.6	29th April 2020	Fix typo for JMP .NNNN description (same as CAL .NNN) plus a number of minor typos (some also in original) in later text sections from 3.0 onwards
0.7	4th May 2020	Improve formatting of tables in section 1.5 for PDF generation

CONTENTS

SECTION 0

INTRODUCTION

INTRODUCTION

This F100-L Data Book outlines the facilities offered by the F100-L microprocessor and its associated hardware support products.

There are three sections covering the F100-L microprocessor itself, the LSI Interface Set, and the Microcomputer systems based upon the F100-L microprocessor.

It is not intended that this Data Book is a fully detailed manual. The emphasis has been on describing the facilities and the capabilities of the various items to a sufficient depth to enable a user to evaluate the F100-L for a given application.

The in-depth, fully detailed information that would be required to enable an item to be accurately designed into a system is provided in the various manuals.

The F100-L, the Interface Set and the system structure are detailed in the

F100-L Hardware & System Manual HSM 150

The cross-product support software and resident debugging aids are covered by the

F100 Software Manual PM 150

Further manuals will be available to cover the resident software and the microcomputer system

SECTION 1

THE F100-L MICROPROCESSOR

Contents

Tables

Figures

1.1 BASIC FEATURES

The F100-L is a single-address accumulator, fixed word length central processor.

The word length is 16 bit, with 2’s complement fixed point arithmetic.

Four addressing modes including direct and indirect (with or without auto-increment/decrement) are provided to a maximum of 32K words (i.e. 32,768 words)

Both conditional and unconditional jumps are provided where the jump address is a 15-bit word, giving access to all of the addressable memory.

Comprehensive bit manipulation instructions are included, along with a full set of double (32 bits) and single length shift instructions.

Subroutine entry and exit are simplified by the provision of Jump Store Link and Return (Obey Link) instructions where the Link is stored in a push-down, pop-up stack (Link Stack).

A program interrupt line is available, which causes the Link to be automatically stored in the Link Stack.

All memory and I/O devices share an I/O Bus. I/O transfers can be to a program addressed device, using one program instruction per word, or block transfers can be implemented using the Direct Memory Access (DMA) facility.

Provision is made in the instruction set for External Functions. This enables special purpose hardware (Special Processing Units) to be added to an F100-L to perform functions not included in the instruction set. The F101-L Fast Multiply/Divide chip currently under development by Ferranti is an example of the use of the External Function facility.

1.2 F100-L IN A SYSTEM (Fig. 1.1)

Fig. 1.1 shoes the typical additions to the processor to build up a simple system

The processor requires clock pulses for its logic which it receives from an external TTL level, single phase oscillator.

It also requires 2 control signals. These are $\overline{Reset}$ and $\overline{Run}$.

The $\overline{Reset}$ signal resets the processor to a known state and loads the Program Counter with a fixed Start Address (2048 or 16,384). (2K if Ad Sel left floating, 16K if it is tied to 0V).

The $\overline{Run}$ signal in the high (Stop) state prevents instructions from being obeyed. In the low state program execution begins at the address held in the Program Counter.

All memory and I/O Devices interface to the F100-L I/O Bus, using Interface Sets.

The Interface Set extends the use of LSI into the complete system, replacing the many conventional TTL circuits normally associated with the interfacing and control of memory and I/O devices.

The Interface Set provides an I/O device with the necessary control and priority multiplexing required for the use of the Direct Memory Access (DMA) facility. (See Sect. 1.10).

The Program Interrupt facility is also handled by the Interface Set, resulting in a fully vectored priority interrupt system (See Sect. 1.9)

Purther information on the Interface Set is given in Section 2.

In systems with ROM storage for the programs there is no requirement to load the store with program. For this sort of system no additional hardware is necessary.

Por systems with volatile memory a means of loading the program is required. With the processor in the 'Stop' condition a peripheral can gain control of the I/O Bus and write into the volatile memory. Thus any suitable peripheral (e.g, paper tape reader, magnetic tape cassette) with its associated interface can be used as a Program Loader.

The optional Control Handset includes a ROM bootstrap to enable programs to be entered into RAM in a simple, straightforward manner. Facilities for manually entering data into memory are also provided. The Control Handset interfaces to the system via the I/O Bus with additional signals going direct to the processor. For further details of the Control Handset see Section 1.14.

1.3 INTERNAL ARCHITECTURE

A block diagram of F100-L is given in Fig. 1.2.

At the heart of F100-L is the Function Unit, which performs functions such as Add, Subtract, AND, Non-Equivalence, increment and decrement. These functions are achieved upon or between the Accumulator register and the Operand register in one bit, serial, form. Shift type functions are also executed via the Function Unit, but using the register input multiplexers only.

The Condition Register holds bits representing the result of a Function Unit action, (e.g. result equal to zero) and the status of the machine, (e.g. fault detected). During Jump-store-link (CAL) instructions it is loaded into the Link Stack, but can be restored using a Return (RTN, RTC) instruction.

The Program Counter register holds the address of the next instruction to be obeyed. It is incremented automatically after each instruction is fetched, and also whenever it is used to access an operand from store, (Immediate Addressing). During jump instructions it is loaded with a new address derived during execution of the instruction

The Instruction Register acts as both an instruction holding register and as an address register. In order to preserve the essential bits of the instruction when the register is required to contain an address, these bits are transferred to the Function Latches. The Operand Register when it is not holding an operand, may also act as an address register.

Parallel transfers between registers, and from registers to the F100-L I/O Bus, are achieved through the 16 bit parallel Internal Highway. In order to isolate this from the I/0 Bus a set of transmit and receive gates is provided.

Internally, F100-L is controlled by a cycle and beat counter. All instructions can be broken down into a set of function cycles, varying from two to five in number. Most function cycles will involve access to memory. In addition each function cycle may be further divided into one, two or three "beats". The cycle and beat counter generates the correct sequence for execution of the received instruction. Combining the cycle and beat counter outputs with the instruction fields and any additional control counters, provides the necessary control waveforms for the registers and the function Unit.

The miscellaneous control area deals with the external control signals. The Bus control lines handle address and data transfers over the I/0 Bus. The DMA Request/Accept handshake is used by devices other than F100-L to request permission to use the I/0 Bus. The Program Interrupt Request/Accept hand shake enables an external device to force a jump out of the program currently being executed into the interrupt program for that device. $\overline{ExtLdPgCt}$ allows the program counter value to be changed by an external device, whilst $\overline{ExtFnAccept}$

is used by Special Processing Units (SPUs) to indicate that there is no further need to hold up the microprocessor. $\overline{Rs}$ and $\overline{Run}$ are used respectively to set the processor to a standard state, and to cause it to start obeying instructions.

1.4 ADDRESSING MODES

The F100-L has 4 address modes:

- Direct (single word instruction)
- Pointer Indirect (single word instruction)
- Immediate Data (double word instruction)
- Immediate Indirect (double word instruction)

There is no need for a separate address mechanism for I/O devices. I/O devices share the I/O Bus with memory and can be allocated addresses within the 32K address range of the machine. Thus I/O devices can be “read from” or “written to” in the same manner as a memory location.

For high rates of I/0 transfer the F100-L has a Direct Memory Access (DMA) facility which is described in Section 1.10.

1.4.1 Direct Addressing (single word)

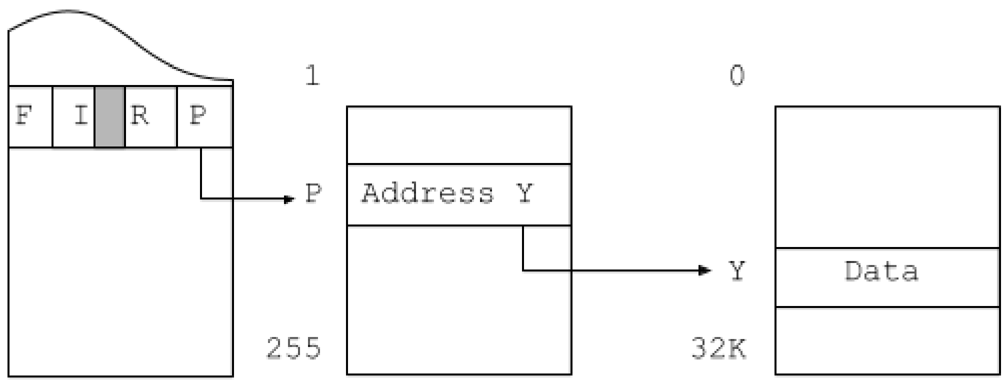
4 bits	1	11
F≠0	I=0	N≠0

The F field defines the operation to be performed upon the contents of location N. N can take the values 1 to 2047 inclusive. (N=0 is used to define Immediate Data addressing, see Section 1.4.3).

1.4.2 Pointer Indirect Addressing (single word)

4 bits	1	1	2	8
F≠0	I=0	▨	R	P≠0

Location P contains the address of the data, and P can have the values 1 to 255 inclusive. (P=0, is used to define Immediate Indirect addressing, see Section 1.4.4).



Optional auto-index is available with Pointer Indirect addressing, and is controlled by the 2-bit R field.

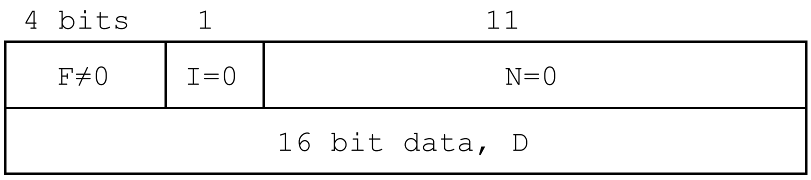
R=0,2	: No auto-index
R=1	: Positive auto-index
R=3	: Negative auto-index

When positive auto-indexing is specified (R=1), the address (Y) held in location P is incremented (Y->Y+1) before the address is used to access the data. Thus the data obtained will be that in location Y+1.

When negative auto indexing is specified (R=3), the address (Y) held in location P is decremented (Y->Y-1) after the store location has been accessed. Thus the data obtained will be that in location Y, but the value left in P will be Y-1.

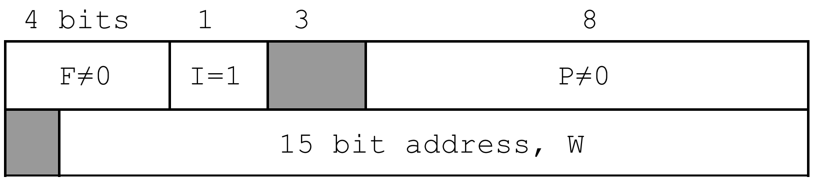
The auto-index facility, therefore, enables data to be sequentially "written to" or "read from" up to 255 data stacks.

1.4.3 Immediate Data Addressing (double word)



When N=0, the data is obtained from the memory location following that used to hold the instruction.

1.4.4 Immediate Indirect Addressing (double word)



When P=0, the location following the instruction provides a 15 bit address (W), which is the address of the data to be used in the instruction.

1.4.5 Immediate Indirect Addressing (shifts, bits, jumps)

1.4.5.1 Double Word

4 bits	2	2	2	2	4
F=0	T	R	S	J	B
▨	15 bit address, W				

If a memory location is being shifted, the address used is W.

If a bit in a memory location is being set/cleared, the address used is W.

If a bit-conditional jump is being performed upon an internal register then the jump address used is W.

1.4.5.2 3-Word

4 bits	2	2	2	2	4
F=0	T	R	S	J	B
▨	15 bit address, W				
▨	15 bit address, W1				

If a bit-conditional jump is being performed upon a bit in a memory location then W is used as the address of the memory location containing that bit, and W1 is used as the jump address.

1.5 INSTRUCTION SET (see Table 1.1)

1.5.1 Abbreviations

F	:	Function Field
I	:	Address Mode
N	:	Memory Address (11 bit) (direct addressing)
W	:	Memory Address (15 bit) (immediate indirect addressing)
P	:	Memory Address containing the pointer for pointer indirect addressing
(P)	:	Contents of location P
D	:	Immediate data (16 bits)
A	:	The Accumulator
R	:	Register Field or Auto-index mode for indirect addressing
J	:	)
T	:	) General fields to define F = 0 instructions
S	:	)
B	:	Shift Number or Bit significance
(N)	:	Contents of location N
PC	:	The current value of the Program Counter
(PC)	:	The contents of the next program location
((PC))	:	Contents of the contents of the next program location (i.e. the contents of the location whose address is in the next program location - Immediate Indirect)
X	:	Current value of the Link Stack pointer (location 0)
M	:	Multi-Length Staticiser
C	:	Carry Staticiser
S	:	Negative Sign Staticiser
V	:	Overflow Staticiser
Z	:	Function Zero Staticiser

1.5.2 Assembler Mnemonics

The Mnemonic used in the F100 Assembly Language is shown in parentheses after the title of each particular instruction.

1.5.3. F=0; Shift, Bit Manipulation, Jump, External Function, Halt

GENERAL FORMAT OF INSTRUCTION

4 bits	2	2	2	2	4
F=0	T	R	S	J	B

The F=0 instructions can be divided into 2 types.

(i) T=0; Shift, Bit Manipulation, Jump.

(ii) T≠0; External Functions, Halt.

1.5.3.1 F=0, T=0; Shift, Bit Manipulation, Jump.

4 bits	2	2	2	2	4
F=0	T=0	R	S	J	B

The F=0, T=0 instructions can be divided into 3 types:

(i) S = 0,1 Shift

(ii) S = 2 Jump

(iii) S = 3 Bit Manipulation

In all cases the R field defines the source of the operand.

R = 0, 2	Accumulator
R = 1	Condition Register
R = 3	Contents of the store location whose address (W) is specified in the location following the F=0 instruction.

1.5.3.1.1 F=0; T=0; S=0,1; Shift

4 bits	2	2	2	2	4
F=0	T=0	R≠3	S=0,1	J	B

F=0	T=0	R=3	S=0,1	J	B
▨	W				

The direction of shift is determined by the S value:-

S=0, Right shift

S=1, Left shift.

The state of the Multi-length Staticiser (M) in the Condition Register defines whether or not the shift is single (16 bit) or double (32) length:-

M=0, Single length

M=1, Double length.

When the Condition Register itself is shifted (R=1), it is treated purely as a bit pattern. The final state of the Condition Register is a function of the original bit pattern and the number of places shifted only, and is totally independent of the conditions that normally affect it (e.g. overflow, negative sign).

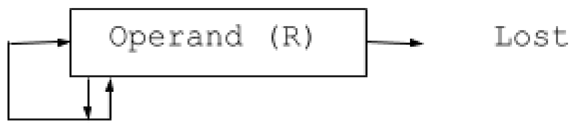
1.5.3.1.1.1 M=0; Single length shifts (SRA, SLA, SRL, SLL, SRE, SLE)

The number of places to be shifted is defined by the 4-bit B field (0 ≤ B ≤ 15).

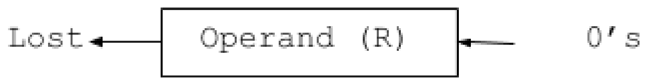
The type of shift is defined by the J field.

J = 0, 1 Arithmetic

S=0, Right arithmetic (SRA)
(sign digit regeneration)



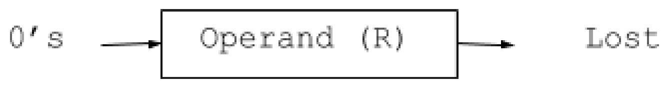
S=1, Left arithmetic (SLA)



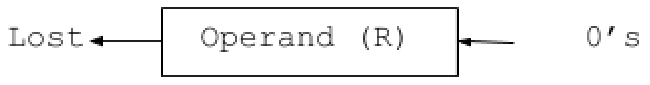
The V and S staticisers are set to the required state. C is unchanged and Z is meaningless.

J = 2, Logical

S=0, Right Logical (SRL)



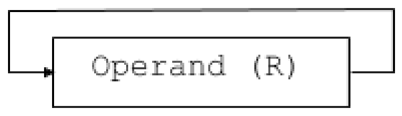
S=1, Left Logical (SLL)



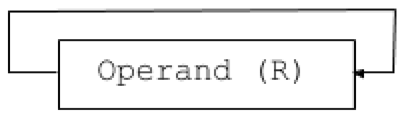
The Z, V and S staticisers are meaningless after this operation. The C staticiser is unchanged.

J=3, End-around (cyclic, rotate)

S=0, Right End Around (SRE)



S=1, Left End Around (SLE)



The Z, V and S staticisers are meaningless after this operation. The C staticiser is unchanged.

1.5.3.1.1.2 M=1; double length shifts (SRA, SLA, SRL. SLL).

The number of places to be shifted is defined by the 5-bit number formed between the least significant bit of the J field and the 4-bit B field.

i.e.

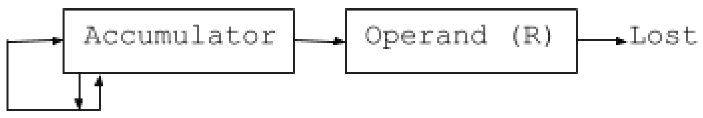
4 bits	2	2	2	2	4
F=0	T=0	R	S=0,1	J	B

<- 5 bit Shift number ->

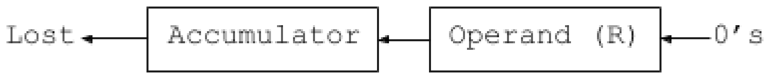
Thus a double-length operand (32 bits) can be shifted 0 to 31 places inclusive. The double-length operand is formed from the Accumulator (most significant half) and an operand defined by R (least significant half). If R=0,2 the Accumulator (m.s) will be shifted double-length with the internal Operand Register (l.s). The contents of the Operand Register are dependent upon the previous instruction.

J=0,1 Arithmetic

S=0, Right arithmetic (SRA)
(sign digit regeneration)



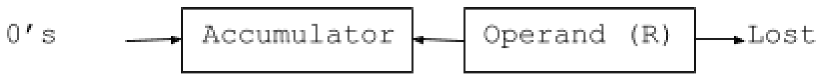
S=1, Left arithmetic (SLA)



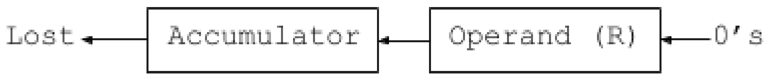
The V and S staticisers are set to the required state. C is unchanged and Z is meaningless.

J=2,3 Logical

S=0, Right Logical (SRL)

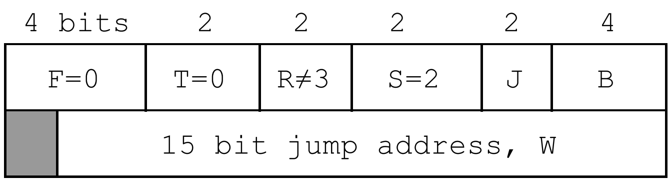


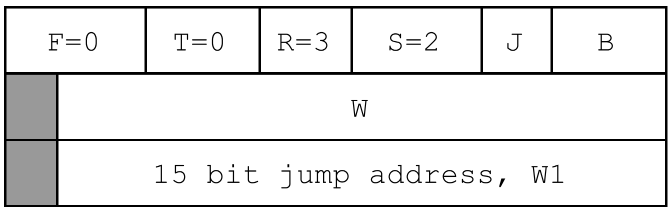
S=1, Left arithmetic (SLL)



The Z, V and S staticisers are meaningless after this operation. C is unchanged.

1.5.3.1.2 F=0; T=0; S=2; Jumps (JBC, JBS, JCS, JSC)





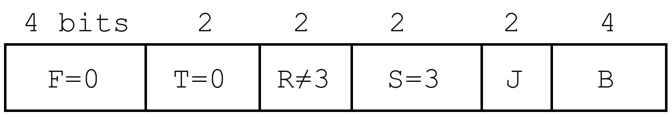
The B field defines the bit position within the operand (defined by R) to be tested :-

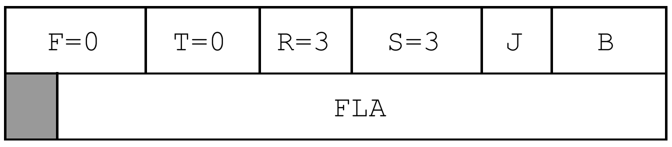
J=0	Jump if bit B clear	(JBC)
J=1	Jump if bit B set	(JBS)
J=2	Jump if bit B clear and then set it	(JCS)
J=3	Jump if bit B set and then clear it	(JSC)

For R≠3, the jump address is provided by the next location after the F=0 instruction.

For R=3, the operand's memory address is provided by the next location, and the jump address is provided by the subsequent location.

1.5.3.1.3 F=0; T=0; S=3; Bit Manipulation (SET, CLR)





The B field defines the bit within the operand (defined by R) to be manipulated:

J=0,1	Not to be used	
J=2	Set bit B	(SET)
J=3	Clear bit B	(CLR)

1.5.3.2 F=0; T≠0; Halt, External Functions.

1.5.3.2.1 T=1; Halt (HLT)

4 bits	2	10
F=0	T=1	Halt Number

Upon execution of this instruction the F100-L will not continue executing any further instructions, until the $\overline{Run}$ line is taken to the high (Stop) level and then returned to the low level. The least significant 10 bits of the instruction are ignored by the F100-L and can be used as required by the programmer (e.g. to provide a Halt Number).

1.5.3.2.2 T=2,3; External Function

	T	
4 bits	1	1 : 10
F=0	1	11 bit External Function Identifier

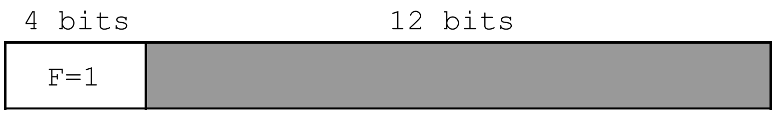
For T=2 or 3 the least significant bit of the T field can be set or clear, hence it can be combined with the remaining 10 bits of the instruction to form an 11-bit External Function identifier.

Upon decoding an External Function, the F100-L suspends operation. Control is passed to an external unit (Special Processing Unit, SPU). The SPU can then perform the function defined by the 11-bit Identifier. At the appropriate time the SPU signals the F100-L to continue processing.

A time-out is provided to detect the failure of an SPU.

For further details on SPUs see Section 1.12.

1.5.4 F=1; Switch Jump. PC = PC+A (SJM)



The contents of the Accumulator are added to the contents of the Program Counter. During an instruction the Program Counter always holds the address of the next instruction to be obeyed. Thus at the start of this instruction the Program Counter contains the address of the location following the SJM instruction. If the value of the Accumulator is zero then the program will continue in sequence.

If the sum of the Program Counter value and the contents of the accumulator exceed the address range of the F100 (i.e. > 32K) then only the least significant 15 bits of the address are used.

The Condition Register is unaltered by this instruction.

Bits 0-11 of this instruction have no significance.

1.5.5 F=2; Jump to N, Store Link in Link Stack (CAL)

NOTE: PC is the address of the F=2 instruction.

F=2	I=0	N≠0

CAL N

Store PC+1 and the Condition Register in locations X+1 and X+2 where X = the initial contents of location 0. Then load the program counter with N. Location 0 will contain X+2 at the end of the instruction. The initial value of X must be an odd number.

F=2	I=1	▨	P=0
▨	W, 15 bit subroutine start address		

CAL .W

Store PC+2 and the Condition Register in locations X+1 and X+2 where X = the initial contents of location 0. Load the program counter with the specified subroutine start address. Location 0 will contain X+2 at the end of the instruction. The initial value of x must be an odd number.

F=2	I=1	▨	R	P≠0

CAL /P

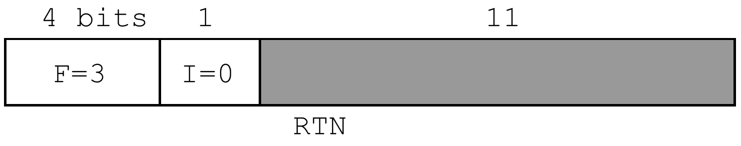
Store PC+1 and the Condition Register in locations X+1 and X+2, where X = the initial contents of location 0. Ignore the R field and load the program counter with the contents of location P. Location 0 will contain X+2 at the end of the instruction. The initial value of X must be an odd number.

F=2	I=0	N=0
D, (obeyed as an instruction, see below)		

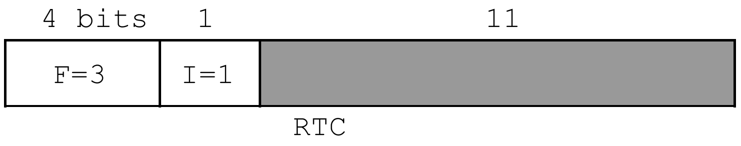
CAL ,D

Store PC+1 and the Condition Register in locations X+1 and X+2, where X = the initial contents of location 0. Then load the Program counter with PC+1. The initial value of X must be an odd number. This instruction only stores the link, it does not cause a program branch since it effectively loads the Program Counter with the value it already contains. Hence D is obeyed as the next instruction. Location 0 will contain X+2 at the end of this instruction. The multi-length staticiser in the Condition Register is cleared. The remaining bits of the Condition Register are not altered by this instruction. The Condition Register bits 0 to 6 are stored in bits 0-6 of location X+2.

1.5.6 F=3; I=0,1; Return (RTN, RTC)



If X is the value of the Link Stack Pointer in Location 0 at the start of the instruction, the contents of Location X-1 are placed in the Program Counter. Bits 0 to 5 of the contents of Location X are placed in bits 0 to 5 of the Condition Register. Bit 6 (the Fail Staticiser) of the Condition Register is not overwritten. The final value of the Link Stack Pointer will be X-2.



If X is the value of the Link Stack Pointer in Location 0 at the start of the instruction, the contents of Location X-1 are placed in the Program Counter. Bits 0 to 6 of the contents of location X are NOT placed into the Condition Register. The whole of the Condition Register is, therefore, not overwritten. The final value of the Link Pointer will be X-2.

The initial value of the Link Stack Pointer must always be an odd number.

1.5.7 F=4; (N) = A (STO)

F=4	I=0	N≠0

STO N

Store the contents of the accumulator in Location N.

F=4	I=0	N=0
16 bit data D		

STO ,D

Store the contents of the accumulator in the Location following the F=4 instruction.

F=4	I=1	▨	R	P≠0

STO /P
STO /P+
STO /P-

Index the contents of Location P as specified by the value of R.

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed

Use the appropriate value of (P) as an address and store the contents of the accumulator in that address.

F=4	I=1	▨	P=0
▨	15 bit address, W		

STO .W

Store the contents of the accumulator in location W

The S and Z condition staticisers are set to the required state. V is cleared.

1.5.8 F=5; M=0; (N)=(N)+A (ADS)

F=5	I=0	N≠0

ADS N

Add the contents of the accumulator to the contents of Location N and store the result in Location N.

F=5	I=0	N=0
16 bit data D		

ADS ,D

Add the contents of the accumulator to the contents of the location following the F=5 instruction. Store the result in the location following the F=5 instruction.

F=5	I=1	▨	R	P≠0

ADS /P
ADS /P+
ADS /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed.

Use the appropriate value of the contents of Location P as an address (P). Add the contents of this address to the contents of the accumulator and store the result at address (P).

F=5	I=1	▨	P=0
▨	15 bit address, W		

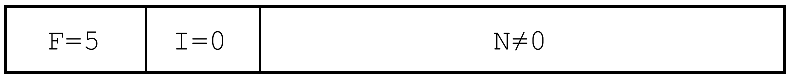
ADS .W

Add the contents of the accumulator to the contents of location W. Store the result in location W.

The C, S, V & Z condition staticisers are set to the required state.

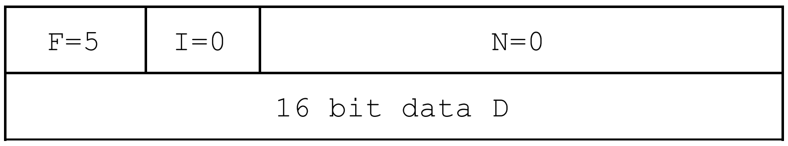
1.5.9 F=5; M=1; (N)=(N)+A+C (ADS)

In this instruction C has the same bit significance as the least significant bit of a word (i.e. bit (0))



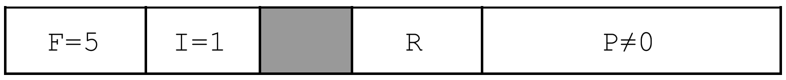
ADS N

Add C into the least significant end of the accumulator and add the result to the contents of the Location N and store the overall result in Location N.



ADS ,D

Add C into the least significant end of the accumulator and add the result to the contents of the location following the F=5 instruction. Store the overall result in the location following the F=5 instruction.

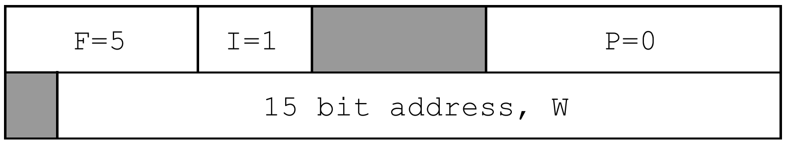


ADS /P
ADS /P+
ADS /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed.

Use the appropriate value of the contents of Location P as an address (P). Add the contents of this address to the contents of the accumulator, add C into the least significant end and store the overall result at address (P).



ADS .W

Add C into the least significant end of the accumulator and add the result to the contents of location W. Store the overall result in location W.

The C, S, V & Z condition staticisers are set to the required state.

1.5.10 F=6; M=0; (N)=(N)-A (SBS)

F=6	I=0	N≠0

SBS N

Subtract the contents of the accumulator from the contents of Location N and store the result in Location N.

F=6	I=0	N=0
16 bit data D		

SBS ,D

Subtract the contents of the accumulator from the contents of the location following the F=6 instruction. Store the result in the location following the F=6 instruction.

F=6	I=1	▨	R	P≠0

SBS /P
SBS /P+
SBS /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed.

Use the appropriate value of the contents of Location P as an address (P). Subtract the contents of the accumulator from the contents of address (P) and store the result in address (P).

F=6	I=1	▨	P=0
▨	15 bit address, W		

SBS .W

Subtract the contents of the accumulator from the contents of location W. Store the result in location W.

The C, S, V & Z condition staticisers are set to the required state.

1.5.11 F=6; M=1; (N)=(N)-A+C-1 (Subtract with Carry) SBS

Multi-length subtract is performed by including the carry from a previous subtract in at the least significant end (i.e. bit (0)) of the current instruction, in accordance with the above equation.

F=6	I=0	N≠0

SBS N

Perform multi-length subtract between the contents of the accumulator and the contents of Location N and store the result in Location N.

F=6	I=0	N=0
16 bit data D		

SBS ,D

Perform multi-length subtract between the contents of the accumulator and the contents of the location following the F=6 instruction. Store the result in the location following the F=6 instruction.

F=6	I=1	▨	R	P≠0

SBS /P
SBS /P+
SBS /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed.

Use the appropriate value of the contents of Location P as an address (P). Perform multi-length subtract between the contents of the accumulator and the contents of address (P). Store the result in address (P).

F=6	I=1	▨	P=0
▨	15 bit address, W		

SBS .W

Perform multi-length subtract between the contents of the accumulator and the contents of location W. Store the result in location W.

The C, S, V & Z condition staticisers are set to the required state.

1.5.12 F=7, (N)=(N)+1; Jump to (PC) if (N) Non-Zero (ICZ)

F=7	I=0	N≠0
▨	15 bit jump address, W	

```
ICZ N W
```

Increment the contents of Location N. Jump to the address specified if the result Non-zero.

F=7	I=0	N=0
16 bit counter, D		
▨	15 bit jump address, W	

```
ICZ ,D W
```

Increment the 16 bit counter. Jump to the address specified if the result is Non-zero.

F=7	I=1	▨	R	P≠0
▨	15 bit jump address			

```
ICZ /P W
ICZ /P+ W
ICZ /P- W
```

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed.

Use the appropriate value of the contents of Location P as an address. Increment the contents of this address and if the result is non-zero jump to the specified jump address.

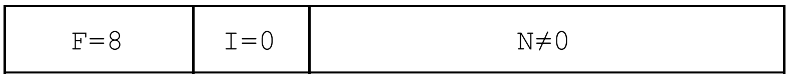
F=7	I=1	▨	P=0
▨	W		
▨	15 bit jump address		

```
ICZ .W W1
```

Increment the contents of location W. Jump to the address specified if the result is non-zero.

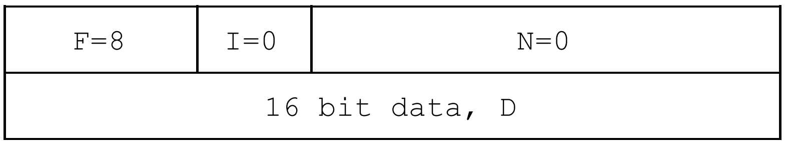
The Condition Register is unchanged is unchanged by this instruction.

1.5.13 F=8; A = (N) (LDA)



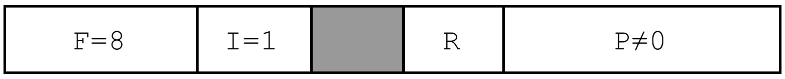
LDA N

Load the accumulator with the contents of Location N.



LDA ,D

Load the accumulator with the contents of the location following the F=8 instruction.

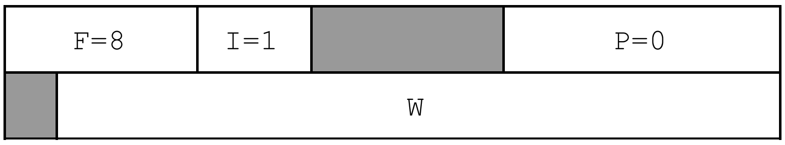


LDA /P
LDA /P+
LDA /P-

Index the contents of Location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Load the accumulator with the contents of the location whose address is the appropriate value of (P).

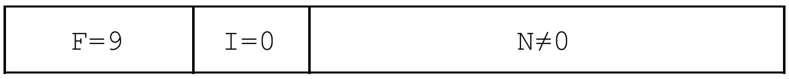


LDA .W

Load the accumulator with the contents of location W.

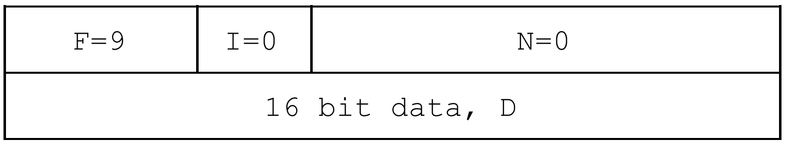
The S and Z condition staticisers are set to the required state. V is cleared.

1.5.14 F=9; M=0; A= (N)+A (ADD)



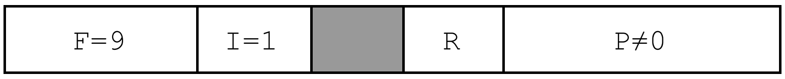
ADD N

Add the contents of the accumulator to the contents of Location N & store the result in the accumulator.



ADD ,D

Add the contents of the accumulator to the contents of the location following the F=9 instruction and store the result in the accumulator.

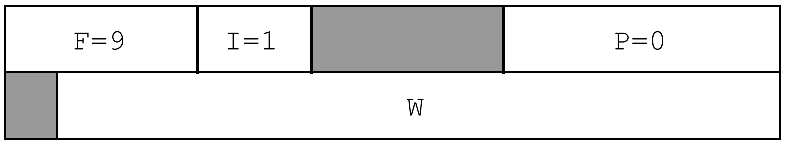


ADD /P
ADD /P+
ADD /P-

Index the contents of Location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Add the contents of the accumulator to the contents of the location whose address is the appropriate value of (P) and store the result in the accumulator.



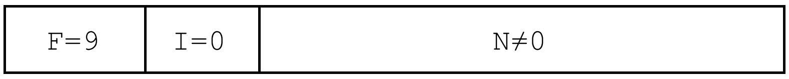
ADD .W

Add the contents of the accumulator to the contents of location W. Store the result in the accumulator.

The C, S, V & Z condition staticisers are set to the required state.

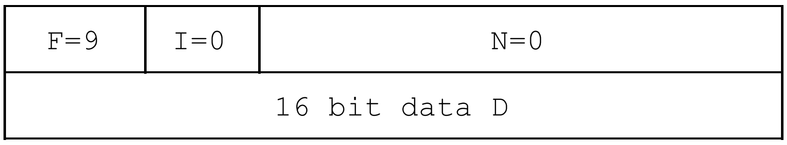
1.5.15 F=9; M=1; A=(N)+A+C (ADD)

In this instruction C has the same bit significance as the least significant bit of a word (i.e. bit (0))



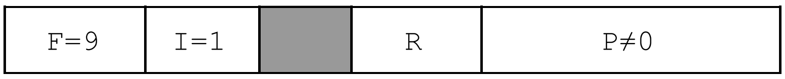
ADD N

Add C into the least significant end of the accumulator and add the result to the contents of the Location N and store the overall result in the accumulator.



ADD ,D

Add C into the least significant end of the accumulator and add the result to the contents of the location following the F=9 instruction and store the overall result in the accumulator.

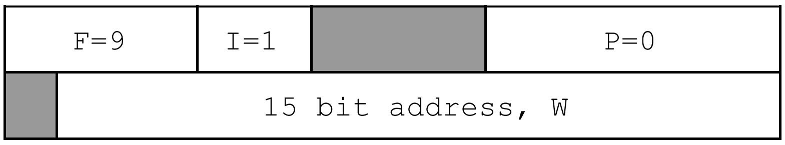


ADD /P
ADD /P+
ADD /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed.

Add the contents of the accumulator to the contents of the location whose address is the appropriate value of (P), add C into the least significant end and store the result in the accumulator.



ADD .W

Add C into the least significant end of the accumulator and add the result to the contents of location W. Store the overall result in the accumulator.

The C, S, V & Z condition staticisers are set to the required state.

1.5.16 F=10; M=0; A=(N)-A (SUB)

F=10	I=0	N≠0

SUB N

Subtract the contents of the accumulator from the contents of Location N and store the result in the accumulator.

F=10	I=0	N=0
16 bit data, D		

SUB ,D

Subtract the contents of the accumulator from the contents of the location following the F=10 instruction and store the result in the accumulator.

F=10	I=1	▨	R	P≠0

SUB /P
SUB /P+
SUB /P-

Index the contents of Location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Subtract the contents of the accumulator from the contents of the location whose address is the appropriate value of (P) and store the result in the accumulator.

F=10	I=1	▨	P=0
▨	W		

SUB .W

Subtract the contents of the accumulator from the contents of location W, Store the result in the accumulator.

The C, S, V & Z condition staticisers are set to the required state.

1.5.17 F=10; M=1; A= (N)-A+C-1 (Subtract with Carry) (SUB)

Multi-length subtract is performed by including the the carry from a previous subtract in at the least significant end (i.e. bit (0)) of the current instruction, in accordance with the above equation.

F=10	I=0	N≠0

SUB N

Perform multi-length subtract between the contents of the accumulator and the contents of location N and store the result in the accumulator.

F=10	I=0	N=0
16 bit data, D		

SUB ,D

Perform multi-length subtract between the contents of the accumulator and the contents of the location following the F=10 instruction. Store the result in the accumulator.

F=10	I=1	▨	R	P≠0

SUB /P
SUB /P+
SUB /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Use the appropriate value of the contents of location P as an address (P). Perform multi-length subtract between the contents of the accumulator and the contents of address (P). Store the result in the accumulator.

F=10	I=1	▨	P=0
▨	W		

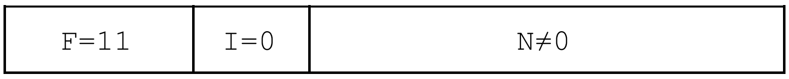
SUB .W

Perform multi-length subtract between the contents of the accumulator and the contents of location W. Store the result in the accumulator.

The C, S, V & Z condition staticisers are set to the required state.

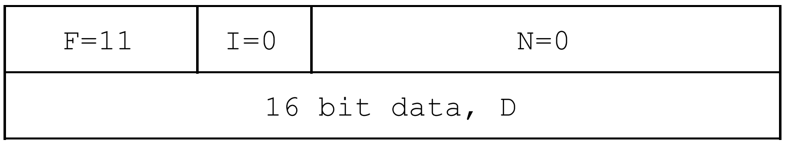
1.5.18 F=11: M=0; (N)-A; set condition staticisers (CMP)

Note: This instruction allows the value of the accumulator to be compared with the value of a specified location by subtracting the contents of the accumulator from the contents of the location. Neither the accumulator nor the location are overwritten, in fact the result is lost, but the appropriate Condition Register staticisers are set and inspection of these will provide comparative information (e.g. ≥,=,<).



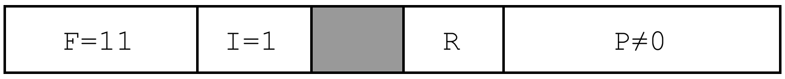
CMP N

Subtract the contents of the accumulator from the contents of location N.



CMP ,D

Subtract the contents of the accumulator from the contents of the location following the F=11 instruction.

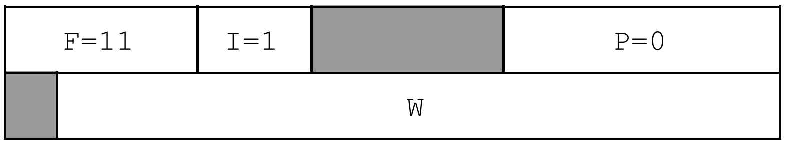


CMP /P
CMP /P+
CMP /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Subtract the contents of the accumulator from the contents of the location whose address is the appropriate value of (P).



CMP .W

Subtract the contents of the accumulator from the contents of location W.

The C, S, V & Z condition staticisers are set to the required state.

1.5.19 F=11; M=1; (N)-A+C-1, set condition staticisers (compare with Carry) (CMP)

Note: This instruction enables two multi-length operands to be compared. A sequence of these instructions perform a multi-length subtraction without overwriting any of the operands, in fact the results are lost. Only the appropriate Condition Register staticisers are set, and subsequent inspection of these will provide comparative information (e.g. ≥,=,<).

Multi-length compare is performed by including the carry from a previous compare in at the least significant end (i.e. bit (0)) of the current instruction, in accordance with the above equation.

F=11	I=0	N≠0

CMP N

Perform multi-length subtract between the contents of the accumulator and the contents of location N.

F=11	I=0	N=0
16 bit data, D		

CMP ,D

Perform multi-length subtract between the contents of the accumulator and the contents of the location following the F=11 instruction.

F=11	I=1	▨	R	P≠0

CMP /P
CMP /P+
CMP /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used
R=3	Decrement (P) after the store location has been accessed.

Use the appropriate value of the contents of location P as an address (P). Perform multi-length subtract between the contents of the accumulator and the contents of address(P).

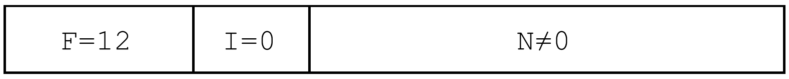
F=11	I=1	▨	P=0
▨	W		

CMP .W

Perform multi-length subtract between the contents of the accumulator and the contents of location W.

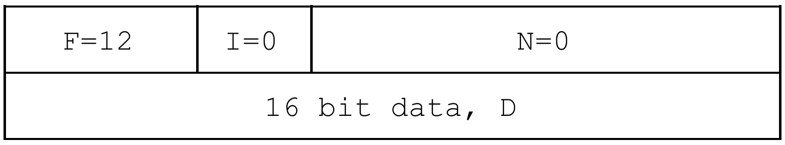
The C, S, V & Z condition staticisers are set to the required state.

1.5.20 F=12; A=(N)&A (AND)



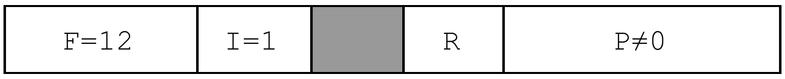
AND N

Perform a logical AND function between the contents of the accumulator and the contents of Location N. Store the result in the accumulator.



AND ,D

Perform a logical AND function between the contents of the accumulator and the contents of the location following the F=12 instruction. Store the result in the accumulator.

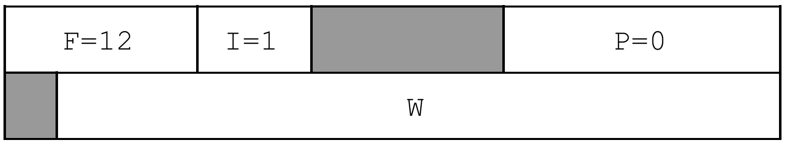


AND /P
AND /P+
AND /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Perform a logical AND function between the contents of the accumulator and the contents of the location whose address is the appropriate value of (P). Store the result in the accumulator.



AND .W

Perform a logical AND function between the contents of the accumulator and the contents of address W. Store the result in the accumulator.

The S and Z condition staticisers are set to the required state. The C staticiser will be set. The V staticiser value will be meaningless.

1.5.21 F=13; A=(N)≠A (NEQ)

F=13	I=0	N≠0

NEQ N

Perform a logical NON-EQUIVALENCE (Exclusive-OR) function between the contents of the accumulator and the contents of Location N. Store the result in the accumulator.

F=13	I=0	N=0
16 bit data, D		

NEQ ,D

Perform a logical NON-EQUIVALENCE (Exclusive-OR) function between the contents of the accumulator and the contents of the location following the F=13 instruction. Store the result in the accumulator.

F=13	I=1	▨	R	P≠0

NEQ /P
NEQ /P+
NEQ /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Perform a logical NON-EQUIVALENCE (Exclusive-OR) function between the contents of the accumulator and the contents of the location whose address is the appropriate value of (P). Store the result in the accumulator.

F=13	I=1	▨	P=0
▨	W		

NEQ .W

Perform a logical NON-EQUIVALENCE (Exclusive-OR) function between the contents of the accumulator and the contents of address W. Store the result in the accumulator.

The S and Z condition staticisers are set to the required state. The C staticiser will be cleared. The V staticiser value will be meaningless.

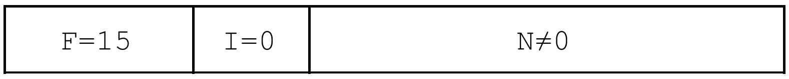
1.5.22 F=14. Not Allocated

Although this function code is not allocated, the sequence generated by this code is defined.

(The value of the Accumulator is altered by this instruction)

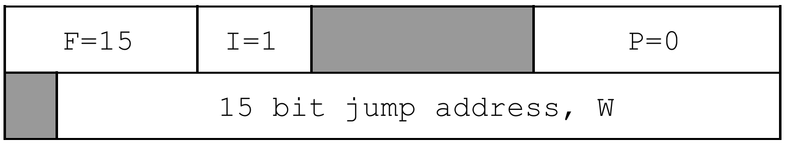
This instruction should not be used, as further members of the F100-L chip set may use the decoding.

1.5.23 F=15; Unconditional Jump (JMP)



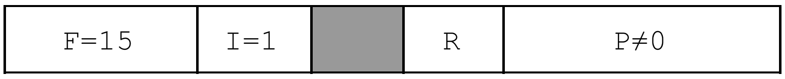
JMP N

Load the Program Counter with N.



JMP .W

Load the Program Counter with the specified jump address.

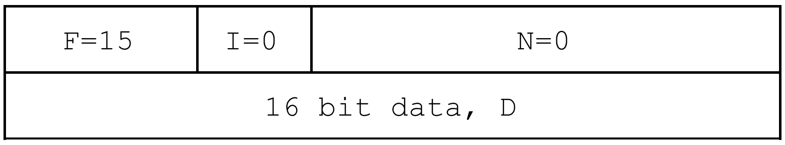


JMP /P
JMP /P+
JMP /P-

Index the contents of location P as specified by the value of R:

R=0,2	No change
R=1	Increment (P) before the address is used.
R=3	Decrement (P) after the store location has been accessed.

Load the Program Counter with the appropriate contents of location P.



JMP ,D

Load the Program Counter with the address of the next location in memory following this instruction. This instruction causes no branching and can be used as a NULL instruction. The 16-bit word, D will be obeyed as the next instruction.

The Condition Register is not altered by this instruction.

1.6 CONDITION REGISTER

The Condition Register contains the following items:

Bit (0)	Program Interrupt Lock-out	I
Bit (1)	Function Zero	Z
Bit (2)	Overflow	V
Bit (3)	Negative Sign	S
Bit (4)	Carry	C
Bit (5)	Multi Length	M
Bit (6)	Fail	F

All the condition Staticisers can be set, cleared or tested by program. The Condition Register can be cleared by a 7-place logical left or right shift (see Sect. 1.5.2).

The I, M & F stats are cleared by reset.

Upon acceptance of a Program Interrupt or by means of an F=2, Jump Store Link (CAL) instruction, the current state of the Condition Register is stored in the Link Stack, and M is then cleared. A Return (RTN) instruction causes the Condition Register to be loaded with the value given in the Link Stack, (with the exception of F which is not overwritten). The circumstances causing the setting/clearing of the individual Condition Staticisers in addition to the normal bit operation instructions are listed below.

1.6.1 Interrupt Lock-out (I)

The Interrupt Lock-out staticiser is automatically set upon acceptance of a program interrupt. It is cleared by a general CPU Reset.

1.6.2 Function Zero (Z)

Function Zero indicates the relevant state of the result of the instructions F = 4-6, 8-13 inclusive. If the result is "all 0's" the Function Zero staticiser is set. If the result is no "all 0's" then the Function Zero staticiser is cleared. This staticiser is meaningless after shift instructions.

1.6.3 Overflow (V)

Overflow indicates the relevant state of the result of instructions F = 5, 6 and 9 to 11 inclusive, or F=0 shifts. The occurrence of overflow conditions cause Overflow to be set. If overflow does not occur Overflow will be cleared. The conditions for setting Overflow are:-

(i) The addition of two numbers of the same sign, when the result is of a different sign.

(ii) The subtraction of two numbers of different sign, when the result is of the same sign as the sub tractor.

(iii) If the sign digit changes temporarily or permanently when a number is shifted left.

(iv) Overflow is meaningless after logical shifts, and F=12 or 13 instructions.

1.6.4 Negative Sign (S)

The Negative Sign staticiser indicates the relevant state of the result of instructions F=0, shifts, and F=4-6, 8-11 inclusive. The Negative Sign staticiser will be set to the same state as the resultant sign digit.

1.6.5 Carry (C)

The Carry staticiser indicates the relevant state of the result of instructions F=5,6 and 9 to 11 inclusive. Carry will be set if there is a carry out of the most significant bits of the arithmetic function. If no carry is generated the Carry staticiser will be cleared.

1.6.6 Multi-Length (M)

When set, the value of Carry is added into the least significant bit of an arithmetic function. It is also used to define the T=0; S=0,1 double length shift instructions (see Sect. 1.5.2)

The staticiser is reset by a general CPU Reset, F=2 (CAL) instructions and a Program Interrupt.

1.6.7 Fail (F)

If an External Function, DMA, or an I/O Bus cycle is not completed within an externally defined time the Pail staticiser is set.

This staticiser is reset by a general CPU Reset.

1.6.7.1 FAILURE INDICATION

The setting of the Fail staticiser will cause the output signal $\overline{Fail}$ to go low (pin 35).

1.7 THE INPUT/OUTPUT BUS

All transfers to and from the processor use the I/O Bus This is a bidirectional 16-bit highway and transfers both addresses and data.

The control of the I/O Bus is via “handshake” signals so that there is no synchronous timing.

During the first phase of an I/O Bus cycle the highway carries the address of the location/device involved, and the type of cycle required (i.e. read, write, read-modify-write). Each device must recognise its address and complete a "handshake".

During the second phase the data concerned is transferred in the direction required.

As part of system integrity considerations the I/O Bus "handshake" signals are subject to a time-out. Detection in the processor of the failure of the "handshake", i.e. the time-out period has been exceeded, will cause the Fail Condition Staticiser to be set, and the $\overline{Fail}$ output to go low.

All memory and I/O devices are bussed onto this highway.

To enable these devices to be interfaced to the I/O Bus without the need for large amounts of TTL circuitry that is often required by other microprocessors Ferranti has introduced the Interface Set. A description of the facilities provided by the Interface Set is given in Section 2.

Since the Interface Set is used whenever it is necessary to make a connection onto the I/O Bus, the user is removed from directly handling the I/O Bus signals. This has enabled Ferranti to introduce the I/O Bus with a minimal number of lines, made possible by multiplexing both addresses and data onto a single set of 16 bidirectional highway lines, and without increasing the complexity of the Bus from the user point of view.

The small number of lines used by the I/O Bus enables systems to be interconnected in a simple, straightforward manner by reducing the complexity of printed circuit board layouts and minimising the number of edge-connections required.

The following paragraphs describe the signals and signal transitions used during I/O Bus cycles.

A detailed specification of the I/O Bus is given in the F100-L Hardware and System Manual, HSM 150.

1.7.1 LINES ACROSS THE I/O BUS

The I/O Bus consists of 21 lines; listed below.

$\overline{Hw(0)} - \overline{Hw(15)}$	the 16 bidirectional lines which carry addresses and data in inverse form i.e. when the data is a logic '1' the line will be low;
$\overline{J(Acv)}$, K(Pas)	the pair of handshake signals which control the timing of address and data transfers to the passive device (i.e. the device on the receiving end of the transfer)
$\overline{J(Pas)}$, $\overline{K(Acv)}$	the pair of handshake signals which control the timing of data transfers to the active device. (i.e. the device initiating the transfer).
$\overline{Wt}$	the signal that identifies a Write cycle. (A Read signal is also provided, but it is multiplexed onto the $\overline{Hw(15)}$ line during an address transfer).

The transitions of these signals which provide the three possible types of Bus cycle are illustrated in Fig.1.3.

1.7.2 Read Only Cycles

Edge A	indicates that the active device has placed an address on bits 0-14 of the highway, the fact that a Read Only cycle is requested is indicated by ensuring that $\overline{Hw(15)}$ is low and $\overline{Wt}$ is high.
Edge B	indicates that the relevant passive device has accepted the address and started a cycle.
Edge C	indicates that the active device has cleared the address from the highway and is now awaiting data from the passive device.
Edge D	has no positive indication for Read Only cycles, it must occur between C and F.
Edge E	indicates that the passive device has placed the data read from the addressed location onto the highway.
Edge F	indicates that the active device has accepted the data.
Edge G	indicates that the passive device has removed the data from the highway.
Edge H	indicates that the interface cycle is complete and a further request may be made; H will normally follow G with little or no delay

1.7.3 Write Only Cycles

Edge A	indicates that the active device has placed address on bits 0-14 of the highway, the fact that a Write Only cycle is requested is indicated by ensuring that $\overline{Hw(15)}$ is high and $\overline{Wt}$ is low.
Edge B	as for Read Only.
Edge L	indicates that the active device has placed the data to be written to the passive device on the highway.
Edge M	indicates that the passive device has accepted the data.

1.7.4 Read/Modify/Write Cycles

Edge A	indicates that the active device has placed an address on bits 0-14 of the highway, the fact that a Read/Modify/Write cycle is requested is indicated by ensuring that $\overline{Hw(15)}$ and $\overline{Wt}$ are both low.
Edge B	as for Read Only.
Edge C	as for Read Only.
Edge E	as for Read Only.
Edge 7	as for Read Only.
Edge G	as for Read Only.
Edge L	as for Read Only.
Edge M	as for Read Only.

1.7.5 Timing and Transfer Failures

It is inherent in the nature of this hand shake controlled 170 Bus that there are no timing rules for data transien Each device will wait for the other to respond. However, it is obvious that, under fault conditions, a device may never respond. To overcome this situation both the F100-L and the Interface Set contain time-out circuits that will detect when a predefined time period has been exceeded. The

I/O Bus cycle in progress at that time is cleared down and failure indication is given to the user.

1.8 F100-L INITIATED I/O BUS CYCLE TIMING

The F100-L uses three types of cycle across the I/0 Bus:

Read

Write

Read-Modify-Write

These cycles are used for transfers to/from memory or with programmed peripheral transfers where the peripheral device is allocated an address and is treated in exactly the same way as a memory location (memory-mapped I/0).

For Write cycles, the F100-L waits until the cycle is complete before continuing, so only the complete interfaced cycle time is relevant (Wc).

The term "Read-Modify-Write Cycle time" is used to define the time taken to perform the initial "read" and the subsequent "write" parts of the cycle. It does not include the time taken by the processor to perform the "modify" function. This "modify" time is quoted separately as a number of logic beats in the Instruction Summary, Table 1.1.

For Read cycles, the F100-L can sometimes, but not always, continue as soon as the data is acquired without waiting for the completion of the interface cycle. The time for such an action is, therefore, the Read Access time (Ra).

In situations where the F100-L has to wait for the complete interface cycle to finish, the Read Cycle time (Rc) is required.

In F100-L systems Read cycles may be performed upon either ROM or RAM storage devices. (e.g. instructions may be read from ROM, whilst data is read from RAM). To enable the user to derive instruction execution times for a particular POM/RAM or all-RAM system the Instruction Summary (Table 1.1) uses the abbreviations:

Ra_1 and Rc_1,	for program area accesses and cycles which may be ROM
Ra_2 and Rc_2,	for data area accesses and cycles which are typically RAM.

The following formulae give the basic I/O Bus cycle times required to calculate the Instruction Times listed in Table 1.1. The read access (Ra), read cycle (Rc), read/modify/write cycle (M), and write cycle (WC) times are calculated in terms of:

L	the Logic cycle time which is equal to twice the period of the square wave clock input to F100-L,
Ac	the device Access time seen on the device interface to an Interface Set,

Wt	the device Write time required on the device interface to an Interface Set

Read Access	Ra = (3L/2+277+Ac)*
Read Cycle	Rc = (3L/2+525+Ac)*
Write Cycle	Wc = (3L/2+182)* + (3L/2+243+Wt)*
Read-Modify-Write Cycle	M = (3L/2+277+Ac)* + (3L/2+243+Wt)*

* Contents of bracket rounded up to next multiple of L/2.

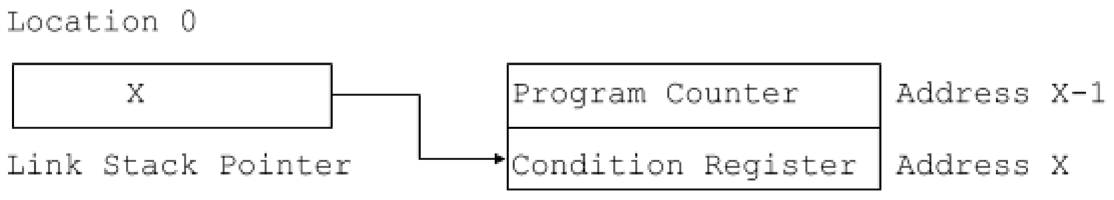
For more detailed timings (e.g. for configurations using a Buffer Interface Set and/or a Bus Extension Unit) see the F100-L Hardware and System Manual, HSM 150.

1.9 INTERRUPT SYSTEM

A Program Interrupt Request line is provided into the processor which may be used directly or by Interface Sets to enable many devices to interrupt the processor.

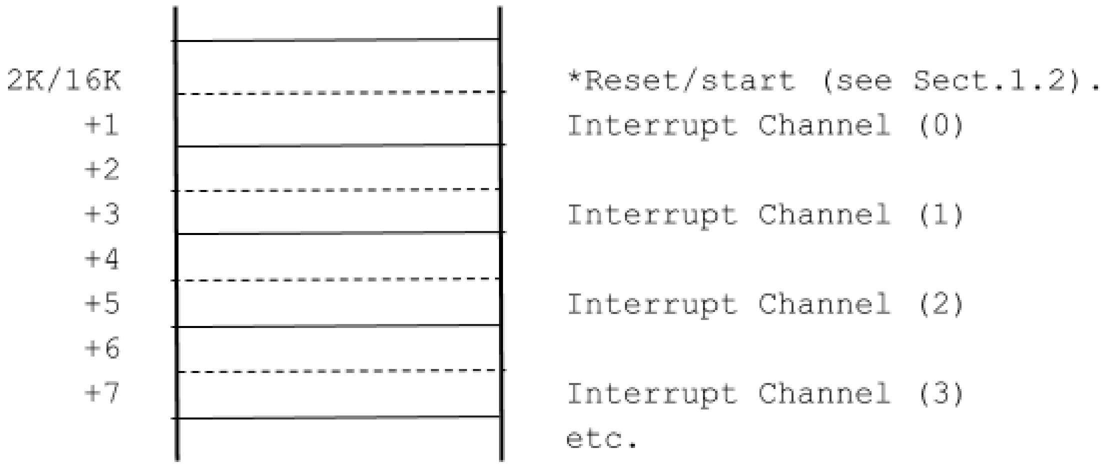
Program interrupts can be inhibited by the Interrupt Lockout staticiser, which is one of the Condition Register(Sect 1.6) bits.

With the Interrupt Lock-out staticiser clear, program interrupt requests are accepted at the end of the instruction being executed. The current state of the Program Counter and the Condition Register is stored in the Link Stack. The Link Stack is a push-down, pop-up stack resident in memory. A Link Stack Pointer is stored in Location '0' that gives the address last word of the last pair of words (Program Counter and Condition Register) to be stored in the Link Stack. i.e.



The initial value of the Link Stack Pointer must be an odd number. Having stored the Link, the Interrupt Lock-out staticiser is set.

As part of the interrupt sequence the Interface Set relevant to the interrupting device loads the Program Counter with the Interrupt Vector having the value 2K/16K + 2n, where n is the channel number of the interrupting device (0 ≤n≤ 63)*. A Jump Instruction (two words) to the start of that devices interrupt program should be placed in location 2K/16K + 2n, therefore providing a fully vectored interrupt system. i.e.



A Program Interrupt Accept line is used to form a "handshake” with the Program Interrupt Request line, and remains set from the acceptance of the program interrupt until the link storage and loading of the Program Counter is complete.

At the end of the interrupt program, control is returned to the main program by using the F=3 Return (RTN, RTC) instructions. (See Sect. 1.5.5)

* Care should be taken when using channel 0 (n=0) since the Interrupt Vector is the same as the Reset/Start address and the program must determine which event caused the branch (i.e. an interrupt or a reset) by inspecting the state of the Interrupt Lock-out staticiser (I) in the Condition Register. Reset always clears I. An interrupt always sets I.

1.10 DIRECT MEMORY ACCESS (DMA)

Whilst the I/O Bus is not being used by the F100-L it is possible for other devices on the I/O Bus to communicate directly amongst themselves by using the Direct Memory Access (DMA) facility.

A DMA Request input is provided by the F100-L. When the F100-L does not require to use the I/O Bus a DMA Accept signal is sent in response to the DMA Request enabling the requesting device to use the I/0 Bus.

The F100-L Interface Set enables up to 64 peripherals to use the DMA facility, providing the necessary control and priority multiplexing functions.

A 'daisy-chain' priority system is used with the DMA Request and Accept signals from a lower priority device passing through the higher priority devices (see Fig 1.1) The device electrically closest to the F100-L therefore, has the highest priority.

1.11 DMA CYCLE TIMING

Devices requiring DMA access to an F100-L I/O Bus are connected into the DMA daisy chain, so that F100-L handles only a single pair of DMA Request and Accept signals. The device electrically closest to F100-L is the highest priority and can ensure that it gains access to the Bus when the next DMA request is answered. Accesses of lower priority devices further down the daisy chain are subject to delays due to the activity of higher priority devices, as well as from F100-L itself.

Ignoring the activity of higher priority devices, which is obviously system dependen, each device connected into the daisy chain via an Interface Set adds an additional delay of approximately 75 nSecs to the response seen at a requesting device of lower priority.

F100-L will accept a DMA whenever it is not waiting to transmit or receive data over the Bus. In each instruction there is at least one such time - immediately after the instruction is accessed. The occurrence of other DMA accept periods is instruction dependent, and for some instructions no such periods exist. Thus a DMA can wait up to a full instruction period before being accepted. After accepting a DMA, F100-L will continue operating until it again requires to use the I/O Bus, at which time it stops processing and waits for the DMA to finish. If F100-L is ready to continue when the DMA is finished, it will immediately reclaim the Bus and any further requests will have to wait until the next acceptable time for DMAs to gain access.

The CPU time lost is, therefore, a function of the amount of overlapping of operation which can occur, and this is instruction dependent. The following figures for lost CPU time are worst case in the sense that they assume no overlapping. In a practical situation it is probably fair to assume an average overlap of one or two logic cycles.

The maximum CPU time lost is:

for a DMA Read	1004 + Ac + DevLd + 50N
for a DMA Write	744 + Wt + DevDa + 50N

where each time must be rounded to the next multiple of L,

Ac is the store access time,

Wt is the store write time,

DevLd is the device load time measured at the handshake signals between the relevant interface set and the device,

DevDa is the time taken for the device to present its data for writing away.

N is the channel number of the device in the DMA daisy chain (0 when adjacent to F100-L, 1 when next, 2 next and so on); the incremental delay is only 50 nSecs since the time taken by the Request signal (35 nSecs per device) to reach F100-L occurs before the lost CPU time and only affects the response time seen at the device.

L is the logic cycle time (twice the input clock period) of the F100-L controlling the I/O Bus.

Further more detailed information is given in the F100-L Hardware and System Manual, HSM 150.

1.12 SPECIAL PROCESSING UNITS

Special Processing Units (SPUS) are used in F100-L systems where a relatively complex function is required to be performed faster than is possible by program in the microprocessor itself (for example Fourier Transforms) and also to supply, where these are required, a hardware implementation of some of the simpler functions not present in the basic instruction set (for example sin/cos) The F100-L instruction set includes a number of External Functions (see Section 1.5.2.2.2) which are ignored by F100-L itself. Such instructions can be recognised and decoded by an Interface Set and used to trigger the associated SPU.

The communication between the SPU and the program can be handled in a number of ways, the selection of the appropriate variant depending upon the time taken by each SPU function. The choices are:

(a) Short Functions (say <20usecs) - the SPU will be triggered by an External Function instruction and will hold the program until the function is completed, so that the result can be assumed available for the next instruction in sequence.

(b) Medium Functions (say up to 100-200usecs) - total system efficiency is increased by allowing the processor to continue in program after it has triggered the SPU, completion of the function being indicated by the SPU setting a marker bit in store for testing by the program.

(c) Long Functions - the only efficient way to inform the program of completion of such functions is by program interrupt, in which case the SPU must be allocated a peripheral channel number and the option is also available to trigger the SPU as a peripheral instead of using an External Function.

In all cases the SPU accesses operands from and writes results to the store using the DMA mechanism.

Ferranti currently provides an SPU for Fast Fourier Transform (FFT) work, (See Sect. 3), and a single chip fast Multiply Divide SPU, the F101-L is currently under development.

1.13 FAILURE TIMEOUTS

Asynchronous control signals using a 'handshake' principle have to be protected by time-out circuitry to detect the failure of a response to a handshake signal.

Handshake signals are used to control the following facilities:

(a) I/O Bus

(b) DMA

(c) External Functions

Hence, all these signals may be subject to a time-out. If the time-out period is exceeded the Fail staticiser in the Condition Register is set along with the $\overline{Fail}$ output line.

One time-out period is shared by all the above facilities. The period may be user defined by adding the appropriate capacitor and resistor to the "CRExt" pin.

Time-outs can be inhibited by taking the "CRExt" pin to 0v. Once the Fail staticiser has been set the F100-L will try to continue program execution. Certain time-out failures can be catastrophic (e.g. I/O

Bus) and further operation may be impossible. Some failures (e.g. external functions) may well allow further operation but in a 'reduced-mode'.

1.14 CONTROL HANDSET (Fig. 1.4)

For normal system use the Control Handset will not be required. Essential control signals to the processor ($\overline{Reset}$, $\overline{Run}$) will be provided by the system.

For some applications (e.g. systems used for program writing and proving) a means of giving the operator detailed information about what is happening inside the processor is required. For this purpose a Control Handset is available.

The Control Handset contains control keys (e.g. Run, Reset) an octal number entry keyboard, and a 6 digit LED display.

The facilities can be summarised as follows:

(a) Normal control of the running of the program using the Reset, Run and Single Step keys

(b) In the Stop condition, loading of information from the keyboard into the PC or memory.

(c) Monitoring the current value of PC, the Instruction Register or a memory location.

(d) Program load from an I/0 device, using the keyboard to define the particular I/O device being used, and the ROM Loader on the Interface card.

(e) System state monitoring (e.g. Stop, Fail).

The Control and set is connected to the F100-L via an interface card. A 3 metre cable links the Control Handset to its interface card. The interface card plugs directly into an F100 Microcomputer system (see sect. 3).

1.15 CONNECTIONS TO F100-L

The functions of the connections to the F100-L chip are outlined below; as indicated, the Interface Set (See Section 2) is designed to handle all the I/O Bus and associated control signals. Mandatory connections are labelled M, optional connections are labelled O. A positive logic convention is used. A11 inputs should be TTL compatible, all outputs are at TTL levels. The connections are listed in numerical pin number sequence in Fig. 1.5. at the end of this section. Circuit parameters are given in Section 1.16.

Pin Number	Signal Name	M/O	Description
Power Supply F100-L requires an external regulator transistor of type Z190 or similar, connected as stated below.			
12	+5V	M	Minimum current drain of 300mA, should be connected to pin 12 of F100-L and the collector of the external transistor. Since there is a total of some 1.2W to be dissipated in this regulator it is recommended that +5V is connected to the collector via a 8.2ohm 11W resistor to reduce the dissipation in the transistor.
30	RefOut	M	Connect to the base of the external transistor.
13,31	RefIn	M	Connect both pins to the emitter of the external transistor. It is recommended that this line is decoupled to 0v with a 3.3uF capacitor.
32	0V	M	Should be decoupled from the +5V line via a 3.3 uF capacitor.
Basic Control			
5	Run	M	When taken to logical '0' will cause the processor to start obeying instructions from the store location specified by the program counter. When taken to a logical '1' the last instruction read from store is completed and F100-L stops. In systems with an automatic reset (see below) it may be connected permanently to OV.
1	$\overline{Rs}$	M	When taken to a logical '0' for a minimum of 20us will reset F100-L. Resetting F100-L clears the Program Interrupt Lock Out, Double Length and Fail staticisers in the Condition Register and

		O	loads the Program Counter with a specific value (see Ad Sel).
Time-out			
7	CRExt	M	Allows the failure time-out period for I/O Bus cycles DMA transfers, and External Functions to be altered to suit system requirements.
Clock			
34	Cklp	M	F100-L requires a square wave clock whose frequency is dependent on the specification against which the particular F100-L has been purchased.
Address Select			
11	AdSel	O	If this input is left unconnected the F100-L Program Counter will be loaded with 2048 when F100-L is reset. If it is tied to OV the Program Counter will be loaded with 16384 when F100-L is reset. It also affects the interrupt program start address. (see below).
Status Indication			
3	$\overline{Sp}$	O	When this signal is at a logical 'O' it indicates that F100-L has stopped.
35	$\overline{Fa}$	O	When this signal is at a logical 'O' it indicates that the Fail staticiser in the Condition Register has been set. This can be caused by: (a) A DMA timeout (b) An External Function timeout (c) An I/0 Bus timeout
The following signals are normally handled exclusively by the Interface Sets which connect devices to the I/O Bus. The Interface Set ensures correct operation of these signals.			
DMA			
36	$\overline{DMARq}$	O	Normally connected to a DMA daisy chain priority network. See Section 2.
37	$\overline{DMAAccept}$		
Program Interrupt			
10	$\overline{PgItRq}$	O	Used to control the Program Interrupt system. Normally connected to a Program Interrupt daisy chain priority network. See Section 2.
9	$\overline{PgItAcept}$		
I/O Bus Control			

2	$\overline{J(Pas)}$	M	See Section 1.7
40	$K(Pas)$		
39	$\overline{J(ACV)}$		
38	$\overline{K(ACV)}$		
8	$\overline{WtExt}$		
I/O Bus Data Highway			
14-29	$\overline{H0} - \overline{H15}$	M	See Section 1.7
Miscellaneous Control Signals			
33	$\overline{ExtLdPgCt}$	O	May be used by external devices to load the Program Counter.
4	$\overline{IRd}$	O	When this signal is at a logical '0' it indicates that F100-L is in the process of reading an instruction from memory.
6	$\overline{ExtFnAccept}$	O	Used in conjunction with External Functions and SPUs.

TABLE 1.1, PART 1: INSTRUCTION SUMMARY AND TIMING

Function Code	Instruction Summary	Execution Time	See Note
F=0	Shift Register* B places left or right (Arithmetic, logical, end around)	$1Ra_1$ + (B+3)L	(i)
	Shift Accumulator & Condition Register double-length B places left or right (Arithmetic or Logical)	$1Ra_1$ + (B+5)L	
	Shift Accumulator & Condition Register double-length B+16 places left or right (Arithmetic or Logical)	$1Ra_1$ + (B+21)L	
	Shift Accumulator & Store location double-length B places left or right (Arithmetic or Logical).	$1Ra_1$ + $1Rc_1$ + $1M_2$ + (B+3)L	
	Shift Accumulator & Store location double-length B+16 places left or right (Arithmetic or Logical).	$1Ra_1$ + $1Rc_1$ + $1M_2$ +(B+19)L	
	Jump if bit B of Register* set or clear.	$1Ra_1$ + $1Rc_1$ + 18L	(i)
	Jump if bit B of Register* set and then clear it	$1Ra_1$ + $1Rc_1$ + 18L	(i)
	Jump if bit B of Register* clear and then set it.	$1Ra_1$ + $1Rc_1$ + 18L	(i)
	Set or clear bit B of Register*	$1Ra_1$ + 18L	(i)
	External Function.	As required	
	Halt	$1Ra_1$ + 20L	

The above subdivisions of F = 0 are defined by the T, R, S and J fields of the instruction.

* "Register" can be the Accumulator, condition register or memory location.

(i) If 'Register' is a memory location add $1Rc_1$ + $1M_2$.
If 'Register' is the Condition Register add 2L.
If 'Register' is the Accumulator add 1L.

(ii) Ra_1 is the time for a program memory Read access (e.g. ROM)

(iii) Ra_2 is the time for a data memory Read access (e.g. RAM)

(iv) Rc is the time for a program memory Read cycle (e.g. ROM)

(v) Rc_2 is the time for a data memory Read cycle (e.g. RAM)

(vi) M is the time for a memory Read-Modify-Write cycle.

(vii) L is the time for a single logic cycle

(viii) If the Accumulator and the internal Operand Register are shifted double length (see Sect. 1.5.2) then the Execution Time will be 1L less than the equivalent shift of Accumulator and Condition Register.

TABLE 1.1, PART 2: INSTRUCTION SUMMARY AND TIMING

Function Code	Instruction Summary	Condition Staticisers Affected				Execution Time
		C	S	V	Z	
F=1	PC=PC + A, Switch Jump					1Ra1 + 20L
F=2	Jump, Store Link I=0					$1Ra_1$ + 1M + 2Wc + 10L
	Jump, Store Link I=1					$1Ra_1$ + $1Rc_1$ + 1M + 2Wc + 19L
F=3	Return (RIN) I=0	✓	✓	✓	✓	$1Ra_1$ + $2Rc_2$ + 1M + 18L
	Return (RTC) I=1					$1Ra_1$ + $1Rc_2$ + 1M + 18L
F=4	(N)=A		✓	0	✓	$1Ra_1$ + 1M + 18L
F=5	(N)=(N) + A	✓	✓	✓	✓	$1Ra_1$ + 1M + 18L
F=6	(N)=(N) - A	✓	✓	✓	✓	$1Ra_1$ + 1M + 18L
F=7	(N)=(N) + 1, Jump if (N)≠0					$1Ra_1$ + $1Rc_1$ + 1M + 18L
F=8	A=(N)		✓	0	✓	$1Ra_1$ + $1Ra_2$ + 18L
F=9	A=(N) + A	✓	✓	✓	✓	$1Ra_1$ + $1Ra_2$ + 18L
F=10	A=(N) - A	✓	✓	✓	✓	$1Ra_1$ + $1Ra_2$ + 18L
F=11	(N)-A Set Condition Stats only	✓	✓	✓	✓	$1Ra_1$ + $1Ra_2$ + 18L
F=12	A=(N) & A	1	✓	*	✓	$1Ra_1$ + $1Ra_2$ + 18L
F=13	A=(N) ≠ A	0	✓	*	✓	$1Ra_1$ + $1Ra_2$ + 18L
F=14	Not Allocated					
F=15	Jump to N, I=0					$1Ra_1$ + 3L
	Jump to N, I=1, P=0					$1Ra_1$ + $1Rc_1$ + 2L
	Jump to N, I=1, P≠0, R=0/2					$1Ra_1$ + 1M + 4L
	Jump to N, I=1, P≠0, R=1/3					$1Ra_1$ + 1M + 19L

Notes (i) Ra_1 is the time for a program memory Read access (e.g. ROM)
(ii) Ra_2 is the time for a data memory Read access (e.g. RAM)
(iii) Rc_1 is the time for a program memory Read cycle (e.g. ROM)
(iv) Rc_2 is the time for a data memory Read cycle (e.g. RAM)
(v) M is the time for a Read-Modify-Write cycle.
(vi) Wc is the time for a memory Write cycle
(vii) For F=4-13, pointer indirect addressing with auto-index adds 1M + 16L,
pointer indirect addressing without auto-index adds 1M + 1L,
immediate indirect addressing adds $1Rc_1$
(viii) ✓ indicates relevant state of result of function
* meaningless
1 set to 1
0 to 0

For other abbreviations see Sect. 1.5.1.

Parameter		Min.	Typ.	Max.	Units
Supply Voltage (Pin 12)		4.75	5	5.25	V
Low Level Output Current			6		mA
Low Level clock width			35		ns
Operating free-air temperature	Military	-55		+125	°C
	Industrial	-25		+85	°C
	Commercial	0		+70	°C

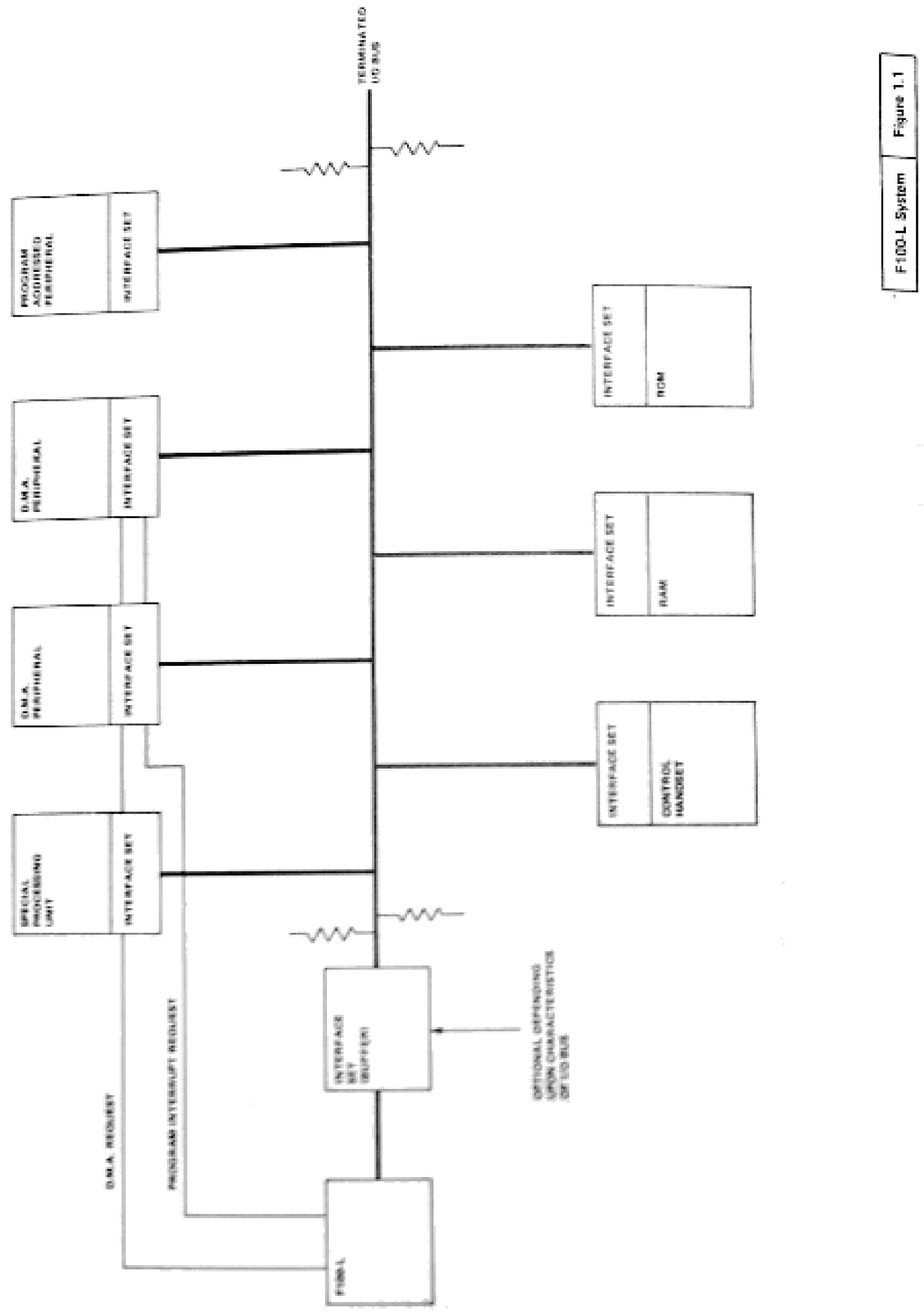
Table 1.2

Recommended Operating Conditions

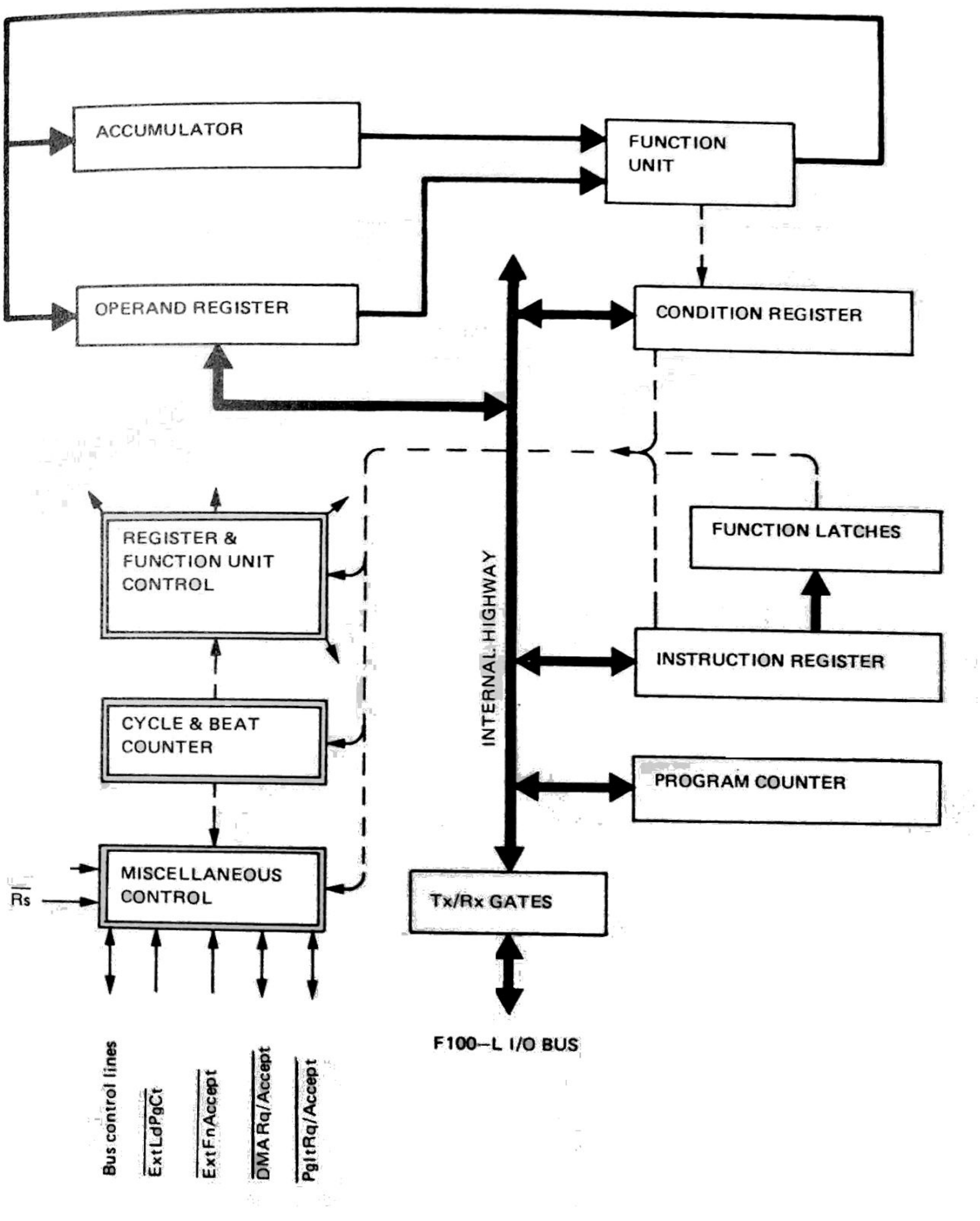
Parameter		Min.	Typ.	Max.	Units
High Level input voltage		1.6			V
Low Level input voltage				1.0	V
Low Level output voltage					V
High Level input current				(Vin-1.5) /5	mA
Low Level input current (Vin=0.4V, Vcc=5.0V)	No pull-up				uA
	15K pull-up		0.92		mA
Short circuit current			1		mA
Supply Current (Sum of 5V & REF In Currents)			270		mA
Voltage at REF In at	-55°C		1.2		V
	25°C		1.1		V
	125°C		0.95		V
Internal pull up resistor on all output pins			5		K

Table 1.3

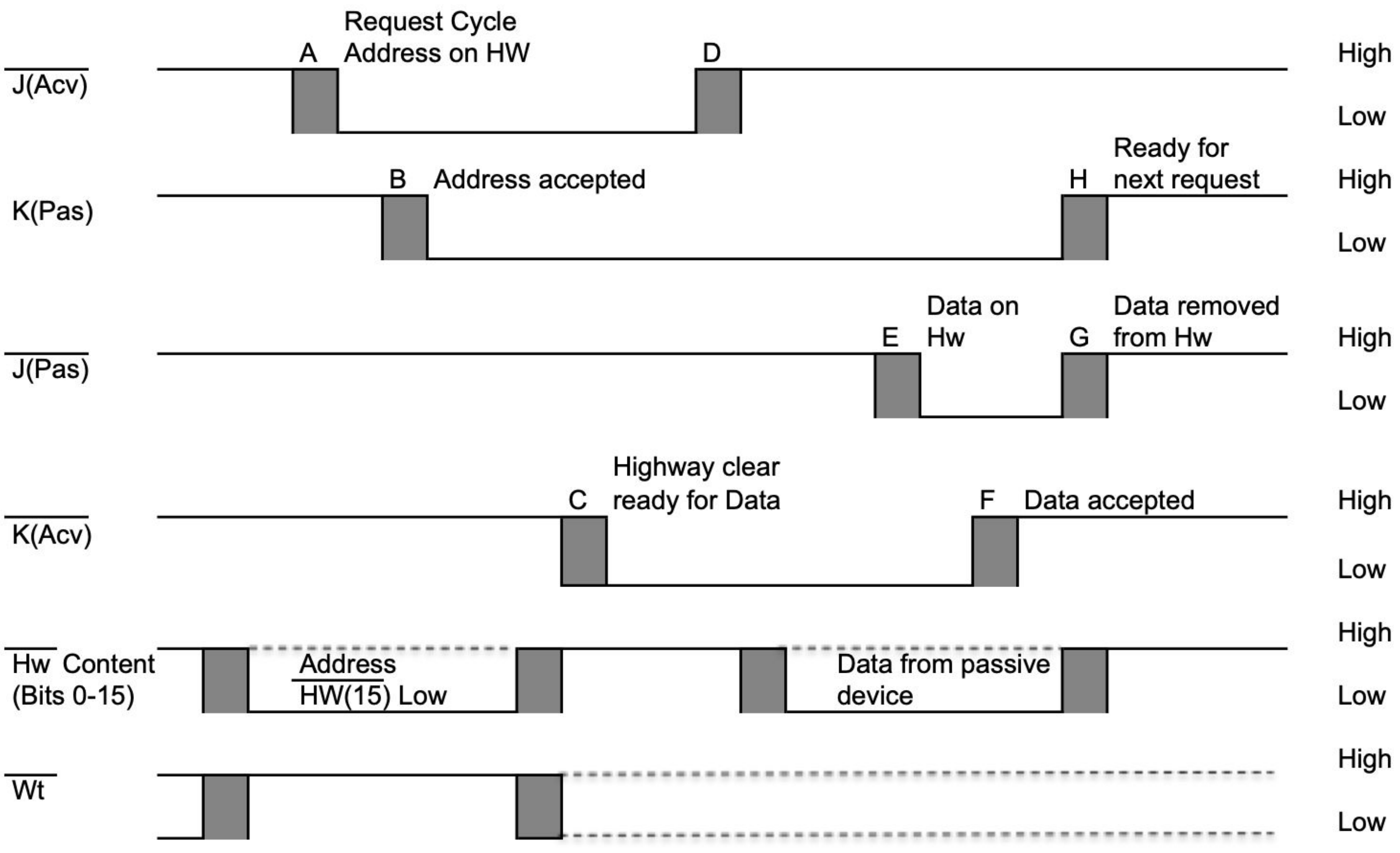
Electrical Characteristics



F100-L Diagram Fig. 1.1



F100-L, Block Schematic Fig. 1.2



Notes

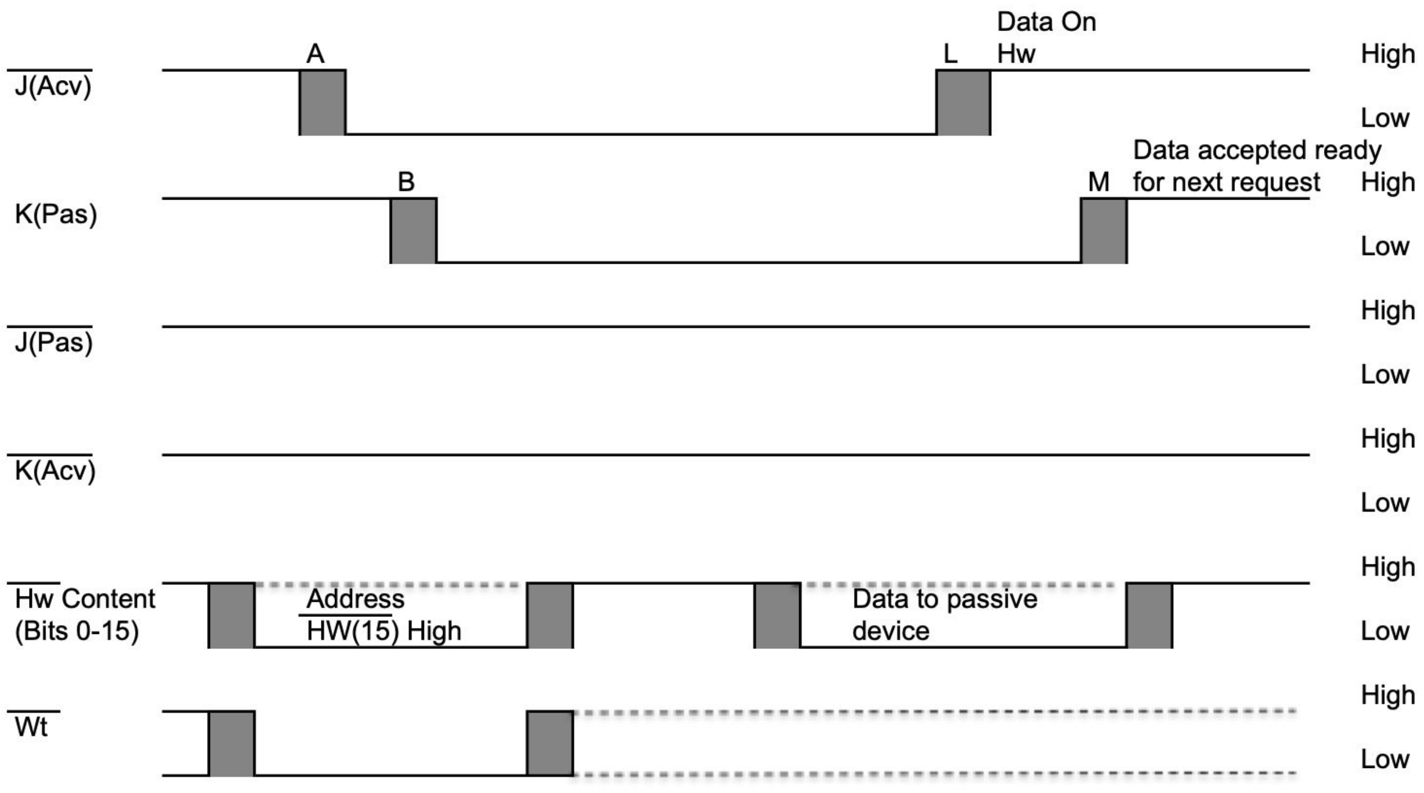
1. Shading indicates cable delay

2. No time scale is implied by the above diagram but note:

   (a) D may occur anywhere between C & F

   (b) H follows G but zero delay is allowed

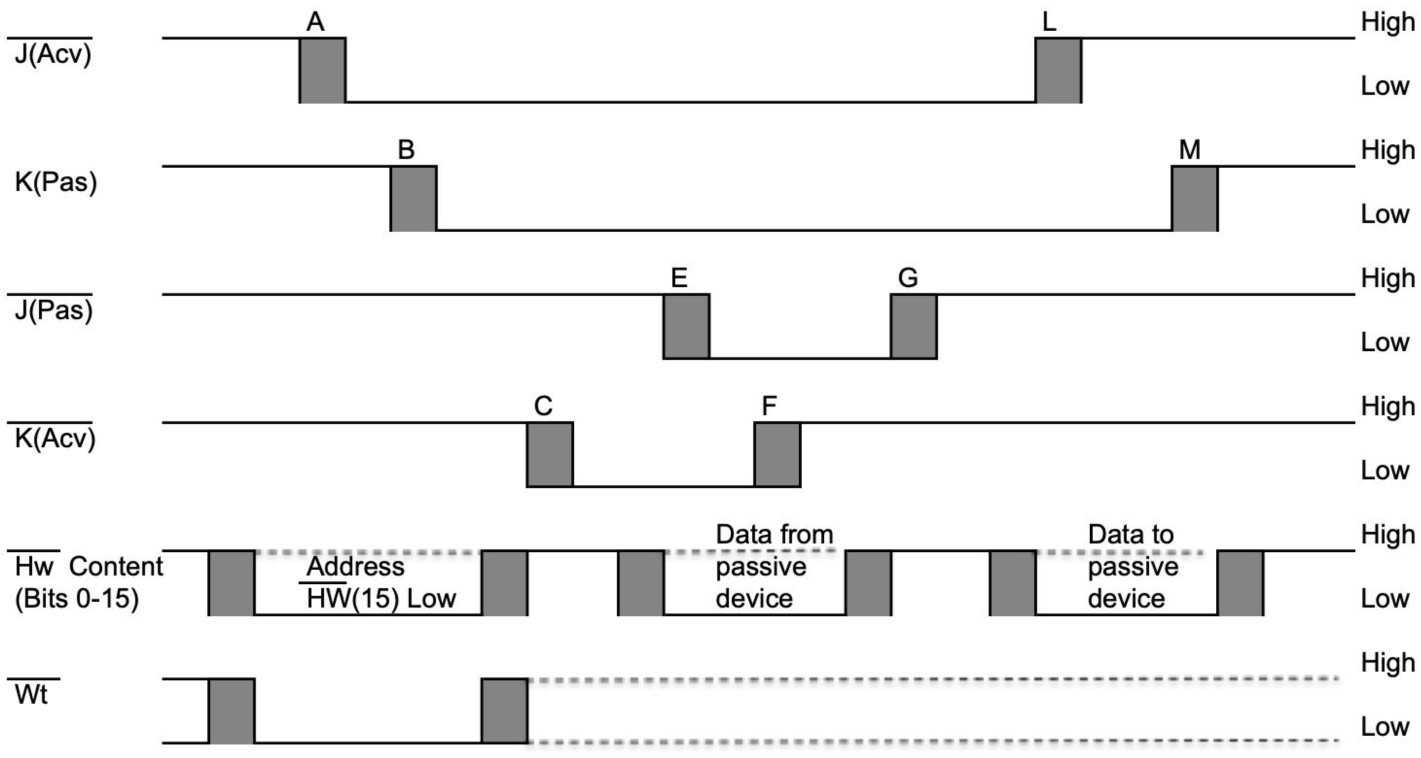
<u>READ-ONLY CYCLE</u> <u>Figure 1.3.1</u>



Notes

1. Shading indicates cable delay
2. No time scale is implied by the above diagram but note:
   (a) D may occur anywhere between C & F
   (b) H follows G but zero delay is allowed.

WRITE-ONLY CYCLE Figure 1.3.2



Notes

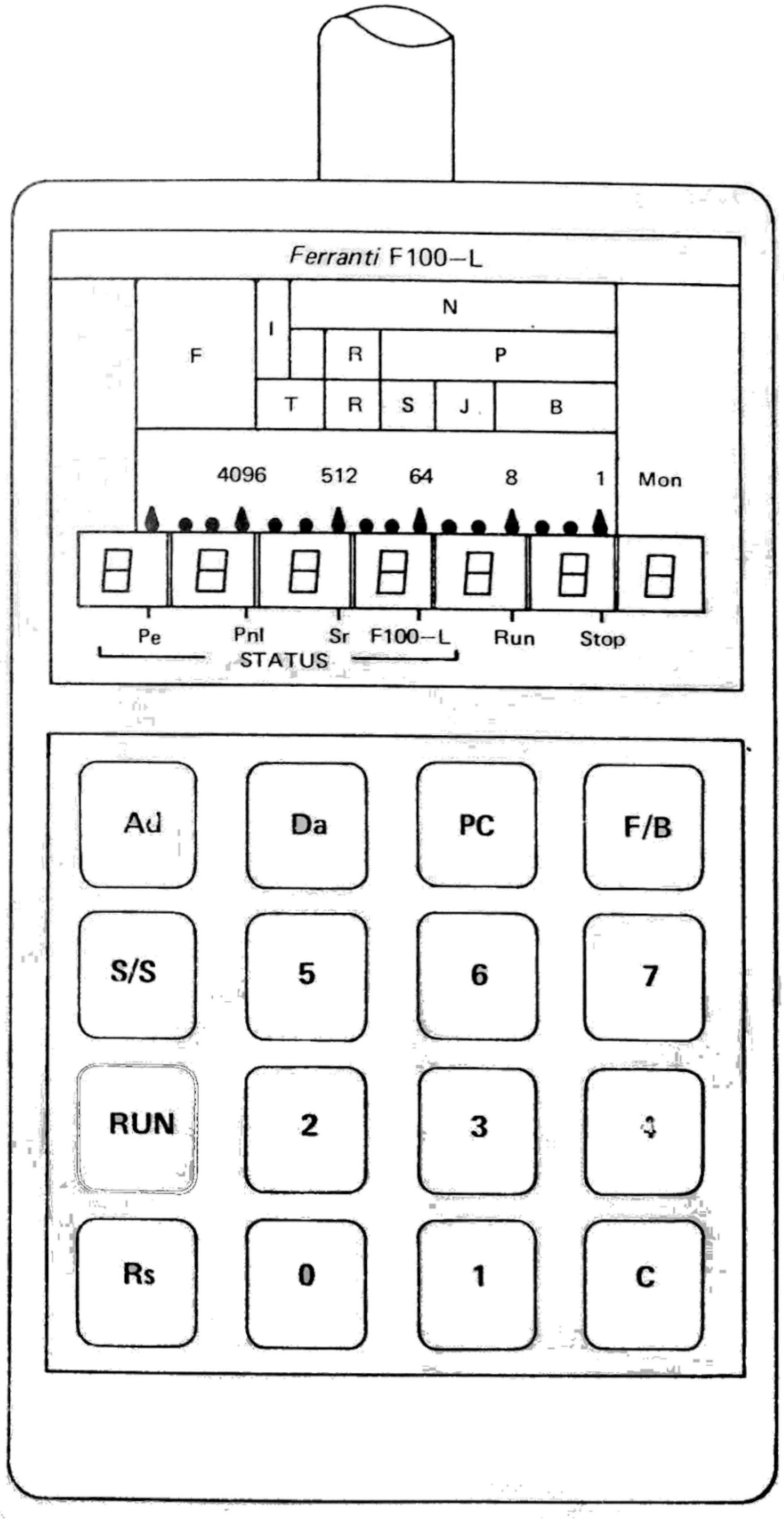
1. Shading indicates cable delay

2. No time scale is implied by the above diagram but note:

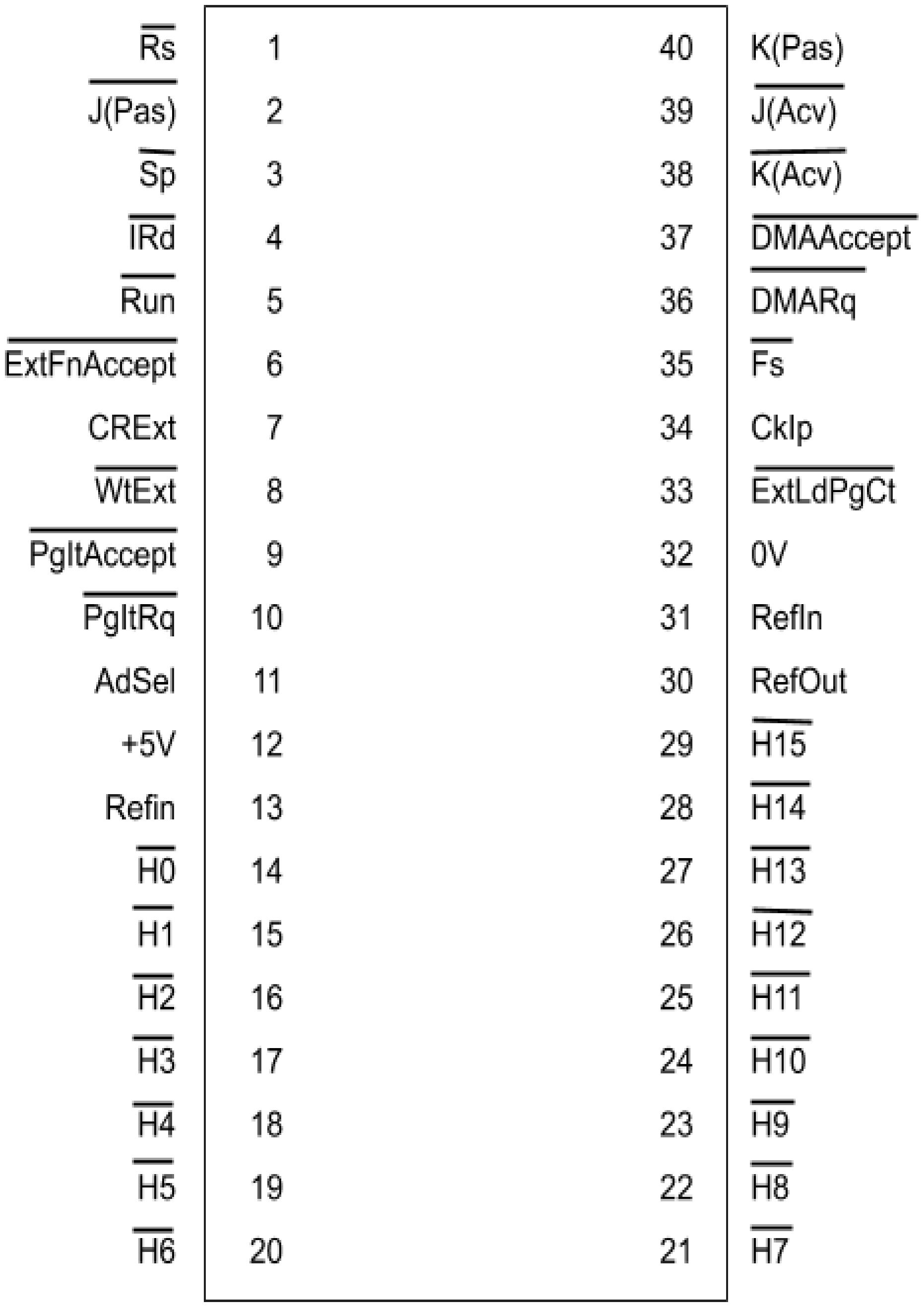
   (a) D may occur anywhere between C & F

   (b) H follows G but zero delay is allowed

READ-MODIFY-WRITE CYCLE   Figure 1.3.3



F100-L Control Handset Fig. 1.4


$\overline{Rs}$ 1 40 K(Pas)
$\overline{J(Pas)}$ 2 39 $\overline{J(Acv)}$
$\overline{Sp}$ 3 38 $\overline{K(Acv)}$
$\overline{IRd}$ 4 37 $\overline{DMAAccept}$
$\overline{Run}$ 5 36 $\overline{DMARq}$
$\overline{ExtFnAccept}$ 6 35 $\overline{Fs}$
CRExt 7 34 CkIp
$\overline{WtExt}$ 8 33 $\overline{ExtLdPgCt}$
$\overline{PgItAccept}$ 9 32 0V
$\overline{PgItRq}$ 10 31 RefIn
AdSel 11 30 RefOut
+5V 12 29 $\overline{H15}$
Refin 13 28 $\overline{H14}$
$\overline{H0}$ 14 27 $\overline{H13}$
$\overline{H1}$ 15 26 $\overline{H12}$
$\overline{H2}$ 16 25 $\overline{H11}$
$\overline{H3}$ 17 24 $\overline{H10}$
$\overline{H4}$ 18 23 $\overline{H9}$
$\overline{H5}$ 19 22 $\overline{H8}$
$\overline{H6}$ 20 21 $\overline{H7}$

F100-L Pin Allocation Fig 1.5

SECTION 2

THE INTERFACE SET

Contents

Tables

Figures

2.1 INTRODUCTION

In computer systems the Central Processing Unit (CPU) often represents only a small percentage of the total hardware cost. The application of large scale integration techniques to the CPU can show dramatic cost savings in that area, but the effect upon the overall systems cost and even physical size may not be so significant.

It is essential that, when LSI techniques are investigated a "total systems" approach is pursued, so that as much of that system hardware as possible can be constructed in LSI form to give the maximum cost and size savings.

The F100-L system architecture has been chosen so that LSI techniques can be applied to significantly more than just the central processor.

In fact the principal design aims of the F100-L system architecture were to permit the processor to interface efficiently, and hence by means of as few different 'standard' types of LSI chips as possible, to LSI or MSI based peripherals or peripheral control devices presenting widely differing data transfer, interrupt and control requirements. Additionally the architecture has been designed to permit the configuration of multi-processor systems such that requirements for special logic, which it is not yet economic to provide in LSI form, is kept to a bare minimum.

This has been made possible by the adoption of an address/data multiplexed I/O Bus by which the F100-L, its system devices and memories communicate with one another.

Consideration of the wide range, and differing characteristics, of the devices required to be handled by a micro processor in the 16-bit class, identifies two distinct classes of LSI interface/control chips:

1) A general purpose chip, or set of chips, capable of operating in a variety of modes (appropriate to the characteristics of the external device), for interfacing to the I/O Bus and to its associated interrupt and DMA lines, and for providing the most generally required facilities for peripheral control. These include interface timing, address recognition, control/data word passing and interrupt handling logic. These chips should present the simplest possible interface to the external devices.

2) An open ended range of functionally dedicated chips for carrying out the specific operations of peripheral control. These chips may take the form of line multiplexers, A/D and D/A converters, interface converters, power drivers and so forth. In practice the variety of requirements are such that this class will include fully custom designed mixed digital/linear chips, Programmable Logic Arrays, Uncommitted Logic Arrays and, for some considerable time, ordinary MSI logic. In order to simplify the design task in this area these devices should interface to

the general purpose chips of (1) above in the simplest possible way consistent with the use of a low number of pins.

The Interface Set fulfils the requirements for the general purpose set of chips described in (1) above.

2.2 THE INTERFACE SET

The Interface Set is a general purpose chipset capable of interfacing/controlling any of the classes of devices that can be supported by an F100-L system.

A single Interface Set comprises three 40-pin DIL packages (one F111-L Control Interface chip and two F112-L Data Interface chips) (See figs. 2.1 and 2.2 for pin allocation).

Each Interface Set enables a system element to be connected to an F100-L in a simple fashion, without the necessity of having a detailed knowledge of the electrical and timing rules of the I/O Bus.

A single Interface Set can replace 25 to 50 standard TTL DILs giving savings of at least 60% in printed circuit board area, with a reduction in heat dissipation, and a significant increase in reliability.

Logically an Interface Set can be regarded as a single component capable of operation in one of five modes depending upon the class of device being driven.

These five modes are:

Memory Mode

Peripheral Mode

Special Processing Unit Mode

Buffer Mode

Bus Extension Mode

These modes are explained in more detail in the following sections.

2.3 MEMORY MODE (See Fig. 2.3)

Since a microsystem may have requirements for a number of blocks of memory, and peripherals may also be required to be driven as though they are locations in memory (memory-mapped I/0), the Interface Set may be programmed (by taking the appropriate pins to 0v) to drive devices containing the following numbers of words:

32, 64, 128, 256, 512, 1K, 2K, 4K, 8K, 16K, 32K.

When a relevant address is present on the I/O Bus the Interface Set staticises it in its internal address register. The address register has 25 mA drive capability on its outputs and is thus able to drive memory directly. Thus, the only additional logic required is that necessary to time the access/write periods of the particular memory being used.

Dynamic RAM, core store, or indeed any other type of memory, may be interfaced easily to the F100-L I/O Bus by this means.

In systems where only program address peripheral transfers are necessary, all the peripheral inputs and outputs can be multiplexed onto one Interface Set and treated as a small block of memory (e.g. 32 addresses) with inputs and outputs being performed by reading to or writing from the appropriate addresses. This may take place in response to a regular timed interrupt sequence, or as otherwise required, the Interface Set operating in memory mode. The Interface Set in memory mode may handle the F100-L Interrupt line to initiate program interrupts, but, since vectoring facilities are not used in this mode, this will normally be the only program interrupt in the system.

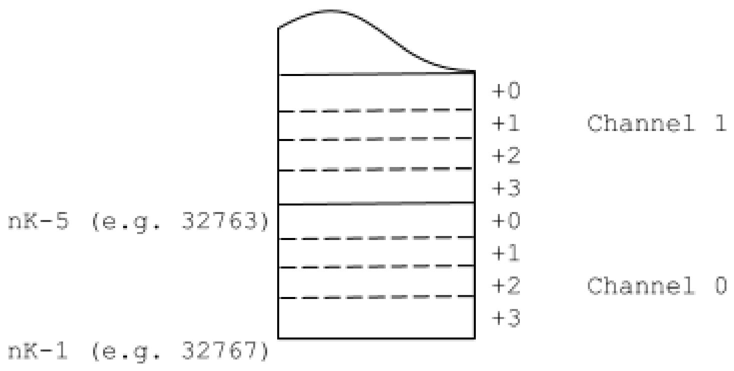
An example of a simple F100-L system with memory, and peripheral devices treated as memory, is shown in fig. 2.4.

2.4 PERIPHERAL MODE (Fig. 2.5.)

In this mode the Interface Set is primarily concerned with the handling of fast, or autonomous, real time peripherals requiring high data capture rates and is therefore centred on the control of Direct Memory Access transfers and Program Interrupts. Before either of these can be initiated, certain conditions must exist: e.g. for DMA transfers it is necessary to load the Interface Set address counter with the start address in memory to or from which data is to be transferred. Also, for a complex peripheral, it may be necessary to pre-define its mode of operation by sending it control words. The transfer of such control information can, in conventional microprocessors, give rise to requirements for a large number of additional TTL devices, but in F100-L based systems this is avoided by means of the following facilities within the Interface Set.

2.4.1 Channel Addresses

Each Interface Set operating in Peripheral Mode is allocated a fixed channel number in the range 0-63, defined by taking the appropriate pins to 0v. Associated with each channel number is a set of four addresses referred to for each Interface Set as +0, +1, +2, +3. These are generally located at the top of the F100-L address range, but can be relocated in units of 1K if required, i.e.



The top address, +3, enables the program to load the start address of a DMA block into the Interface Set address counter, simply by writing to that address.

The remaining three addresses are not associated with hardware internal to the Interface Set, but can be used to access any registers (of up to 16-bits) within the external logic as required.

By convention these addresses are used as below:

	Reading from	Writing to
+0	Read data (Programmed Peripheral Transfers)	Write data
+1	Read status	Write additional control bits
+2	Not allocated	Write Master Control Bits
+3	Read Address Counter L	oad Address Counter

2.4.2 Direct Memory Access

DMA enables peripheral devices which have either a high data transfer rate or a large number of words to be transferred, to carry out transfers directly to/from memory via the I/O Bus.

A DMA sequence will normally be initiated by F100-L loading the Address Counter of the Interface Set with the starting address of an area in memory.

The Interface Set requests F100-L for use of the bus each time the external logic is ready to transmit or receive a data word. Upon access to the bus being granted, the data word is transferred by the Interface Set to the address specified by the Interface Set address counter. The address counter is incremented before each transfer so that successive autonomous transfers may take place.

The sequence will normally be terminated by a peripheral generated Program Interrupt, the necessary count of the number of words transferred being carried out in the logic external to the Interface Set.

DMA transfers are effectively interleaved with the F100-L's use of the bus since a DMA access to the I/O Bus is permitted at least once in each instruction. This enables high data transfer rates to be achieved without significantly slowing the F100-L. Should it be necessary to provide exceptionally high rates, F100-L may be held by the peripheral concerned. for the duration of a sequence of transfers, providing that a Buffer Interface Set, which provides the Hold facility, is included in the system (See section 2.6). Up to two peripherals may be provided with this facility. Alternatively, in the interests of reducing processor load and/or not holding up transfers associated with other devices, a critical peripheral may be allocated an extended area of bus, thus allowing simultaneous operation. This is described under "Bus Extension Mode", Sect. 2.7.

When a system includes more than one peripheral requiring DMA facilities the F100-L DMA line is connected in series via successive Interface Sets. This establishes a natural priority system in which the peripheral whose Interface Set is electrically nearest F100-L has the highest priority, and so forth in decreasing priority order. DMA requests from lower priority Interface Sets then pass through the higher priority Interface Sets, with a delay of only 75 nsecs at each stage unless inhibition of the signal takes place due to use of the DMA facilities by an interposed Interface Set. This method avoids the necessity of the system designer adding yet more TTL logic to his system to perform priority multiplexing as is commonly required by other microprocessors, and is doubly valuable because it removes the associated design development load of a difficult area in logic design. (see Fig. 2.6.)

2.4.3 Program Interrupts

This facility is required by real time peripherals to inform the processor that a particular event has occurred.

physically, the Interface Sets are connected to the processor in the same way as for DMA, the F100-L program interrupt line being daisy chained via successive Interface Sets as illustrated in Fig. 2.6.

A Interface Set having requested a Program Interrupt receives a signal back down the daisy chain asking it to send a channel identifier to F100-L, over the I/O Bus, thus enabling F100-L to branch to the specific interrupt program associated with the peripheral concerned. This fully 'vectored' interrupt system saves the overhead of a polling program that would otherwise be required to check each peripheral every time an interrupt is received to ascertain the identity of the calling peripheral.

When an Interface Set generates an interrupt, the current program being actioned by F100-L is suspended, the values of the program counter and the condition register are put on the Link Stack and the required interrupt program is entered. The response time, assuming no other interrupt is being actioned, is typically about 9 usecs. The suspended program can be re-started at the end of the interrupt program by a single Return instruction. This will restore the program counter and the condition register, the total time taken, again, being only 9 usec. This is exceptionally low in the micro processor context.

2.5 SPU MODE

This mode is required to enable Interface Sets to handle Special Processing Units. The ability to incorporate such Units in F100-L systems is described in Section 1. By means of the External Function instruction, an 11-bit field allows the definition of any function or functions which may then be actioned by external MSI or LSI hardware, as appropriate. An SPU, handled by an Interface Set operating in SPU mode, is included in the system illustrated in fig. 2.6.

Since an SPU may be capable of performing more than one External Function, the Interface Set is, in SPU Mode, able to monitor each instruction on the I/0 Bus and to compare it with the instruction, or range of instructions, that its SPU can perform. This range is defined by taking the appropriate select inputs to 0v. Having recognised one of its own External Functions, the Interface Set places the instruction in its internal register. The bit pattern necessary to define which of its range of External Functions is required is then output to the SPU control logic to initiate the required sequence.

All data for use with External Functions is obtained using the DMA facility, the address of any data to be obtained being loaded into the Address Counter within the Interface Set by the SPU control logic.

Normally an Interface Set controlling an SPU will be positioned electrically closest to F100-L in the priority 'daisy chain', in the interests of processing performance.

2.6 BUFFER MODE

Unlike MOS based microprocessors F100-L outputs are at TTL levels and have a current sink capability of as high as 6 mA, which is sufficient to drive a large number of Interface Sets. In practice, however, when the I/O Bus exceeds a length of approximately 15 cm, or has multiple spurs, it will behave as a transmission line and must, therefore, be terminated by its characteristic impedance. The F112-L Data Interface Chip has high current sink driver circuits to enable a terminated I/O Bus to be driven. Thus, just two F112-L chips can be used as a current buffer between the F100-L itself and the terminated bus. The I/O Bus can then be extended to a distance of a few metres. Typically, the use of a Buffer Interface Set will only add 100 nsec to the instruction time. A Buffer Interface Set is included in the system illustrated in Fig. 2.6.

2.7 BUS EXTENSION MODE

This mode of operation enables the simple and economical realisation of multi-bus and multi-processor systems. Such configurations often occur when there is a need :

a) to have a very long I/O Bus which exceeds the permitted length of a single Bus (see Fig. 2.7);

b) to off-load the data transfers associated with a specific peripheral or SPU/peripheral combination (see Fig. 2.8);

c) to provide memory and peripheral devices which may be accessed by more than one processor (see Fig. 2.9).

These system structures use a Bus Extension Unit, which comprises two Interface Sets, connected back-to-back without the need of any additional circuitry. One of the two Interface Sets functions in Bus Extension Mode and the other in Memory Mode. Further details are given in Sect. 2.7.1 below.

2.7.1 Bus Extension Units (see Fig. 2.10)

Bus Extension Units enable the generation of an additional I/O Bus. The original Bus is known as the Primary Bus and the additional one as the Secondary Bus.

The Interface Set on the Primary Bus is in Memory Mode. The address range recognised by the Memory Mode I-F Set is that of the addresses on the Secondary Bus that it is necessary to access from the Primary Bus.

When the Memory Mode I-F Set accepts such an address, it passes it to the I-F Set on the Secondary Bus along with a request for the appropriate cycle type (i.e. Read, Write, Read-Modify-Write). The I-F Set on the Secondary Bus is in Bus Extension Mode which enables direct connection to be made between the two I-F Sets.

The Bus Extension Mode I-F Set now has to generate the appropriate Bus cycle on the Secondary Bus. Since there may be DMA devices on the Secondary Bus, this cycle is performed as a DMA by the Bus Extension Mode I-F Set. A DMA daisy chain particular to the Secondary Bus is set up (see fig. 2.8), the allocation of priorities being at the user’s discretion. Any devices on the Secondary Bus using the Program Interrupt facility are connected directly into the Program Interrupt daisy chain on the Primary Bus, as shown in fig. 2.8.

The use of further Bus Extension Units enables additional Secondary Buses to be added to the Primary Bus.

2.7.2 Use of Bus Extension Units to Increase Bus Length (fig.2.7).

Since the I/O Bus obeys the laws of physics and acts as a transmission line it is sometimes necessary to partition it to maintain clean signal edges. This is often the case when most of the system is local to the F100-L and in one shelf unit with an additional device a few metres away. It is then convenient to have all the local devices on a single Primary Bus and to generate a Secondary Bus to go to the remote device. Further Secondary Buses can be added if necessary.

2.7.3 Use of Bus Extension Unit for Functionally Divided Systems (fig. 2.8)

In some systems a device may impede the use of the I/O Bus by requesting DMA transfers at a very high rate. (e.g. a Fast Fourier Transform Processor SPU). In these situations it is convenient to put the high data rate device and its associated memory (and possibly I/0 interfaces) onto a Secondary Bus, where any high speed transfers can occur in parallel with the normal operation of the Primary Bus

In the situation shown in fig. 2.8 the F100-L is on the Primary Bus with its ROM, RAM and other peripherals. The F100-L can control the devices on the Secondary Bus by writing control information to them through the Bus Extension Unit. The I/O devices can then transfer data, using DMA from the local RAM store and the SPU can process the data all totally independent of processing and peripheral control transfers taking place on the Primary Bus.

More than one such Secondary Bus can be attached to the Primary Bus.

2.7.4 Multi-Processor Configurations by Bus Extension Units (fig. 2.9)

The Bus Extension Unit is a very powerful facility for generating functionally partitioned multi-processor systems. A Secondary Bus can be accessed from two or more Primary Busses, each with its own F100-L. In this way 'common' peripheral, memory and SPU resources may be accessed by each F100-L processor. The priority of access of each Primary Bus to the Secondary Bus may be defined by implementing the DMA daisy chain via the Interface Sets operating in Bus Extension Mode.

The Bus Extension Unit is not, by itself, of course, the complete answer to multi-processor configurations, and is therefore backed up by certain facilities inherent in F100-L. For example, in some applications inter-processor communication of status information may be necessary, which can be implemented conveniently by means of message areas in the shared memory. For integrity it is necessary, however, to prevent the originating processor changing a message area while that area is being read by another processor. This requirement can be met by defining a single bit in memory to be associated with each message area. The appropriate bit can be tested by a processor when it requires access to a particular area. If the bit is clear, then access is granted: but the bit must then be set immediately to ensure lock-out of any other processor while the area is read. To permit this the 'bit test and invert' instruction must be implemented in such a way that no other processor can test the bit until the 'invert' action has been completed, i.e. the test and invert procedure must be treated as an uninterruptable cycle. F100-L provides just this facility by means of the 'jump if bit clear and then set' instructions. The reading processor will, of course, clear the bit when it has read the desired data.

2.8 How Many Interface Sets In A System ?

In figs. 2.4 and 2.6 each system device is shown interfaced to the I/O Bus by an Interface Set. However, when it comes to designing a specific system the number of Interface Sets required will depend to a large extent on how the system partitions onto the individual printed circuit cards.

If in fig 2.4 the memory size and amount of I/O were small, and the printed circuit card was large enough to hold the entire system, then only one Interface Set would be required. The memory and entire I/O could be multiplexed onto the Interface Set.

Should the printed circuit card be very small, as perhaps in an avionics application, then the partitioning would dictate the use of an Interface Set per function per card as shown in fig 2.4. The I/O Bus would be run on an interconnecting backplane.

The use of one Interface Set per function per card is used in the F100-L Microcomputer System (Section 3), giving the advantage of one standard module with the I/O Bus on a backplane; each individual system being simply configured by plugging-in the appropriate standard cards in the necessary slot positions.

For any given system application it will first be necessary to partition the system onto the number of cards required. At this stage it will be possible to identify any cards that can be regarded as a possible “standard”. On any one card it is convenient to group together functions requiring the same Mode Interface Set. Having multiplexed these functions onto the Interface Set, the card now presents a standard I/O Bus interface, and can be connected directly onto the I/O Bus.

Thus the use of the Interface Set to drive the standard I/O Bus enables F100-L systems to be constructed in a simple modular fashion, possibly making use of any other cards previously developed for other applications, thereby reducing development costs.

2.9 Summary

The interface Set is a key development in constructing micro-processor systems quickly, efficiently and with large hardware savings. Not only does the Interface Set interface every class of device onto the I/0 Bus, it also provides the control structure for the powerful F100-L facilities.

The user side of the Interface Set has been designed to be as simple to use as possible. The user requests the particular function and receives a signal back when that function has been completed.

When used in a Bus Extension Unit, the full flexibility of F100-L systems is realised, with the ability to configure functionally partitioned systems.

Recent system designs have shown that the use of the Interface Set dramatically reduces the card-count with the attendant savings in size, weight, cost, and spares holdings and, of course, with an increase in reliability.

The Interface Set ensures that not only the small systems are simple to design and construct but that the larger, more complex (and traditionally more complicated) systems are equally simple to configure.

Parameter		Min.	Typ.	Max.	Units
Supply Voltage (Pin 12)		4.75	5	5.25	V
Low Level Output Current	Normal			6	mA
	Driver			25	mA
	Line Driver			33	mA
Operating free-air temperature	Military	-55		+125	°C
	Industrial	-25		+85	°C
	Commercial	0		+70	°C

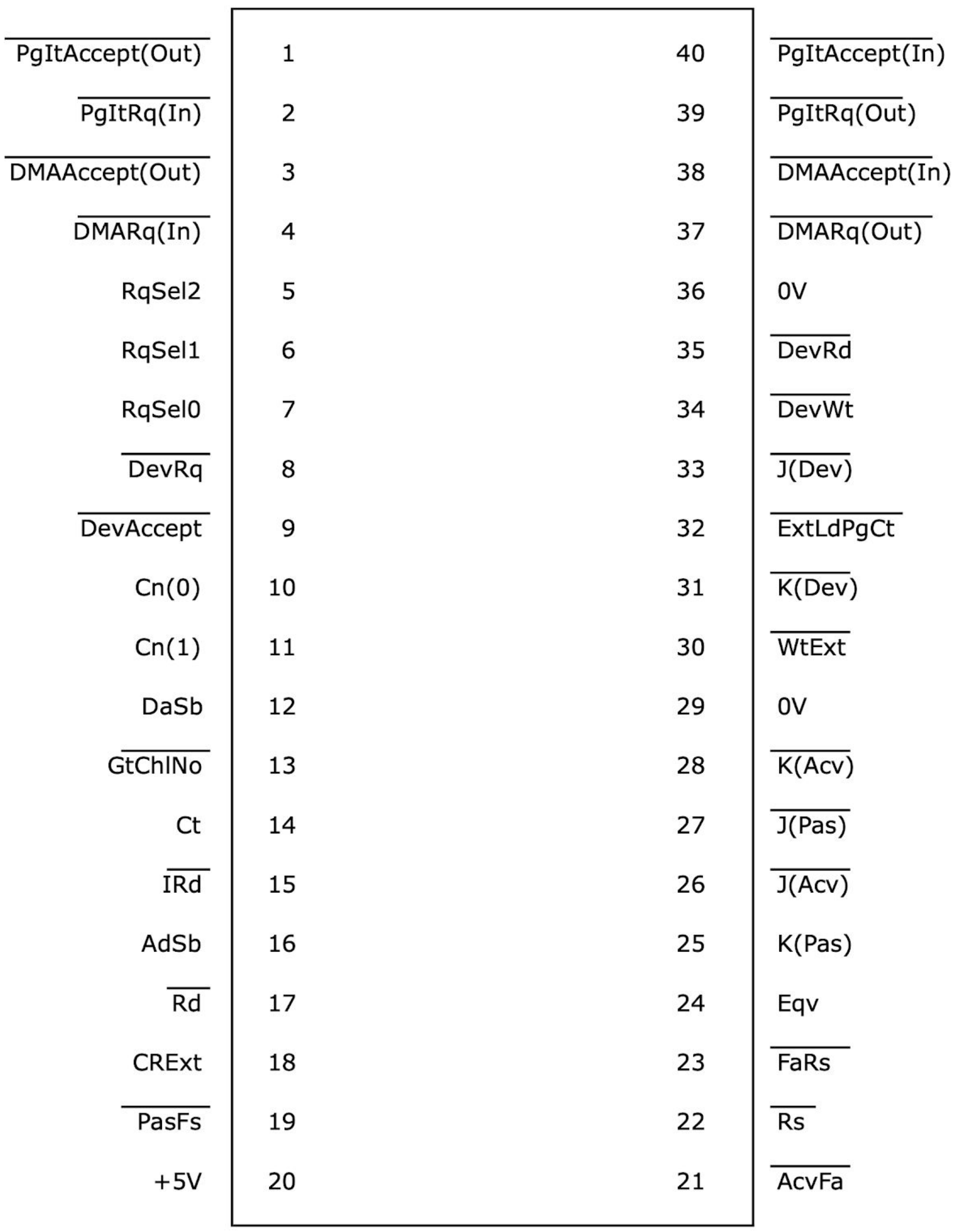
Table 2.1

Recommended Operating Conditions

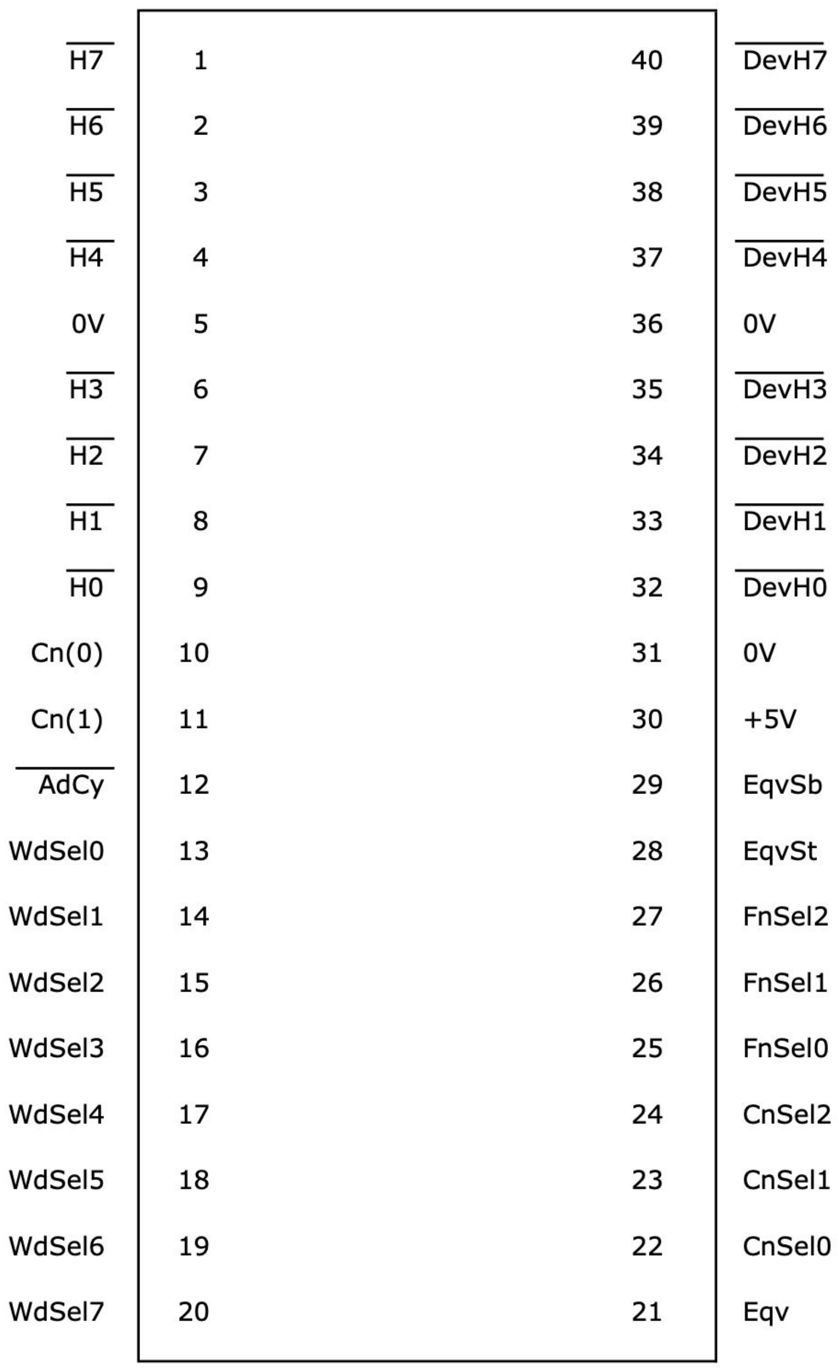
Parameter		Min.	Typ.	Max.	Units
High Level input voltage		1.6			V
Low Level input voltage				1.0	V
Low Level output voltage					V
High Level input current (Except F112-L pins 6-9 & 23)				(Vin-1.5)/5	mA
High Level input current (F112-L pins 6-9 & 23 only)				(Vin-1.5)/2.5	mA
Low Level input current (Vin=0.4V, Vcc=5.0V)	No pull-up				uA
	2K pull-up		2.3		mA
	5K pull-up		0.92		mA
	15K pull-up		0.46		mA
Short circuit current			1		mA
Supply Current	F111-L		85		mA
	F112-L		120		mA

Table 2.2

Electrical Characteristics

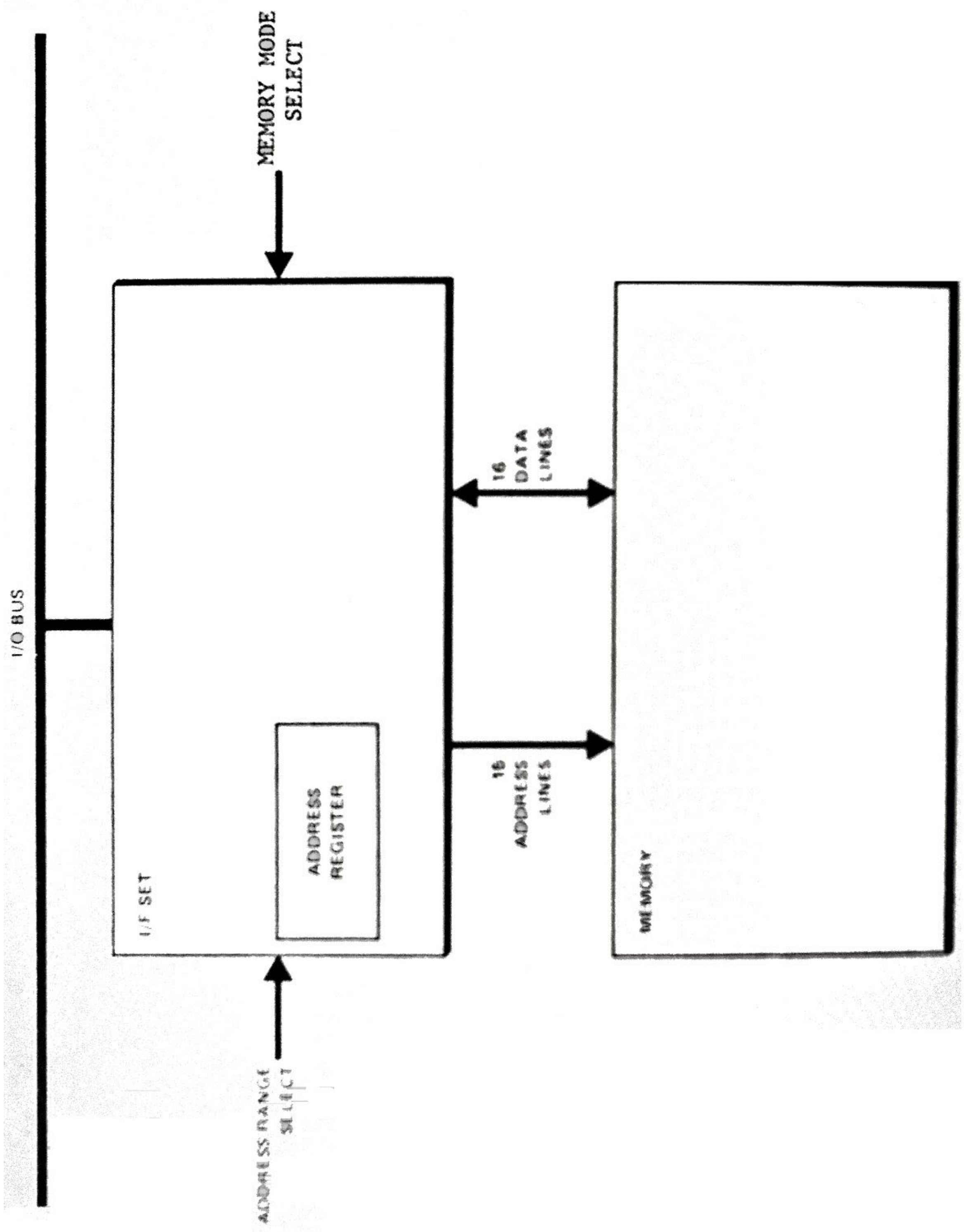

$\overline{\text{PgItAccept(Out)}}$
1
$\overline{\text{PgItRq(In)}}$
2
$\overline{\text{DMAAccept(Out)}}$
3
$\overline{\text{DMARq(In)}}$
4
RqSel2
5
RqSel1
6
RqSel0
7
$\overline{\text{DevRq}}$
8
$\overline{\text{DevAccept}}$
9
Cn(0)
10
Cn(1)
11
DaSb
12
$\overline{\text{GtChlNo}}$
13
Ct
14
$\overline{\text{IRd}}$
15
AdSb
16
$\overline{\text{Rd}}$
17
CRExt
18
$\overline{\text{PasFs}}$
19
+5V
20
40
$\overline{\text{PgItAccept(In)}}$
39
$\overline{\text{PgItRq(Out)}}$
38
$\overline{\text{DMAAccept(In)}}$
37
$\overline{\text{DMARq(Out)}}$
36
0V
35
$\overline{\text{DevRd}}$
34
$\overline{\text{DevWt}}$
33
$\overline{\text{J(Dev)}}$
32
$\overline{\text{ExtLdPgCt}}$
31
$\overline{\text{K(Dev)}}$
30
$\overline{\text{WtExt}}$
29
0V
28
$\overline{\text{K(Acv)}}$
27
$\overline{\text{J(Pas)}}$
26
$\overline{\text{J(Acv)}}$
25
K(Pas)
24
Eqv
23
$\overline{\text{FaRs}}$
22
$\overline{\text{Rs}}$
21
$\overline{\text{AcvFa}}$

F111-L Pin Allocation Fig. 2.1

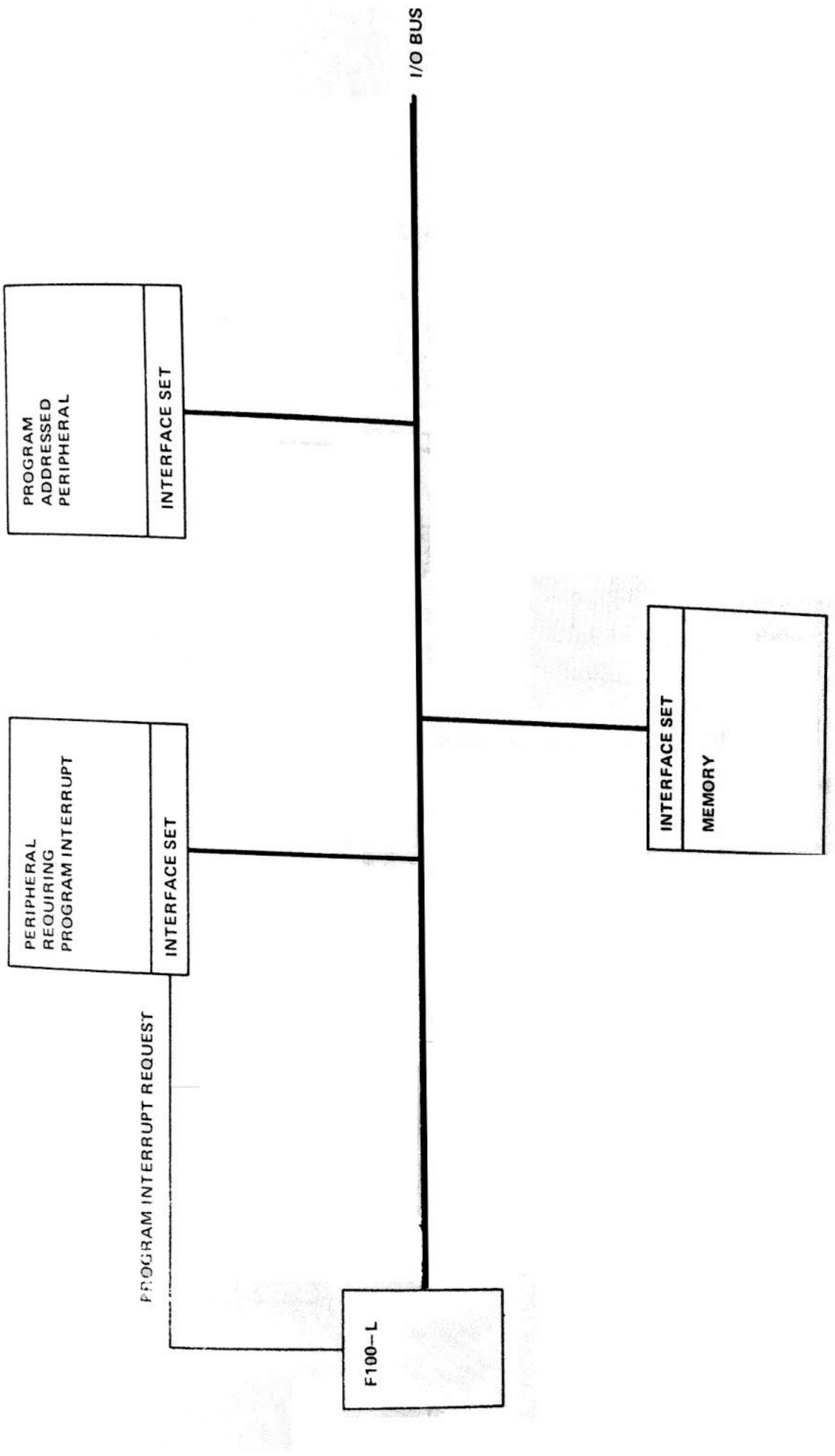


$\overline{H7}$ 1 40 $\overline{DevH7}$
$\overline{H6}$ 2 39 $\overline{DevH6}$
$\overline{H5}$ 3 38 $\overline{DevH5}$
$\overline{H4}$ 4 37 $\overline{DevH4}$
0V 5 36 0V
$\overline{H3}$ 6 35 $\overline{DevH3}$
$\overline{H2}$ 7 34 $\overline{DevH2}$
$\overline{H1}$ 8 33 $\overline{DevH1}$
$\overline{H0}$ 9 32 $\overline{DevH0}$
Cn(0) 10 31 0V
Cn(1) 11 30 +5V
$\overline{AdCy}$ 12 29 EqvSb
WdSel0 13 28 EqvSt
WdSel1 14 27 FnSel2
WdSel2 15 26 FnSel1
WdSel3 16 25 FnSel0
WdSel4 17 24 CnSel2
WdSel5 18 23 CnSel1
WdSel6 19 22 CnSel0
WdSel7 20 21 Eqv

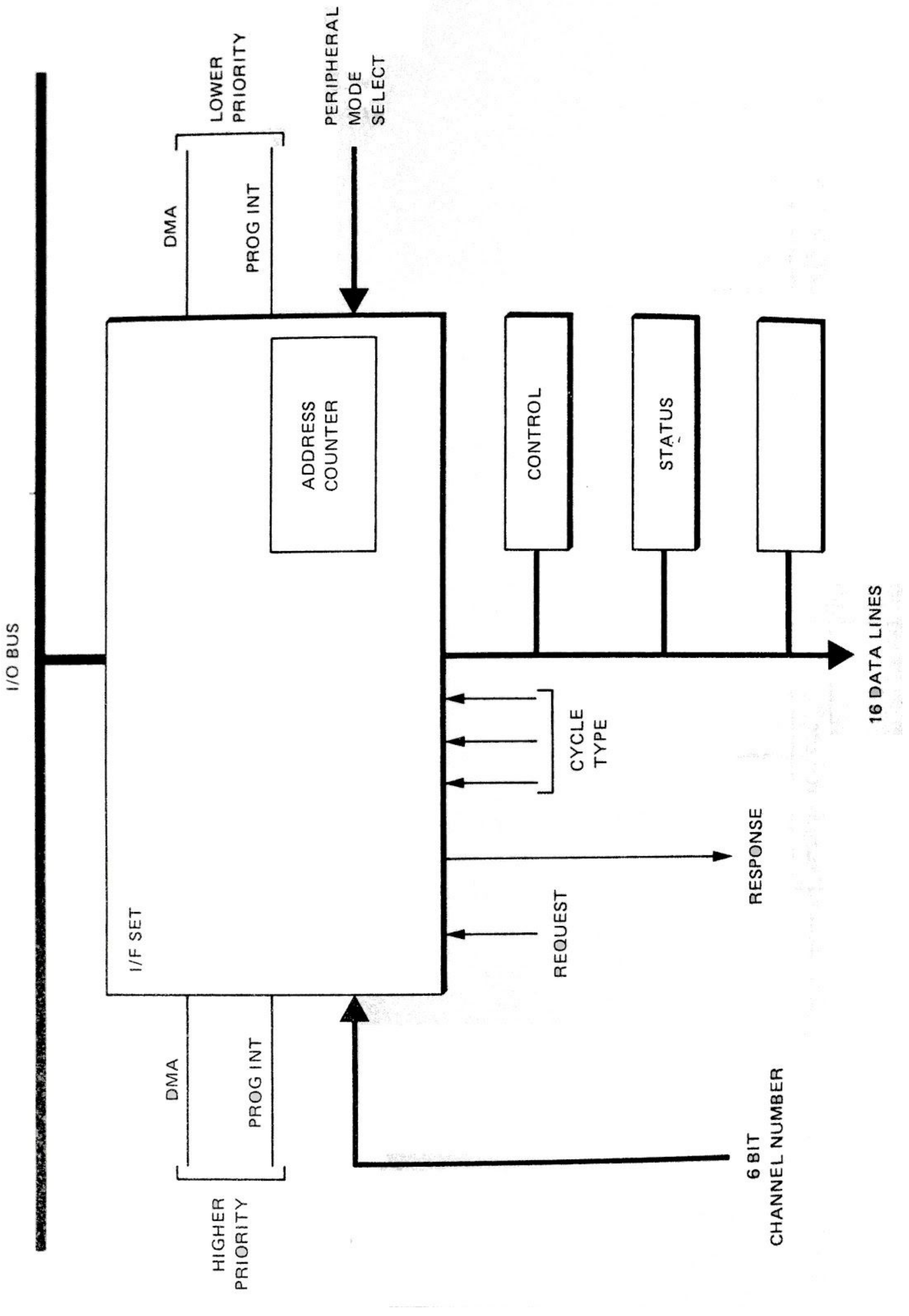
F112-L Pin Allocation Fig. 2.2



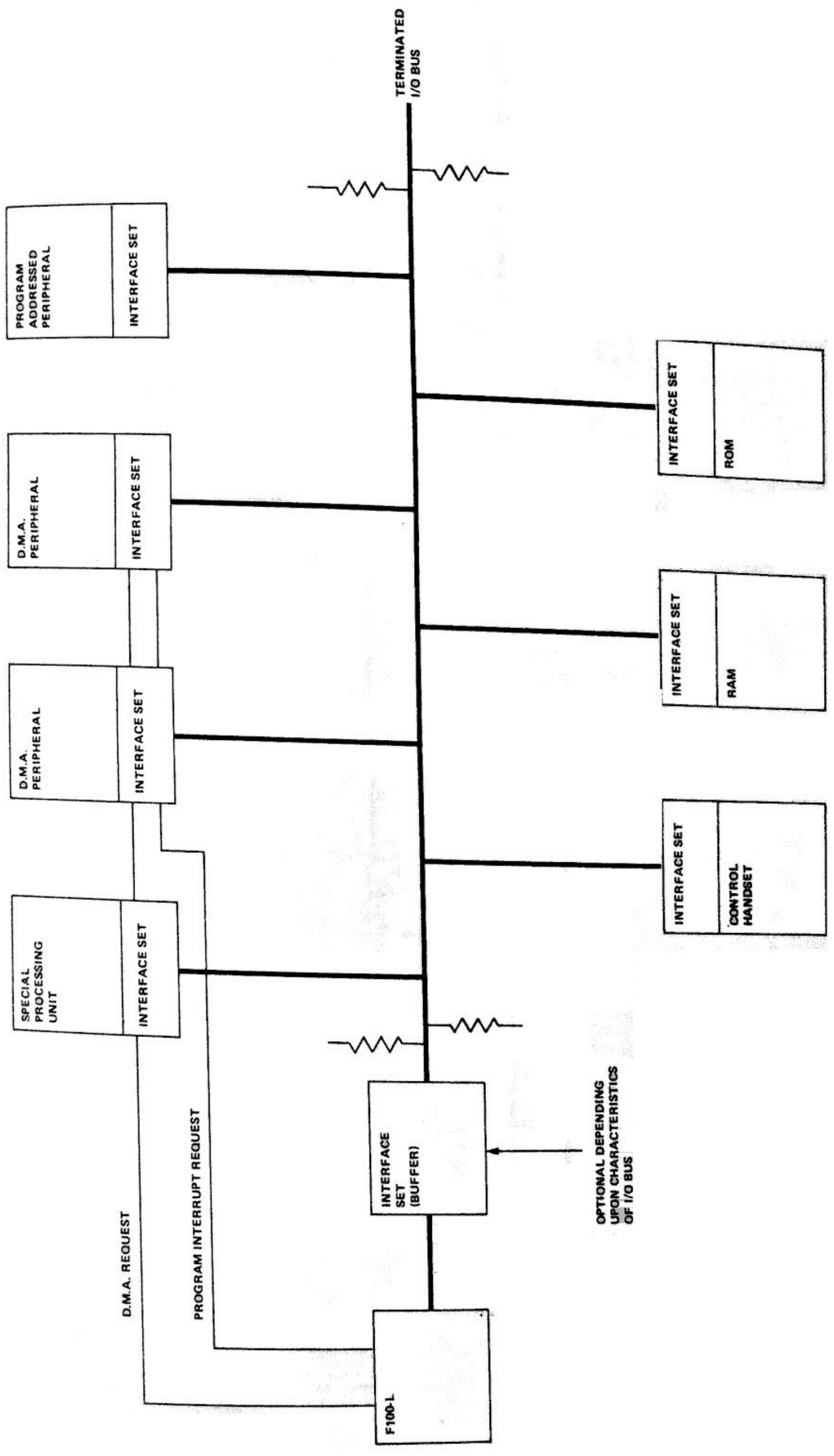
<u>INTERFACE SET - MEMORY MODE</u>   <u>Fig. 2.3</u>


I/O BUS
PROGRAM ADDRESSED PERIPHERAL
INTERFACE SET
PERIPHERAL REQUIRING PROGRAM INTERRUPT
INTERFACE SET
INTERFACE SET
MEMORY
PROGRAM INTERRUPT REQUEST
F100–L

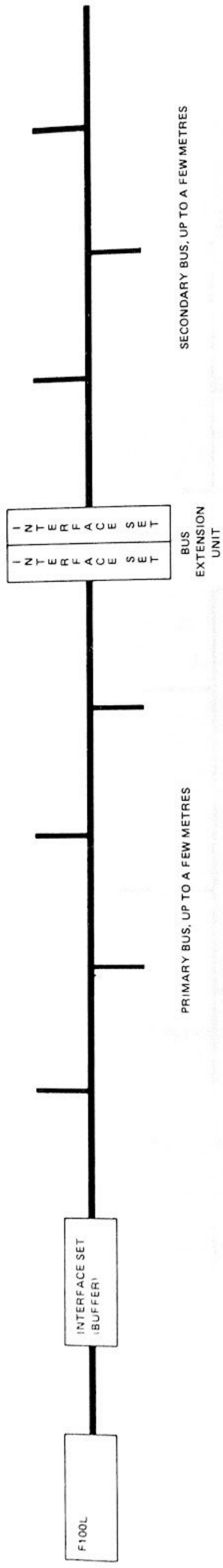
SIMPLE F100-L SYSTEM   Fig. 2.4



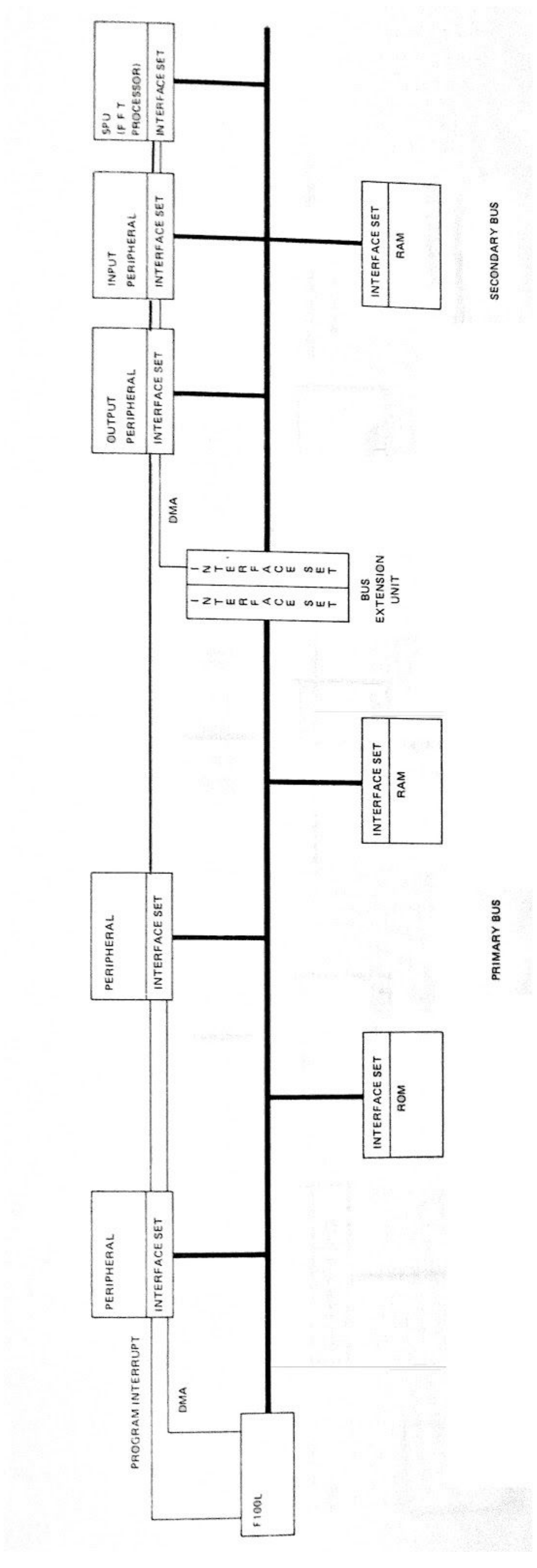
I/F SET PERIPHERAL MODE Fig. 2.5



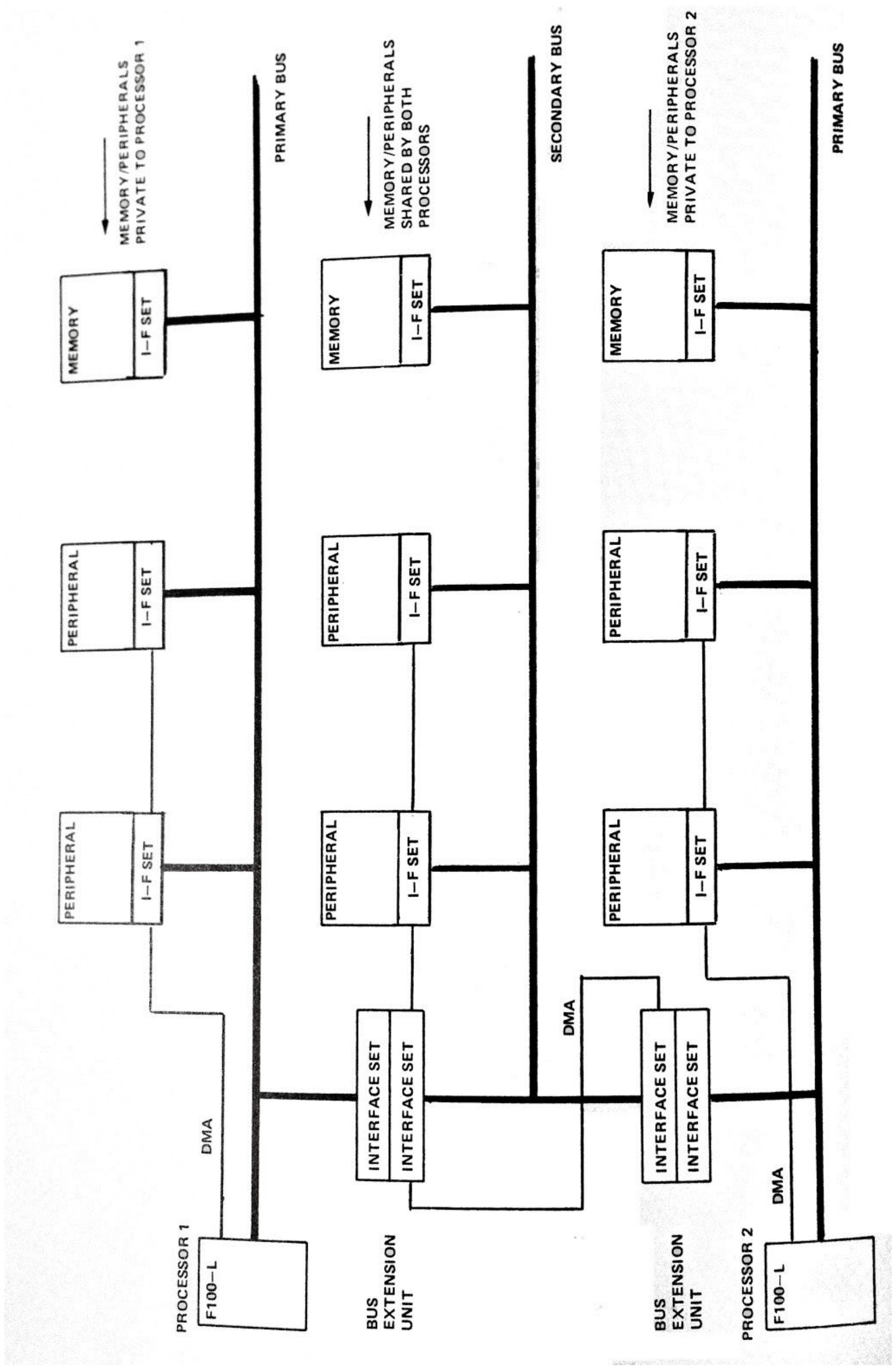
<u>More Powerful F100-L Real Time System With Special Processing Unit</u> <u>Fig.2.6</u>



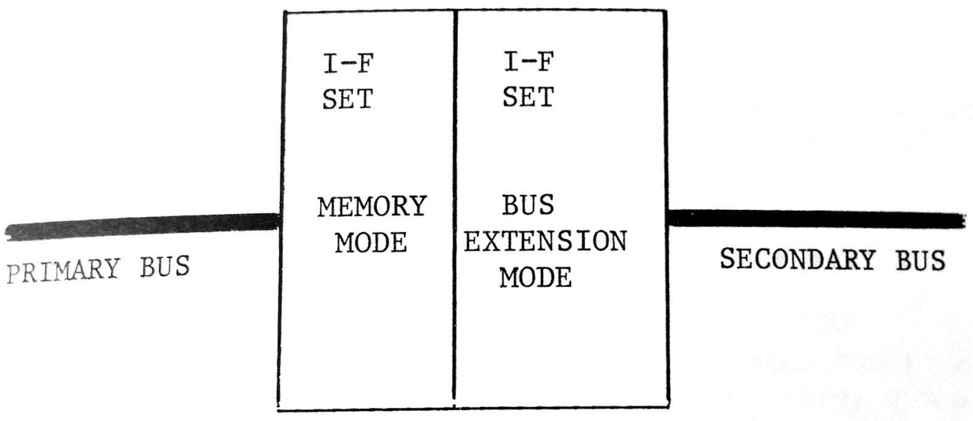
USE OF BUS EXTENSION UNIT TO INCREASE BUS LENGTH Fig. 2.7



<u>USE OF BUS EXTENSION UNIT FOR FUNCTIONALLY DIVIDED SYSTEM</u> <u>Fig. 2.8</u>



MULTI F100-L CONFIGURATION WITH SHARED SYSTEM RESOURCES Fig. 2.9



BUS EXTENSION UNIT Fig. 2 10

SECTION 3

THE F100-L MICROCOMPUTER

Contents

Table

Figures

3.1 INTRODUCTION

In addition to supplying the F100-L microprocessor and its circuits as discrete LSI components, Ferranti also provides fully engineered F100-L Microcomputer Systems.

Standard printed circuit cards containing the F100-L micro-processor, memory and peripheral interfaces slot directly into standard shelf units to form an F100-L Microcomputer System.

The inherent modularity of an F100-L system (see fig. 1.1) is used to advantage in the Microcomputer, with the backplane of the shelf unit carrying only the I/O Bus, the daisy-chained DMA and Program Interrupt signals, and a small number of other control lines. Each system function (e.g. memory, peripheral interface) is packaged onto a single printed circuit card with its own Interface Set and can, therefore, be plugged directly into the module’s backplane. The position of a card in the shelf unit is determined only by its priority requirements if it uses the DMA or Program Interrupt facilities.

The F100-L, itself, has been put onto a RAM card along with a Buffer Mode Interface Set to drive the I/O Bus as a terminated line.

Two standard modules are available for operation in a commercial/laboratory environment, one containing a single 13-slot backplane, the other having two 13-slot backplanes for multibus/multiprocessor applications.

Ruggedised modules, built using the Ferranti STEP Equipment Practice containing 9-slot or 13-slot backplanes are also available.

The availability of the F100-L Microcomputer as an OEM product removes the need for the user to commit development time, effort and money to producing the basic processor/memory/peripheral interface cards and module. Only that development specific to the users application (e.g. custom I/O interface, software) need be necessary.

The F100-L Microcomputer also enables resident versions of the F100 Support Software to be made available, essential for the user that does not have access to other computing facilities. Standard Microcomputer configurations (known as F100 Development Systems) are available to support the different levels of resident software (see Sect. 3.4).

3.2 MICROCOMPUTER MODULES

Two different mechanical constructions are available depending upon whether the application requires to be ruggedised or is intended for a commercial/laboratory environment.

3.2.1 Commercial/Laboratory Environment

The commercial microcomputer module is a 482mm (19") rack-mounting unit 222mm (8.75") in height, and is available in two forms:

(1) 01000/A, which can hold up to 13 standard cards

(2) D1000/B, which can hold up to 26 standard cards

In both versions the backplane is vertically mounted across the rear of the module, with the printed circuit cards plugging-in vertically from the front. The backplane is only half the height of the module, and connects with the lower of the two edge-connectors of the printed circuit card (see Sect. 3.3). This connector carries the I/O Bus and all other necessary connections to the F100-L microprocessor.

The space above the backplane enables connections to peripheral devices to be made directly to the upper edge connector of the standard cards.

3.2.1.1 13-slot Module (D1000/A) (fig. 3.1)

This is a self-powered (mains input) microcomputer module fitted with a 13-slot backplane. It can be mounted directly in a 19" racking system provided a vertical column of forced air cooling is supplied.

For stand-alone operation a case can be provided containing a tangential fan-unit to cool the system. The approximate overall size is 270mm (10.6 in.), 500mm (19.7 in.), 450mm (17.7 in.) D.

3.2.1.2 26-slot Module (D1000/B)

In this version two 13-slot backplanes are fitted which occupy the full width of the module, and hence no power supplies can be incorporated within the unit. A separate 19" rack-mounting power supply system is available.

The provision of two separate 13-slot backplanes enabled multibus and multiprocessor systems to be assembled, using the Bus Extension Card described in Sect. 3.3.9.

3.2.2 Ruggedised Equipment

Where operation in a more severe environment is required ruggedised modules are available. These are standard modules from the Ferranti STEP range of equipment designed to meet Def 133, Class N1 and L2. Two modules are produced containing 9-slot and 13-slot backplanes respectively.

The backplanes used in these ruggedised units are, again, mounted vertically across the rear of the module. A hinge-down front plate enables the cards to be plugged-in vertically from the front (see fig. 3.2). Full height backplanes are used making connection with both of the edge-connectors of standard cards. This enables a more rigid location of the cards. Input/Output connections are wired from the upper edge-connector (now firmly supported on the full height backplane) to multi-way connectors on the module backplate from which cables can be taken to the peripheral devices concerned. Ruggedised power supply units are also available from the STEP range of equipment to drive F100-L Microcomputers.

3.3 STANDARD CARDS

A range of standard cards are available that plug directly into any of the commercial or ruggedised modules.

All of the cards used the same printed circuit construction. (see fig. 3.3). The cards are 167.4mm X 231mm (6.6 in. x 9.1. in.) in size and are multilayer with buried ground and voltage planes. Two edge-connectors are provided at one end of the card. The lower edge-connector (2.54mm, 0.1 in. pitch) mates with the I/O Bus and other control signals that are provided by the module backplane. The upper edge connector consists of 4 groups of 12 connectors (2.54mm, 0.1 in. pitch) and is used for input-output connections. An edge-connector for monitoring/test purposes is provided at the other end of the card.

The specification of the components fitted to the standard cards differs depending upon the users requirement. Commercial/Laboratory standard equipment will be fitted with 0°C to 70°C commercial specification devices.

3.3.1 F100-L 4K/16K RAM and Timer Interrupt

This card contains:

(a) F100-L microprocessor and clock oscillator

(b) A Buffer Mode Interface Set, to drive the terminated I/O Bus supplied in the microcomputer modules together with a set of line terminator resistors

(c) a Timer Interrupt facility generating program interrupts at the rate of 2^n Hz, where $0 \leq n \leq 11$. The rate is defined by linking on the Input/Output edge connector

(d) either 4K, or 16K 17-bit words of dynamic RAM with the following characteristics:

  (i) a parity bit is included, together with the necessary parity generation and checking logic

  (ii) the access time (Ac in Sect. 1.8) is 280ns

(iii) the write time (Wt in Sect. 1.8) is 185ns

(iv) the start address of the 4K/16K block is selectable at any multiple of 4K/16K by linking on the input/output edge connector.

(v) all necessary refresh circuitry is provided

(e) standard CR networks to define the failure time-out periods for F100-L and the RAM as 80µs and 10µs respectively.

(f) a circuit to generate a system reset automatically when power is applied

(g) provision for external battery support to be applied to the RAM system to prevent loss of data on power failure.

3.3.2 4K/8K/16K/32K RAM

This memory extension card comprises dynamic RAM and its associated refresh circuitry. A 17th (parity) bit is included along with generation and checking logic. Provision is made for external battery support to the RAM system to prevent loss of data on power failure. The start address of the RAM block can be selected in multiples of the block size by linking on the input-output edge connector.

In a system containing a full 32K words of memory, it is necessary to remove a small number of addresses from the memory system for use as peripheral addresses. Provision is made on this card for one or more 256 word blocks to be optionally masked-out for this purpose.

The access time (Ac in Sect. 1.8) is 280ns.

The write time (Wt in Sect. 1.8) is 185ns.

3.3.3 1K/2K/4K/8K/16K PROM

This card contains fusible link PROM, although mask-programmable ROM can be supplied if required.

The start address of the PROM block is selectable in multiples of the block size by linking on the input-output edge connector.

Provision is made to optionally mask-out one, or more, 256 word blocks of addresses, so that they may be used by peripherals in the system.

The access time (Ac in Sect. 1.8) is 150ns.

3.3.4 Terminal (V24) and Line Printer Interface

The card is designed to interface to any peripheral using the 20mA Current Loop or CCITT V24 serial interfaces (e.g. Teletype, terminal, VDU).

The use of either 10 or 11 unit code and the selection of the baud rate (110 to 9600) is by linking on the input-output edge connector.

The channel number allocated to this Interface is selected by switches on the card.

Output transfers are under DMA control. The program sets up a positive count of the number of characters to be output and then sends a control word to initiate the transfer. As an option two 8-bit characters can be packed into a 16-bit word. A program interrupt is generated when the output has been completed.

Input transfers initiated by the Interface are under DMA control. The program sets up a positive count of the number of characters to be input and then sends a control word to initiate the transfer. As an option two 8-bit characters can be packed into a 16-bit word. A program interrupt is generated when the input has been completed. Alternatively a control character (NL) can be used to generate the interrupt.

Input transfers initiated by the external device result in a program interrupt being generated for each character. The character is accessed from the Status Word location (See Sect. 2.4).

An 8-bit parallel interface is also provided on this card to drive a Data Products 2200 Series line printer using the DMA output facility. The F100 Operating System used in the F100 Development System uses this DMA mode of operation when a line printer is required. Therefore this Interface is mandatory if a line printer is required to work with the F100 Operating System.

Both the Line Printer and the Terminal device can be handled by just one Interface card; the destination of data being determined by a control word.

A Data Products 2200 Series line printer can be driven in a non-DMA manner by the Paper Tape Reader and Punch Interface (see Sect. 3.3.5). but the user must provide his own software driver routine.

3.3.5 Paper Tape Reader and Punch/Line Printer Interface

This card is a single channel Interface transferring data to or from a single peripheral address (memory mapped I/O). The address used is O+ of the 4-word Interface Block (see Sect. 2.4) associated with the channel number which is selected by switches on the card. A character is input every time an instruction reads from the address, and a character is output every time an instruction writes to the address.

The card is designed to interface to a Trend HSR 700 high speed paper tape reader for input transfers and either a Facit 4070 paper tape punch or a Data Products 2200 Series line printer for output transfers.

This Interface must not be used when using a line printer in conjunction with the F100 Operating System.

A Data Products 2200 Series line printer can also be driven in DMA mode by the parallel output interface provided on the Terminal (V24) and Line Printer Interface (see Sect. 3.3.4) as required for use with the F100 Operating System,

3.3.6 Floppy Disk Interface

This card interfaces with up to six Shugart SA 800 Floppy Disk Drives via a Shugart Formatter. The channel number for this Interface is selected by switches on the card.

Format on the disk can be either single density (IBM compatible) or double density. Single-density capacity is 120K words per disk, and double-density capacity is 300K words per disk.

Data transfers use the DMA facility and are initiated by the program sending control information to the Interface. A program interrupt is generated when a complete sector of data has been transferred (64 words for single density and 128 words for double density).

A ROM loader is incorporated within the Interface that enables track 0 (1664 words single density, 4K words double density) to be automatically read down and loaded into the system memory.

3.3.7 Magnetic Tape Cassette Interface

This card interfaces to a Racal P70, or P70N, magnetic tape cassette deck using the Racal Standard 1200 format. The channel number of this Interface can be selected by switches on the card.

Transfers to or from the cassette deck use the DMA facility.

On output the program sends a control word to the Interface to define the number of 16-bit words to be written onto the tape ($1 \leq$ number of words ≤ 128). These words will be output by DMA and written onto the tape as a single block followed by an End of Block marker. The Interface then generates a program interrupt to signify that the transfers have been completed.

On input the program defines the number of 16-bit words it requires to read ($1 \leq$ number of words ≤ 128) by sending a control word to the Interface. These words will then be input using the DMA facility. The Interface then generates a program interrupt. If the program tries to read less words than are contained within the block on the tape, the entire block will be read but only the number of words requested will be transferred. The program Interrupt Status Word will contain a specific indication that this has occurred. If the program tries to read more words than are in the block, only the block will be read and the program interrupt Status Word will indicate that further words are outstanding.

3.3.8 Special Processing Unit: Fast Fourier Transforms

This unit comprises three cards that plug directly into an F100-L Microcomputer module, One of the three cards accesses the I/O Bus and associated control signals by its lower edge-connector.

Interconnection between the three cards is via the upper, input/output edge-connectors using a special harness.

A group of 32 instructions within the External Function decodings of the F100 instruction set is recognised by this unit. Since it is possible to have more than one of these units per system it is necessary to define which group of 32 instructions is recognised by which unit. This is selected by linking across the input/output edge connector of one of the cards.

Relative to the selected first instruction the functions supplied are:

0	Load SPU internal memory
1	Unload SPU internal memory
2	Frequency Shift and Low Pass Filtering
3	FFT Pass - Decimation in Frequency-Forward
4	FFT Pass - Decimation in Frequency-Forward (Interpolated sine/cosine values)
5	FFT Pass - Decimation in Frequency-Inverse
6	FFT Pass - Decimation in Frequency-Inverse (Interpolated sine/cosine values)
7	FFT Pass - Decimation in Time-Forward
8	FFT Pass - Decimation in Time - Forward (Interpolated sine/cosine values)
9	FFT Pass - Decimation in Time-Inverse
10	FFT Pass - Decimation in Time - Inverse (Interpolated sine/cosine values)
11	Re-order FFT input/output
12	Conjugate complex number groups
13	Power calculations on complex number groups
14	Phase calculations on complex number groups
15	Power and phase calculations on complex number groups
16	Multiply complex number groups
17	Integrate (α filter)
18	Scale Internal Memory group
19	Scale External Memory group
20	Clear Internal Memory group

21	Clear External Memory group
22	Multiply (Fractional)
23	Simple Sine/Cosine
24	Simple Arctan
25	Simple Square Root
26	Divide (Fractional)
27-30	Not Allocated
31	Test Mode

The internal memory comprises 2K X 16-bit words and is used locally by the unit. Its contents may be loaded in one instruction (e.g. Load SPU internal memory), operated upon by a succession of further instructions (e.g. FFT passes, re-order, power calculations), and the results output into the system main memory using the Unload internal memory instruction.

Typical times for complete Fourier transforms are:

256 complex points	22mSec
512 complex points	50mSec
1024 complex points	110mSec

The multiply and divide functions are two's complement operations. Multiplication of two 16-bit numbers results in a 32-bit answer. A 32-bit dividend is divided by a 16-bit divisor resultant in a 16-bit quotient and a 16-bit remainder.

Multiplication of two numbers already in the internal SPU memory takes 2.5 µs. Division, again using the internal SPU memory, takes 5.9 µs.

3.3.9 Bus Extension Card

This card contains the Interface Sets necessary to interconnect two I/O Buses. A conventional Bus Extension Unit, as described in Sect. 2.7., enables a range of addresses on a Secondary Bus to be accessed from a Primary Bus. In this card implementation additional F112-L data chips have been added to enable two address ranges on the Secondary Bus to be recognised. This is particularly useful when the Secondary Bus has peripheral addresses that have to be recognised in addition to memory addresses. The selection of these ranges is made by switches on the card.

The two address ranges available on the Secondary Bus must be in the overall range 4K to 32K.

One of the address ranges can be selected in blocks of

256, 512, 1K, 2K, 4K, 8K, and 16K words;

the other range in blocks of

32, 64, 128 and 256 words.

The Primary Bus connections are made via the lower edge connector to the backplane of the module containing the Primary Bus. The Secondary Bus is wired via the upper input-output edge-connector to the backplane containing the Secondary Bus.

3.3.10 Control Handset and Interface

The standard hand-held F100-L Control Handset is connected to the I/O Bus via an Interface card which plugs into the Microcomputer module. A 3-metre ribbon cable connects the Control Handset to the Interface card. The Control Handset is illustrated in fig. 1.4.

Basic control of an F100-L system can be exercised from the Control Handset using the Reset (Rs), RUN and Single Step (S/S) keys.

Numbers, both data and addresses, are entered in octal using the keys 0 - 7. The number is shifted into the LED displays from the right hand side in the normal 'calculator' fashion. The key C (Clear) is used to clear the display.

Loading and monitoring are accomplished using the keys labelled Ad (address), Da (data), PC (program counter), and F/B (function or instruction monitor/bootstrap load). The hexadecimal LED indicates to the user which function the Control Handset is currently performing/displaying.

A ROM Loader is contained within the Interface and is entered into the system main memory by using the F/B key.

System status indication is given by the decimal points of the 7-segment data LEDs. They are:

Pe (Peripheral) Pnl (Panel, i.e. Control Handset} Sr (Store, i.e. memory) F100-L	When lit, signify that the Fail outputs of the relevant parts of the system are set. This is either the result of a reset operation of an operational failure.
Run	When lit, signifies that the F100-L is in a 'Run State'.
Stop	When lit, signifies that the F100-L has stopped.

3.3.11 Customer Wiring Card

This card enables a user to wire up his own circuitry on a card that fits into the standard microcomputer modules.

The card is multilayer with 0v and 5v buried planes connected to a matrix of DIL positions.

Two through-hole plated holes are provided for every DIL pin enabling the DIL pin to be inserted in one hole and interconnecting wires in the other.

Provision is made for an Interface Set to be mounted close to the lower (I/O Bus) edge-connector, with room for an additional 74 16-pin DILs. 14, 18, 24 and 40 pin DILs can also be mounted using the standard grid pattern.

3.4 F100 DEVELOPMENT SYSTEMS

Standard configurations of the F100-L Microcomputer are available to enable system programs to be developed using resident versions of the F100 Support Software.

Three configurations are illustrated in Table 3.1 and uport three different levels of software.

3.4.1 Minimal System

This is designed to enable programs generated by cross-product software, possibly without access to the F100 Simulator, to be checked-out on a standard F100-L Microcomputer system using the Debug package.

3.4.2 Intermediate System

This system provides basic facilities for running the Assembler and Link Editor in addition to the other facilities provided by the Minimal System. With the hardware shown it is recommended that a terminal equipped with dual cassettes is used via the Terminal Interface. The various programs can then be conveniently loaded from cassette, removing the need for paper-tape equipment.

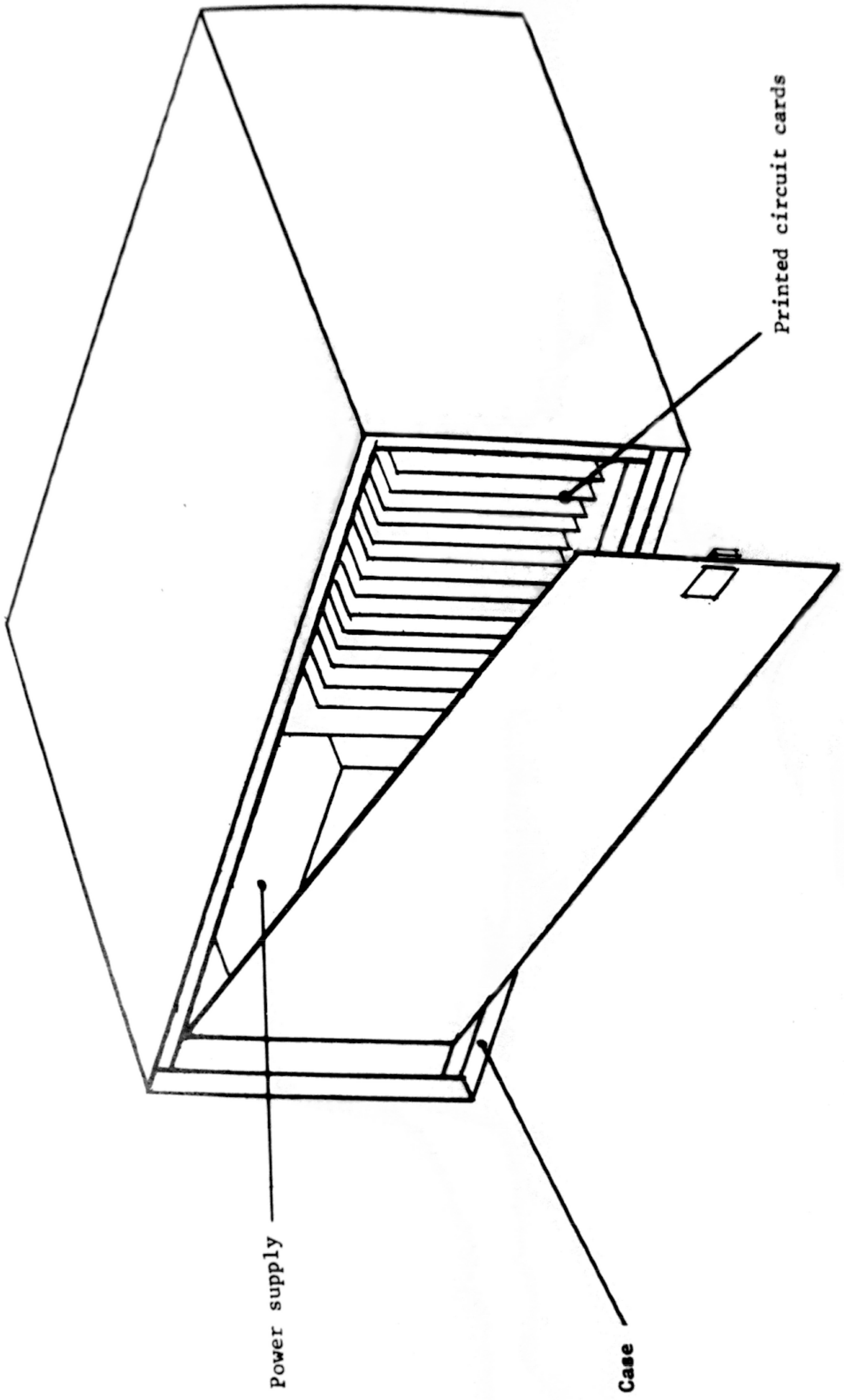
3.4.3 Comprehensive System

The addition of a dual floppy disk to the Intermediate System and the use of a full 32K of memory enables the F100 Operating System to be used. This vastly simplifies the use of the development system by providing a convenient command language for the generation/deletion of files and the operation of the resident software.

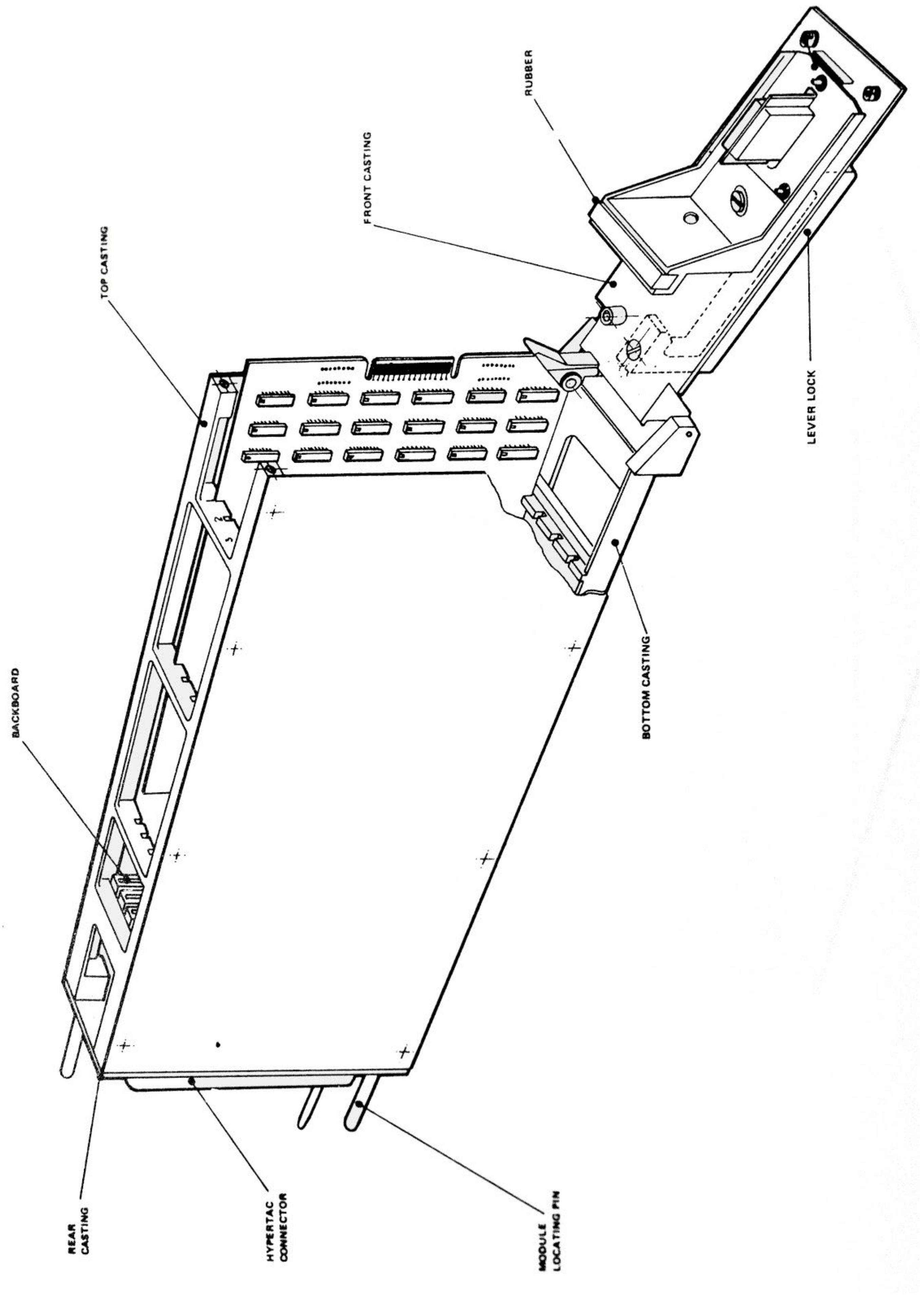
With this hardware configuration the resident F100 CORAL 66 Compiler can also be used.

	HARDWARE	SOFTWARE
MINIMAL SYSTEM	F100-L 4K RAM Control Handset V24 Interface	Bootstrap Loader SCM Debug Package
INTERMEDIATE SYSTEM	F100-L 16K RAM Control Handset V24 Interface	Bootstrap Loader SCM Debug Package Assembler Link Editor Text Editor
COMPREHENSIVE SYSTEM	F100-L 16K RAM Control Handset Terminal Twin Floppy Disk (twin drive, both exchangeable)	Bootstrap Loader SCM Debug Package Assembler Link Editor Text Editor Operating System CORAL 66

F100 Development System   Fig. 3.1

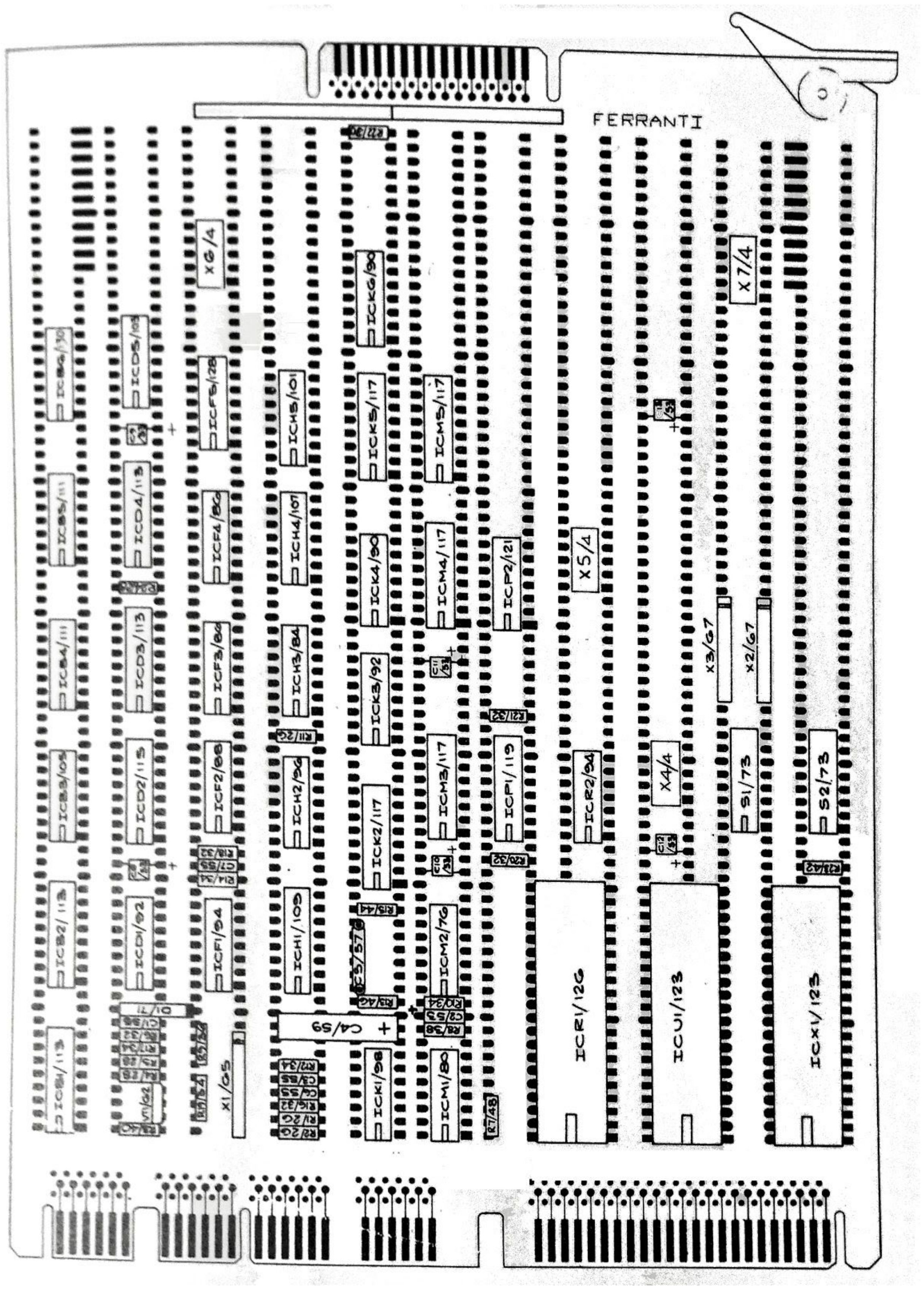


<u>13-Slot Module</u> <u>Fig. 3.1</u>



Ruggedised Module - STEP Fig. 3.2

Monitor/Test
Edge Connector



Upper Edge Connector
(Input-Output)

Lower Edge Connector
(I/O Bus and Control)

231.0mm (9.1in) X 167.4mm (6.6in)

<u>Standard Printed Circuit Card</u> <u>Fig3.3</u>

FERRANTI

For further details contact:
Ferranti Limited,
Microprocessor Marketing Unit
Western Road, Bracknell
Berkshire RG12 1RA
Telephone: Bracknell (0344) 3232
Telex: 848117

Printed in England DSD/777/3