FT-86C and FT-86C/FP
USER'S MANUAL

FORWARD TECHNOLOGY
INCORPORATED

document number 4301000, 9/81

This manual is intended to give the user of the FT-86C and FT-86C/FP a guide to system functionality and also to enable the user to add peripheral, memory and other devices to satisfy specific requirements. The FT-86C and FT-86C/FP give users the capability to quickly add components without needing a detailed understanding of the 8086 or its support chips. For specific device timings we recommend that you refer to the manufacturer's literature listed in Appendix B.

FORWARD TECHNOLOGY INC., 2595 Martin Avenue, Santa Clara 95050
PHONE: (408) 988-2378
TWX: 910-338-2186

TABLE OF CONTENTS

SECTION	TITLE	PAGE
	SPECIFICATIONS	v
1.0	INTRODUCTION	1-1
2.0	INSTALLATION PROCEDURE AND OPTIONS	2-1
2.1	MULTIBUS CONTROL	2-1
2.1.1	MULTIBUS OPTIONS	2-2
2.2	COMMUNICATIONS OPTIONS	2-2
2.2.1	COMMUNICATIONS CLOCK STRAPPING	2-2
2.3	EPROM TYPE	2-9
2.4	WAIT STATE TIMING	2-10
2.5	INTERRUPT STRAPPING	2-11
3.0	THEORY OF OPERATION FT-86C	3-1
3.1	BUS ELEMENTS	3-1
3.1.1	BUS CONTROL	3-2
3.1.2	LOCAL BUS	3-2
3.1.3	I/O BUS	3-2
3.2	MEMORY	3-2
3.2.1	RANDOM ACCESS MEMORY	3-2
3.2.2	PROGRAMMABLE READ ONLY MEMORY	3-2
3.3	TIMING	3-5
3.3.1	CLOCK GENERATOR	3-5
3.3.2	PROCESSOR TIMING	3-5
3.3.3	BUS CONTROL TIMING	3-5
3.3.4	BUS TIMING	3-6
3.4	COMMUNICATIONS	3-7
3.4.1	READ REGISTER FUNCTIONS	3-9
3.4.2	WRITE REGISTER FUNCTIONS	3-9
3.4.3	PROGRAMMING THE WRITE REGISTERS	3-9
3.4.4	PROGRAMMING THE READ REGISTERS	3-10
3.5	INTERRUPT CONTROL	3-12

TABLE OF CONTENTS (Cont.)

SECTION	TITLE	PAGE
4.0	BREADBOARD INTERFACE	4-1
4.1	MEMORY ADDRESS BUS	4-1
4.2	I/O ADDRESS BUS	4-1
4.3	DATA LINES	4-1
4.3.1	MEMORY DATA BUS	4-1
4.3.2	I/O DATA BUS	4-1
4.4	I.O SELECT LINES	4-3
4.5	CHIP SELECT DECODING	4-3
4.5.1	EPROM CHIP SELECTS	4-3
4.5.2	RAM SELECT LINES	4-5
5.0	FIRMWARE	5-1
5.1	CONTROLFORTH	5-1
5.2	MONITOR COMMANDS	5-1
5.2.1	FILL	5-1
5.2.2	SUBSTITUTE	5-2
5.2.3	MOVE	5-2
5.2.4	MATCH	5-2
5.2.5	P! AND WP!	5-3
5.2.6	P@ AND WP@	5-3
5.2.7	GO	5-3
5.2.8	RECEIVE	5-4
5.2.9	SEND	5-4
5.2.10	DUMP	5-4
5.3	MONITOR COMMANDS WITH EXPLICIT SEGMENT ADDRESSES	5-5
5.3.1	SMOVE	5-5
5.3.2	SMATCH	5-5
5.4	MONITOR CONTROL COMMANDS	5-5
5.4.1	SEGMENT	5-5
5.4.2	TIMES . . . RUN	5-6
5.4.3	MISMATCHES	5-6

TABLE OF FIGURES

FIGURE NUMBER	TITLE	PAGE
2-1	FT-86C SELECTED PAD LOCATIONS	2-3
2-2	MULTIBUS CONTROL STRAPPING	2-5
2-3	CLOCK GENERATOR STRAPPING	2-6
2-4	MODEM CONTROL AND COMMUNICATIONS STRAPPING	2-7
2-5	EPROM STRAPPING	2-9
2-6	WAIT STATE STRAPPING	2-10
2-7	INTERRUPT STRAPPING	2-12
3-1	BLOCK DIAGRAM FT-86C/FP	3-3
3-2	ADDRESS TIMING CONSTRAINTS	3-7
3-3	USART INTERNAL STRUCTURE	3-8
3-4	READ REGISTER BIT FUNCTIONS	3-10
3-5	WRITE REGISTER BIT FUNCTIONS	3-11
4-1	MEMORY ADDRESS AND DATA BUS'S	4-2
4-2	I/O ADDRESS PADS	4-3
4-3	I/O SELECT PADS	4-4
4-4	EPROM AND RAM CHIP SELECT LINES	4-5

TABLE OF TABLES

TABLE NUMBER	TITLE	PAGE
2-1	TELECOMMUNICATION PAD ASSIGNMENTS	2-8

APPENDICES

APPENDIX NUMBER	TITLE	PAGE
A	CONTROLFORTH GLOSSARY	A-1
B	RECOMMENDED READING	B-1
C	FT-86C PIN ASSIGNMENTS	C-1

SPECIFICATIONS

PHYSICAL	Width:	12.0"	(30.48cm)
	Height:	6.75"	(17.15cm)
	Depth:	.27"	(.83cm)
	Weight:	13.0 oz. approx.	(370 gm)
	Shipping Weight:	20.0 oz. approx.	(570 gm)
	Form Factor:	IEEE P-796	

ENVIRONMENTAL	Operating Temperature:	$0^{o}C$ to $55^{o}C$
	Storage Temperature:	$-10^{o}C$ to $70^{o}C$
	Relative Humidity:	90% non-condensing

ELECTRICAL CHARACTERISTICS		5V $\pm$5%	12V $\pm$10%	-12V $\pm$10%
	FT-86C	2.75 A	40 mA	35 mA
	FT-86C/FP	3.25 A	40 mA	35 mA

SYSTEM CLOCK 5.0 MHz $\pm$0.1%

CONNECTORS

BUS:

86 pin 0.156" center (0.4cm)
Viking 3KH43/9AMK12

SERIAL I/O:

50 pin header type
AUGAT 110-50001-102

ELECTRICAL INTERFACE

P-796 Bus TTL compatible
Interrupt request TTL compatible
Serial I/O RS-232C compatible

PROCESSORS

FT-86C

Intel 8086 or equivalent
Space for 8087-co-processor provided
Direct addressing to 1 Mbyte of memory
Bit, byte, word and block operation

SPECIFICATIONS (Cont.)

PROCESSORS (Cont.)

24 operand addressing modes

Fourteen (14) registers

8 and 16-bit signed and unsigned arithmetic

FT-86C/FP

Intel IAPX 86/20 consisting of an Intel 8086 and an Intel 8087 co-processing configuration

Direct addressing of up to 1 Mbyte of memory

Bit, byte, word and block operations

24 operand addressing modes

Fourteen registers in the 8086. Eight 80-bit numeric data registers and six 16-bit registers in 8087

Single and double precision floating point arithmetic, BCD arithmetic and transcendental functions

PROCESSOR WORD SIZE

FT-86C

Instruction: 8, 16, 24, 32, 40 or 48-bits

Data: 8 and 16-bits

FT-86C/FP

Instruction: 8, 16, 24, or 32-bits

Data: Internal up to 80-bits

INSTRUCTION CYCLE TIME

FT-86C

Typical instruction cycle:

1.0 microsecond

FT-86C/FP

Typical instruction cycles:

Multiply double precision - 27 microseconds

Square root - 36 microseconds

Divide single precision - 39 microseconds

Tangent - 90 microseconds

SECTION 1.0

INTRODUCTION

The FT-86C is a Multibus compatible single board 16-bit computer offering a customizing area. The processor is an Intel 8086 with the 8087 Numeric Data Processor available as an option. The customizing area allows the user to add peripheral and memory chips to meet the user's specific needs.

The FT-86C and FT-86C/FP provide ample drive current on local busses to support most types of peripheral or memory chips. Spare select lines are provided for user-added PROM, RAM or I/O devices. All pads in the customizing area are drilled to take 0.025" square wire wrap pins.

The customizing area may be used for up to 27 16-pin chips and 5 40-pin chips, or many combinations of 0.3" wide and 0.6" wide devices.

The optional controlFORTH monitor is an implementation of FORTH with monitor command extensions. It provides the user with a real time programming language and also with a powerful testing and debugging tool. The FORTH supplied with the 8087 numeric data processor option contains additional extensions to facilitate the use of the 8087.

SECTION 2.0

INSTALLATION PROCEDURE AND OPTIONS

The FT-86C is shipped with the following options and straps. Option straps and IC's can be located by using the x-y coordinate system etched on the PCB. Along the length of the PCB is a set of alphabetic coordinates (A, B, D, etc.). Along the width of the PCB, a set of numeric coordinates (1, 2, 3, etc.) can be found. These coordinates form an x-y grid so that straps and IC's can be located rapidly.

2.1 MULTIBUS CONTROL

The FT-86C is optioned to act as bus master with the highest priority. This is done by a strap (pad 1 to pad 2 at board location 8J) which holds pin 9 of the 8289 to ground. (Refer to Figure 2-1 for pad locations and Figure 2-2 for specific strapping information.) Cutting the ground strap allows the FT-86C to respond to the Multibus bus priority in (BPRN) signal.

The FT-86C can be used in either parallel or serial bus priority arbitration schemes. For parallel priority arbitration external logic must be provided.

The 8289 bus arbiter has the signal ANYRQST option strapped to ground via pads 9 and 10. (Refer to Figure 2-1 for pad locations and Figure 2-2 for specific strapping information.) In this mode the FT86C will not release the Multibus unless it has completed its immediate bus access requirements. Cutting the ground trace between pads 9 and 10, and strapping ANYRQST to the adjacent option hole (pad 8) will hold ANYRQST high. The FT-86C will now relinquish the bus as soon as the current bus transfer cycle (if any) has been completed.

	ANYRQST	BPRN	CRQLCK	FUNCTION
FACTORY STANDARD	Low	Low	Low	Gives the FT-86C priority. It will not relinquish the bus.
USER OPTIONS	Low	Driven by pin 15 on Multibus	Low	The FT-86C will relinquish the bus only to a higher priority master.
	Low	Driven by pin 15 on Multibus	High	The FT-86C relinquishes the Multibus after each transfer cycle.

2.1.1 MULTIBUS OPTIONS (Figure 2-2)

The FT-86C is factory optioned to provide bus clock and common clock to the Multibus. The clocks can be disabled by cutting the straps between pads 6 and 7 and 11 and 12 (Refer to Figure 2-2 for pad locations). The FT-86C will now draw its bus clock from the Multibus.

2.2 COMMUNICATIONS OPTIONS

Most of the communications options for the FT-86C are software controlled. The strapping options for the communications channels allow the user to select the communications clock source and speed for each channel. The user can also select local mode (direct connection to a terminal) or select modem operation through a combination of software and hardware strapping options.

2.2.1 COMMUNICATIONS CLOCK STRAPPING

The communications clock for the USART can come from either of two sources: the on-board clock generator (used for asynchronous protocols) or from an external clock source such as a modem (used for synchronous protocols). The user may select the clock source and speed by removing or installing jumpers. The FT-86C is strapped at the factory for 300 baud operation on both communications channels. Figure 2-1 is a pictorial representation of the FT-86C PCB showing the jumper pad locations and numbers for option strapping. Figure 2-3 shows the telecommunications clock generator and the associated pads for each clock frequency. An example of how to strap Channel "B" for 9600 baud is given on page 2-6. All baud rates assume that the USART is initialized to ÷ 64 clock mode on the appropriate channel.

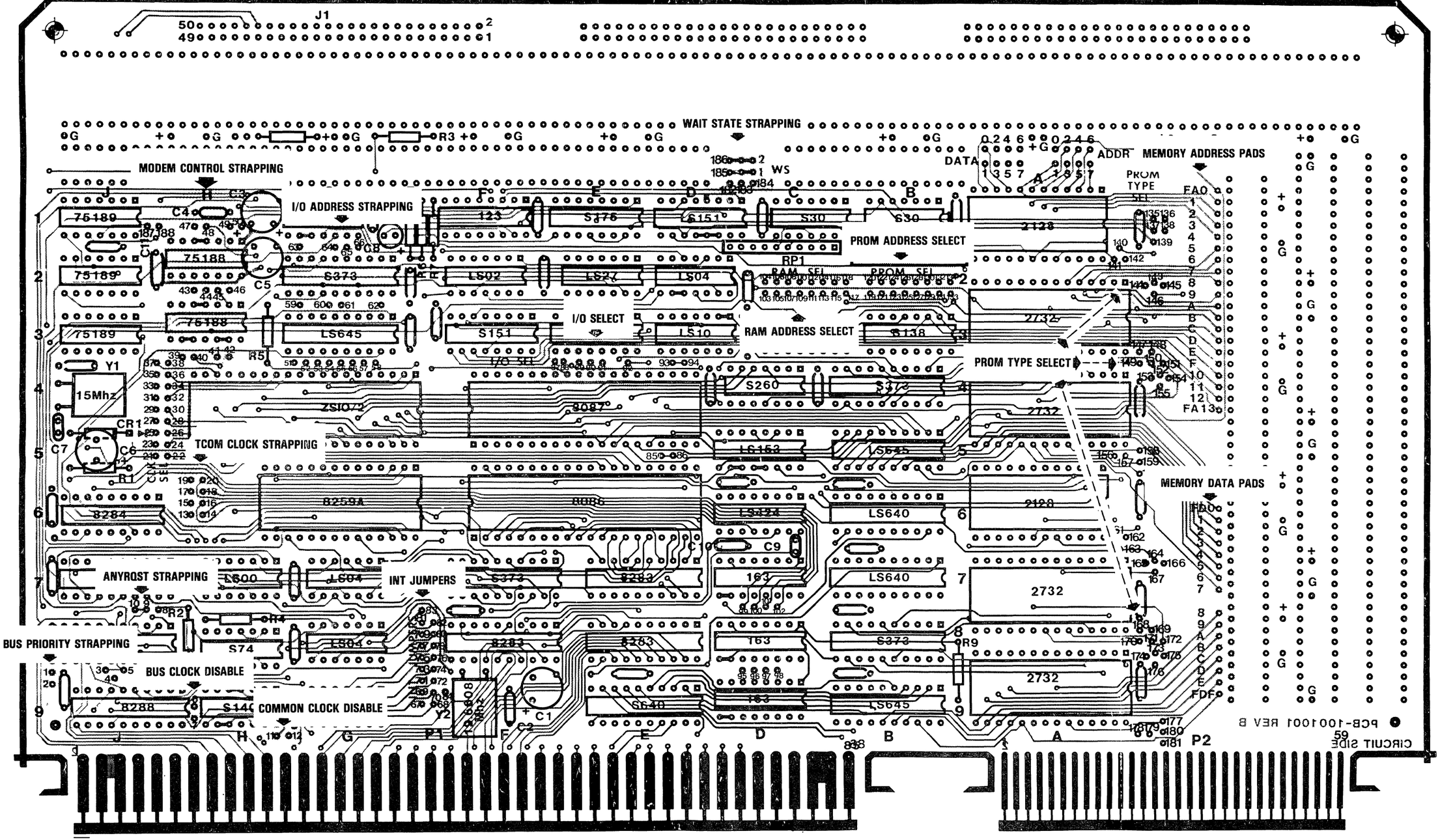


Figure 2-1. FT-86C SELECTED PAD LOCATIONS

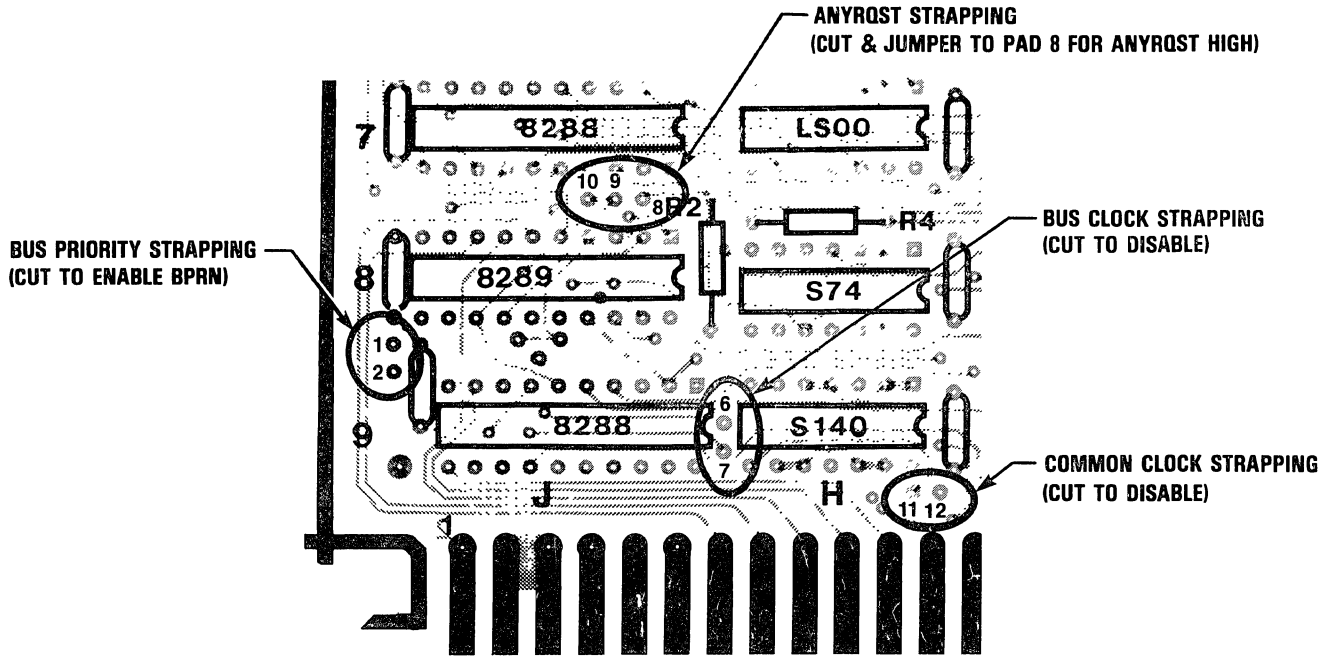


Figure 2-2. MULTIBUS CONTROL STRAPPING

EXAMPLE 1 - Change Channel "B" to 9600

1. Cut trace between IC 7D pin 11 and pad 99, to free trace from pad 99 to pad 16.
2. Add jumper between pads 98 and 99 to place 9600 baud clock on pad 16.
3. Cut jumpers between pads 15, and 17 to separate Channel "A and B" clocks.
4. Jumper pad 16 to pad 15, make sure jumper between 13 and 15 is intact.

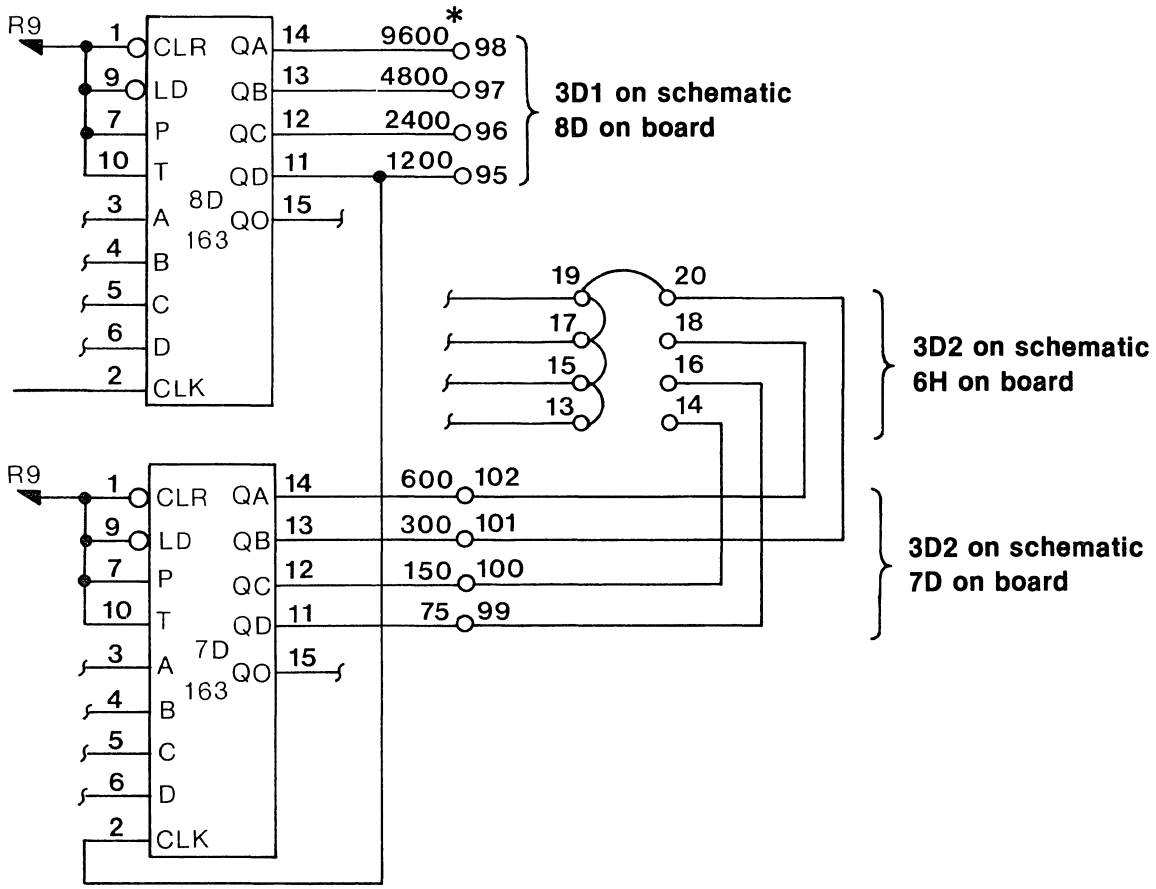


Figure 2-3. CLOCK GENERATOR STRAPPING

* NOTE: Labeled baud are for 64 USART clock mode.

The FT-86C can also be strapped to operate from an external clock source. Figure 2-4 is a schematic representation of the telecommunications circuitry.

EXAMPLE 2 - OPERATE CHANNEL "B" FROM MODEM CLOCK

1. Remove jumpers between pads 13, 15, and 17 removing the internal clock from Channel "B".
2. Jumper pad 27 to 28.
3. Jumper pad 29 to 30. Clock now comes from J1 pins 44 for TXCB and 46 for RXCB.

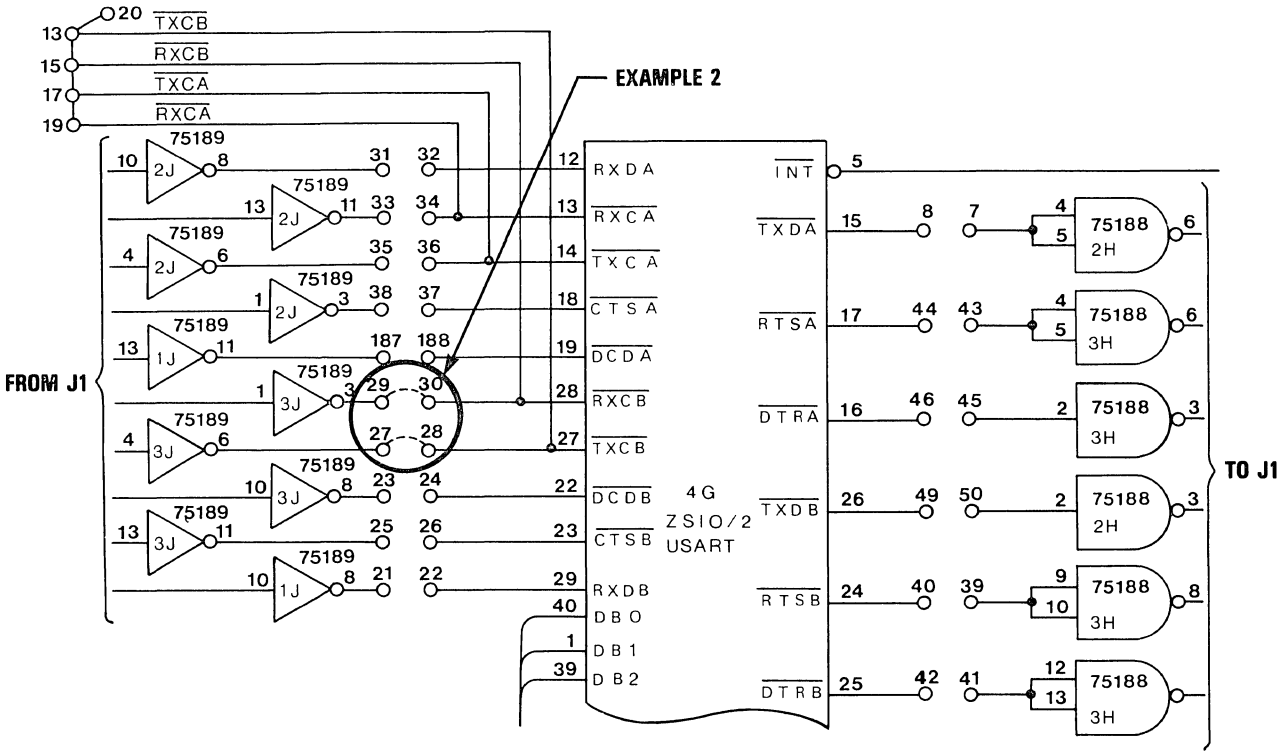


Figure 2-4. MODEM CONTROL AND COMMUNICATIONS STRAPPING

Additional straps may be needed if modem control signals are to be used. The complete list of the telecommunications option pads can be found in Table 2-1. By refering to this list the user should have no difficulty in strapping the telecommunications interface for his application.

TABLE 2-1. TELECOMMUNICATION PAD ASSIGNMENTS

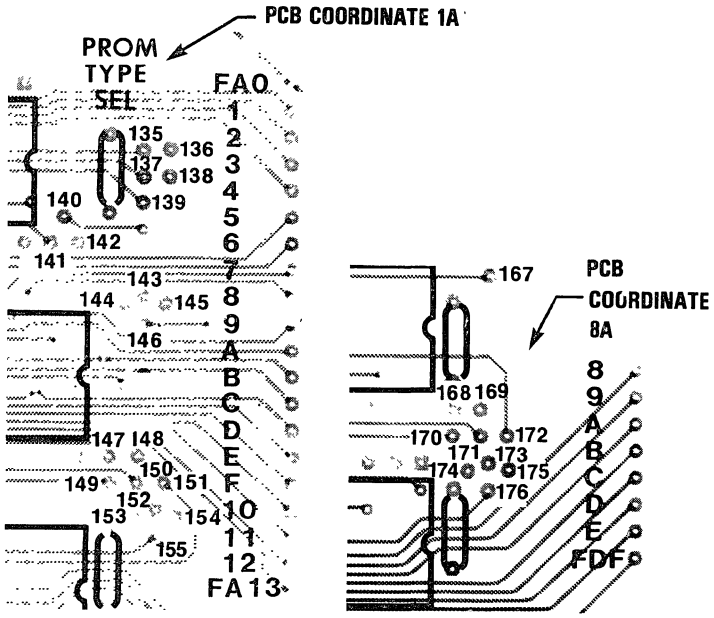
PAD #	SIGNAL NAME	SOURCE/DESTINATION
13	TXCB Transmit Clock Channel "B"	(To USART)
14	150 Baud TCOM Clock	(Option Strap)
15	RXCB Receive Clock Channel "B"	(To USART)
16	75 Baud TCOM Clock	(Option Strap)
17	RXCA Receive Clock Channel "A"	(To USART)
18	600 Baud TCOM Clock	(Option Strap)
19	TXCA Transmit Clock Channel "A"	(To USART)
20	300 Baud TCOM Clock Option Strappable to: 1200, 2400, 4800, 9600	
21	RXDB Receive Data Channel "B"	(From Driver)
22	RXDB Receive Data Channel "B"	(To USART)
23	DCDB Data Carrier Detect Channel "B"	(From Driver)
24	DCDB Data Carrier Detect Channel "B"	(To USART)
25	CTSB Clear to Send Channel "B"	(From Driver)
26	CTSB Clear to Send Channel "B"	(To USART)
27	TXCB Transmit Clock Channel "B"	(From Driver)
28	TXCB Transmit Clock Channel "B"	(To USART via pad 13)
29	RXCB Receive Clock Channel "B"	(From Driver)
30	RXCB Receive Clock Channel "B"	(To USART via pad 15)
31	RXDA Receive Data Channel "A"	(From Driver)
32	RXDA Receive Data Channel "A"	(To USART)
33	RXCA Receive Clock Channel "A"	(From Driver)
34	RXCA Receive Clock Channel "A"	(To USART via pad 17)
35	TXCA Transmit Clock Channel "A"	(From Driver)
36	TXCA Transmit Clock Channel "A"	(To USART via pad 19)
37	CTSA Clear to Send Channel "A"	(To USART)
38	CTSA Clear to Send Channel "A"	(From Driver)
39	RTSB Request to Send Channel "B"	(To Driver)
40	RTSB Request to Send Channel "B"	(From USART)
41	DTRB Data Terminal Ready Channel "B"	(To Driver)
42	DTRB Data Terminal Ready Channel "B"	(From USART)
43	RTSA Request to Send Channel "A"	(To Driver)

TABLE 2-1. TELECOMMUNICATION PAD ASSIGNMENTS (Cont.)

PAD #	SIGNAL NAME	SOURCE/DESTINATION
44	$\overline{\text{RTSA}}$ Request to Send Channel "A"	(From USART)
45	$\overline{\text{DTRA}}$ Data Terminal Ready Channel "A"	(To Driver)
46	$\overline{\text{DTRA}}$ Data Terminal Ready Channel "A"	(From USART)
47	TXDA Transmit Data Channel "A"	(To Driver)
48	TXDA Transmit Data Channel "A"	(From USART)
49	TXDB Transmit Data Channel "B"	(From USART)
50	TXDB Transmit Data Channel "B"	(To Driver)
187	$\overline{\text{DCDA}}$ Data Carrier Detect Channel "A"	(From Driver)
188	$\overline{\text{DCDA}}$ Data Carrier Detect Channel "A"	(To USART)

2.3 EPROM TYPE

When ordered with firmware, chip locations 3A and 7A will be strapped for the appropriate EPROM type. Locations 4A and 9A are capable of being user optioned for 2532, 2732, or 2764 parts. A gate must be added to use 2764 parts in locations 4A and 9A. If 2732 or 2532 parts are used they should be left justified in the EPROM pads, (i.e. pin 1 of the 2732 or 2532 should be put in pin 3 of the EPROM pad and pin 24 into pin 26 of the pad).



To strap sockets 3A and 7A for the following devices strap as indicated:

2532 PAD# TO PAD#	2764 PAD# TO PAD#
121-122	120-121
135-136	122-123
147-149	143-146
150-153	139-138
148-151	155-151
169-172	176-172
168-170	147-149
171-174	148-150
	168-170
	169-171

Figure 2-5. EPROM STRAPPING

2.4 WAIT STATE TIMING

Wait state timing is selected by straps at board location 1D. Either one or two wait states may be inserted for RAM, EPROM or I/O. The selection of the number of wait states is dependent on the speed of the slowest device in each category. Refer to Figure 2-6 for strap locations.

To calculate the number of wait states required for a chip, take the response time of the chip and subtract 400 ns. Divide the result by 200 ns (the period of each wait state) and use the next highest multiple.

The factory settings are:

RAM	1 wait state	Pad 183 to 185
EPROM	1 wait state	Pad 182 to 185
I/O	1 wait state	Pad 184 to 185

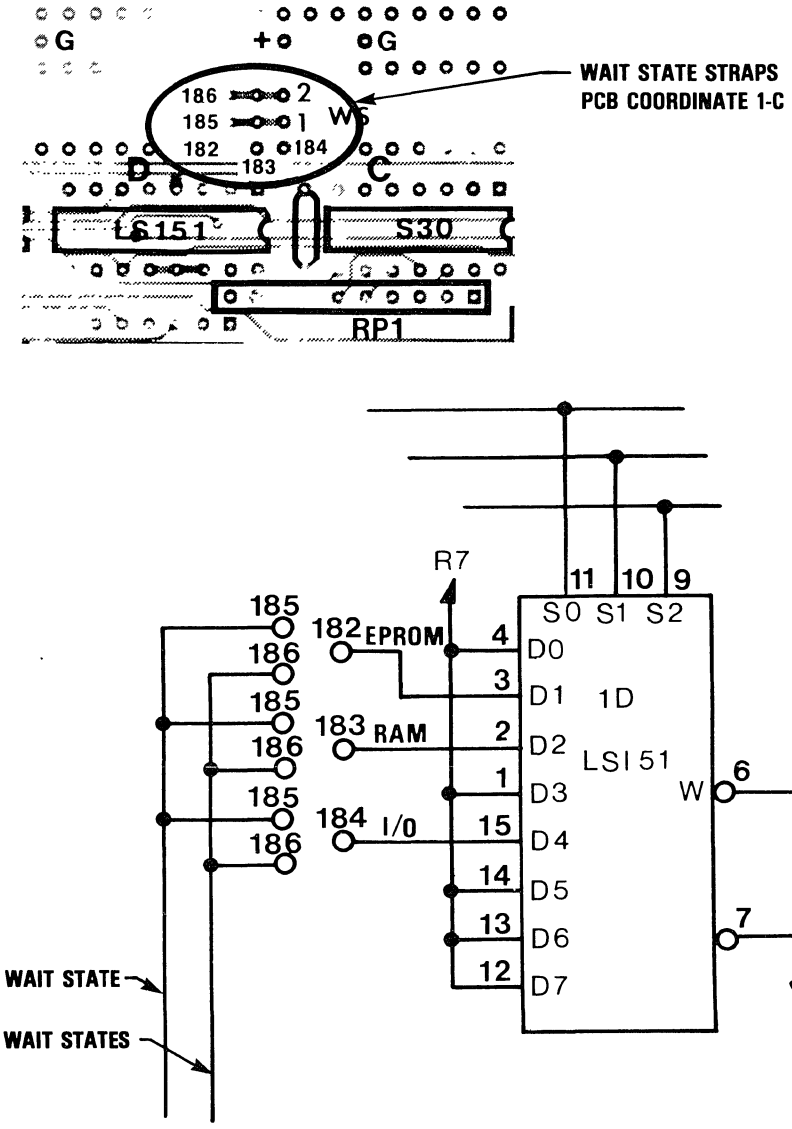


Figure 2-6. WAIT STATE STRAPPING

2.5 INTERRUPT STRAPPING

Interrupts from the Multibus are available on the following pads:

INTERRUPT PAD ASSIGNMENTS

PAD #	SIGNAL NAME	SOURCE/DESTINATION
67	IR4	(To PIC)
68	IR4	(From Multibus Driver)
69	IR7	(To PIC)
70	IR7	(From Multibus Driver)
71	IR6	(To PIC)
72	IR6	(From Multibus Driver)
73	IR5	(To PIC)
74	IR5	(From Multibus Driver)
75	IR3	(To PIC)
76	IR3	(From Multibus Driver)
77	IR2	(To PIC)
78	IR2	(From Multibus Driver)
79	IR1	(To PIC)
80	IR1	(From Multibus Driver)
81	IRØ	(To PIC)
82	IRØ	(From Multibus Driver)
83	INT	(From USART)
84	INTN	(From Optional 8087)

Refer to Figure 2-7 for interrupt pad locations.

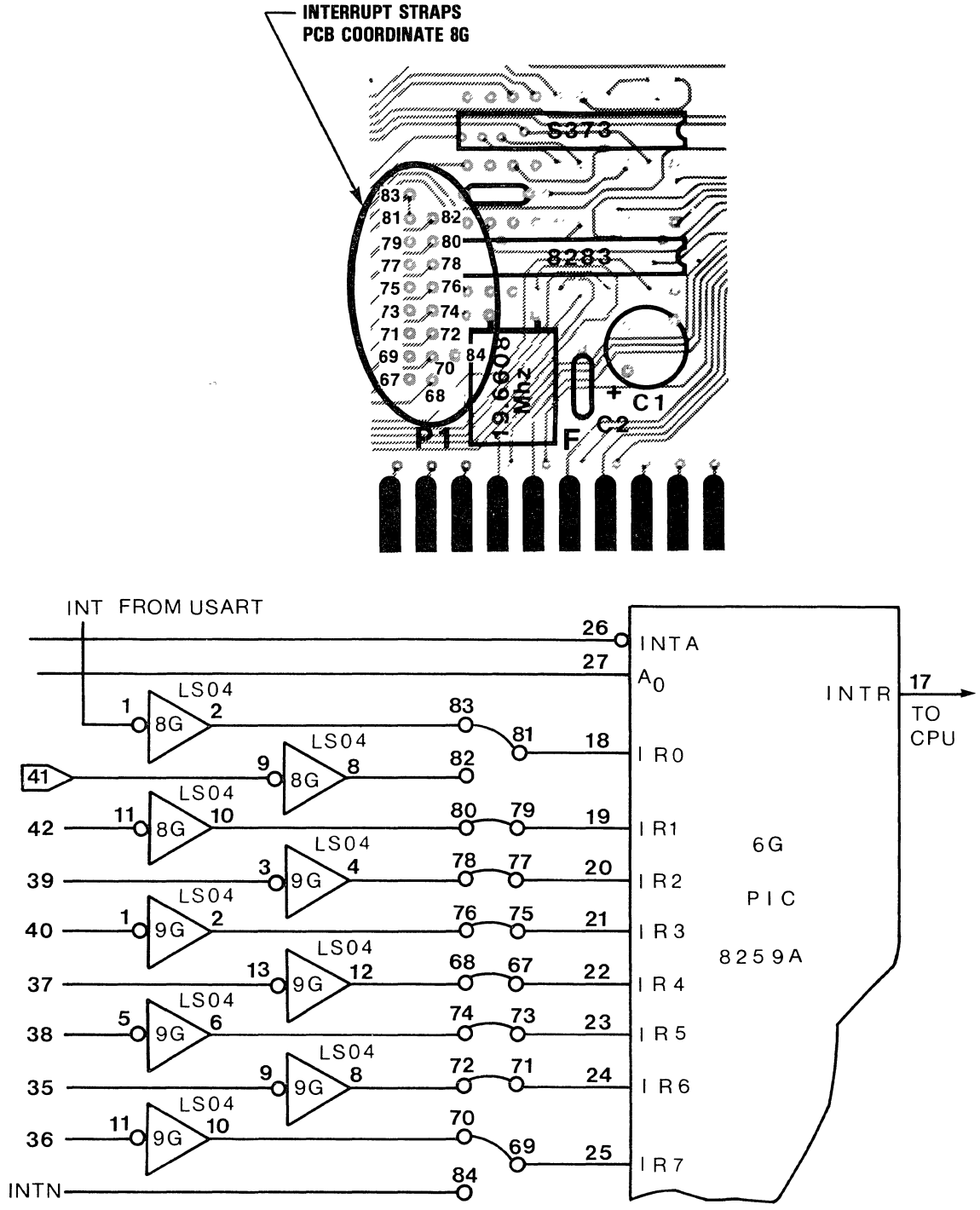


Figure 2-7. INTERRUPT STRAPPING

SECTION 3.0

THEORY OF OPERATION FT-86C

The FT-86C processor is the Intel 8086 5 MHz 16-bit microprocessor. The 8086 communicates to the outside world via a 20-bit wide multiplexed address and data bus, i.e., addresses and data exist on the same pins but at different times. The separation of data and address values is achieved by using signals derived from the 8086 status lines. Eleven additional 8086 lines provide the timing and control interfaces.

Internally, the 8086 can be considered as three major elements: the Bus Interface Unit (BIU), the Execution Unit (EU) and the timing and control unit.

The bus interface unit operates asynchronously to the execution unit. The BIU controls an internal 6 byte long instruction queue. The BIU will prefetch instructions from memory whenever there are 4 bytes or less in its internal queue and the executive unit doesn't require use of the bus. The BIU has access to 5 of the 8086's 16-bit registers.

The execution unit is not directly involved with bus management. The execution unit executes instructions taken off the internal 6 byte instruction queue that were prefetched by the BIU. When the EU requires immediate access to the bus, it does so via the BIU.

The control and timing unit provides status information to external devices in addition to the EU and the BIU. The processor status lines S0, S1 and S2, together with the processor clock, are provided to the external bus control elements to enable demultiplexing of the address and data lines from the 8086. The bus control elements also decode the status lines into the appropriate operational commands.

The 8087 numerical data processor operates in a close coupled configuration with the 8086. The 8087 can execute instructions in parallel with the 8086. The 8087 provides trigonometric, logarithmic, and exponential functions in addition to its arithmetic processing capabilities. The 8087 conforms to the proposed IEEE Floating Point Standard.

Internally, the 8087 consists of two units: a control unit and a numeric execution unit.

The 8087 control unit maintains synchronization with the 8086 by monitoring the 8086 status lines S0, S1, S2, and S6. The 8087 control unit moitors the data bus to obtain 8087 specific instructions.

The numeric execution unit has a register stack of 8 80-bit data registers which are used for computation. Instructions can addres the data registers either implicitly or explicitly.

3.1 BUS ELEMENTS

There are four major elements within the FT-86C bus system: bus control, local bus, I/O bus and the Multibus. Figure 3-1 is a block diagram of the FT-86C.

3.1.1 BUS CONTROL

Bus control is implemented with three LSI chips. The on-board bus and the I/O bus are controlled by an Intel 8288 bus controller. The Multibus is controlled by a second 8288. Selection of which bus controller to use is made through an Intel 8289 bus arbiter. The 8289 resolves access contention to the Multibus when operating in a multi-master environment.

3.1.2 LOCAL BUS

The 20-bit memory addresses output by the 8086/8087 are always latched on-board. A range test is then carried out by the memory decoding logic to determine if this is within the local (on-board) address range. If the address is not within this range the bus arbiter contends for access to the Multibus.

The local bus controller is disabled and when access to the Multibus is granted the bus arbiter enables the Multibus bus controller. If the address is a valid local address the local bus controller is enabled and issues the appropriate commands and enable signals.

3.1.3 I/O BUS

The local I/O bus consists of the low order 8-bits of both the data and address lines. It is activated for an input or output operation. The I/O bus is only active for local I/O addresses in the range 00 to 3F Hex.

3.2 MEMORY

The on-board memory resides in two overlapped 64 Kbyte address areas.

F0000 to FFFFF Hex

00000 to 0FFFF Hex

An address of Hex FFFF0 will also address Hex 0FFF0. This overlaid 64 Kbyte area is decoded into two sections of 32 Kbytes for EPROM and 32 Kbytes for RAM.

3.2.1 RANDOM ACCESS MEMORY

The FT-86C comes with 4 Kbytes of RAM. This is at addresses Hex 00000 to 00FFF, and also Hex F0000 to F0FFF. The RAM is configured with 2 x 8 Kbytes 200ns static RAMs.

Decoding is provided for up to seven additional pairs of RAMs up to a maximum of 32 Kbytes.

3.2.2 PROGRAMMABLE READ ONLY MEMORY

Four configuration pads are provided for EPROMs. Each pad may be configured for 2532, 2732 or 2764 parts. Decoding is provided for four pairs of 32 Kbit EPROMs. One chip select line is provided for one pair of 2764 EPROMs.

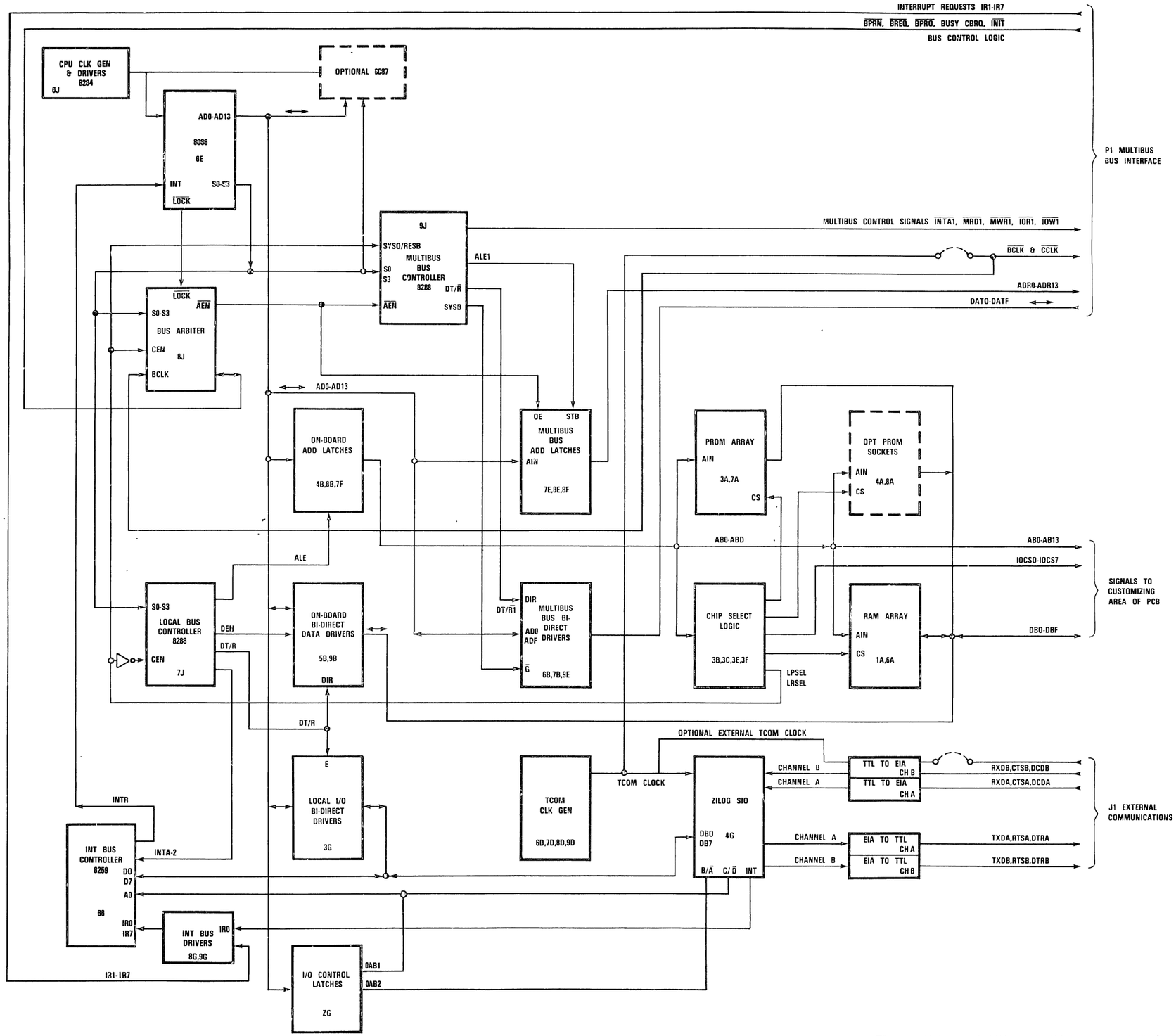


Figure 3-1. BLOCK DIAGRAM FT-86C/FP

If the FORTH monitor is ordered with the FT-86C it will be in 32 Kbit EPROMs located at board positions 3A and 7A. The FORTH monitor resides at memory addresses Hex FE000 through Hex FFFFF, and also at addresses Hex 0E000 through, Hex 0FFFF.

If the 8087 numeric data processor option is ordered, an enhanced FORTH system is supplied in EPROM. The enhancements follow the guidelines of the proposed standards committee working group version of Floating Point FORTH.

3.3 TIMING

Two timing elements are used in the FT-86C: the processor clock and the bus clock.

The processor clock is generated by an Intel 8284. The oscillator input is 15 MHz. The 8284 divides by three and provides a 5 MHz 33% duty cycle clock to the processor and to the bus control elements.

The bus clock can either be generated by the FT-86C and fed onto the Multibus or can be driven via the Multibus from another bus master. The bus clock is used by the bus arbiter in its bus contention circuits and also to synchronize its output commands to the bus controllers.

3.3.1 CLOCK GENERATOR

In addition to providing the processor clock, the 8284 synchronizes and controls the READY and RESET lines to the 8086/8087. Generation of the Multibus initialization signal $\overline{\text{INIT}}$ holds the RESET line active.

The READY line to the 8086 is controlled by two pairs of input signals on the 8284. One pair of inputs are used for controlling wait states for on-board devices, the other pair is used for external bus control, i.e., Multibus.

3.3.2 PROCESSOR TIMING

The 8086 processor cycle operates in a minimum of four clock cycles called T1, T2, T3, and T4. Depending on the speed of attached memory or I/O devices a variable number of wait states may be inserted between processor clock cycles T3 and T4 (e.g., T1, T2, T3, Tw . . . Tw, T4.)

The FT-86C provides separately strappable wait states for on-board I/O, RAM and EPROMs. Multibus access, being asynchronous, will automatically result in 0 to N wait states being inserted. The number of wait states inserted depends on bus contention and arbitration and also on the access time of the specific device or memory type accessed.

3.3.3 BUS CONTROL TIMING

The three elements that make up the bus control section derive their timing from the processor clock and the processor status lines (S0, S1, and S2) to

indicate what function is going to be performed during the current T1 to T4 cycle. This is done at T1. The 8086 also places the address on the multiplexed bus at this time.

Both bus controllers (8288s) use the status lines and the processor clock to generate a pulse (ALE) to latch the address into both the local bus and the Multibus address drivers. The Multibus address drivers do not at this stage have their outputs enabled.

The output of the local address drivers is decoded by the local PROM, RAM and I/O decoders to establish whether this address falls within the on-board address range. If it does, a signal is generated and input to the bus arbiter to indicate a resident bus access only. The Multibus address latches are not output enabled and the Multibus bus controller is held disabled. If the address is not within the resident address space, the signal to the 8289 bus arbiter is raised and the 8289 contends for the Multibus. As soon as the 8289 has gained control of the Multibus, the Multibus address drivers are enabled as is the Multibus bus controller.

At T2 time the 8086 floats its multiplexed address/data lines preparatory to outputting or inputting data. If the resident bus controller is enabled, it will now generate the appropriate command which has been decoded from the processor status lines. If the Multibus bus controller is enabled, the appropriate commands are issued to the Multibus and the resident bus controller is held disabled.

At T3 the appropriate control signals are issued from whichever bus controller is active to condition one of the sets of data bus transceivers. The control signals will be held active through T3 and Twait, where Twait may be 0 up to N. The number of Twaits is dependent on the speed of the addressed device.

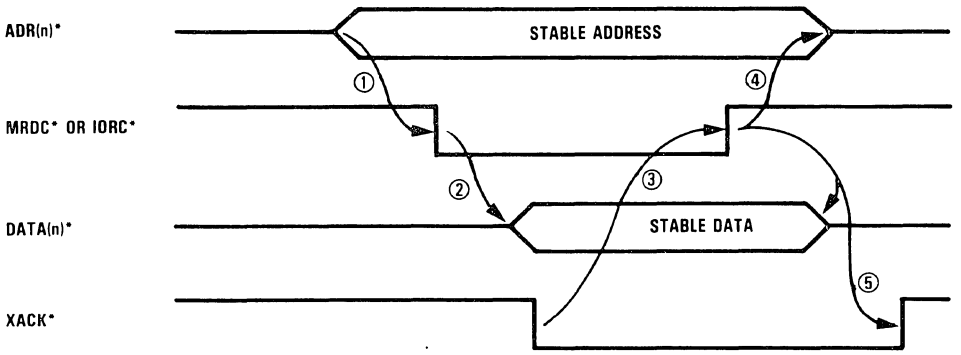
When addresses are within resident bus address space, the FT-86C allows the user to strap select separate wait states for EPROM, RAM and I/O. When addresses are not within the resident bus space, Twait will be issued until the addressed Multibus device responds with an acknowledgement (XACK).

At T4 time the 8086 floats its address/data lines preparatory to issuing a new address at the following T1. The commands are terminated as are the control signals. The processor status lines (S0, S1, and S2) all go inactive.

3.3.4 BUS TIMING

Although the Multibus is an asynchronous bus, two clock lines are present on the bus -- bus clock and constant clock. The Multibus also has certain timing constraints regarding the relationship of the address, data and command presentation. Bus clock is used to synchronize bus arbitration. Enabling of the Multibus address drivers (AEN) is synchronized with bus clock; however, the disabling of the Multibus address drivers is synchronized with T4 of the processor clock. Bus clock can, in theory, be any frequency; however, the lower the frequency the longer the Multibus access arbitration time. The FT-86C generates a bus clock frequency of 9.83 MHz.

Constant clock is provided to the Multibus for general use. It is not specifically related to the timing of bus clock or to the timing of other bus signals. The FT-86C can provide a 9.83 MHz constant clock.



① ADDRESS SETUP TIME: 50 NANOSECONDS MINIMUM.
② TIME REQUIRED FOR SLAVE TO GET DATA ONTO BUS IN ACCORDANCE WITH SETUP TIME REQUIREMENT. XACK* CAN BE ASSERTED AS SOON AS DATA IS ON BUS.
③ TIME REQUIRED FOR MASTER TO REMOVE COMMAND.
④ ADDRESS AND DATA HOLD TIME: 50 NANOSECONDS MINIMUM.
⑤ XACK* AND DATA MUST BE REMOVED FROM THE BUS A MAXIMUM OF 65 NANOSECONDS AFTER THE COMMAND IS REMOVED.

Figure 3-2. ADDRESS TIMING CONSTRAINTS

3.4 COMMUNICATIONS

Two independent communications ports are provided via a Zilog ZSIO USART. Each port can be configured via software to operate in several different modes. Two full sets of RS-232C modem control signal drivers are provided, allowing modems to be attached to these ports. Access to these I/O ports is achieved through the FORTH words P@ and P!. These words are described in Section 5.0. All port programming and I/O is accomplished using these two words.

BASE ADDRESS (Hex)	ZSIO PORT
0000	Channel "A" Data
0002	Channel "A" Control Registers
0004	Channel "B" Data
0006	Channel "B" Control Registers

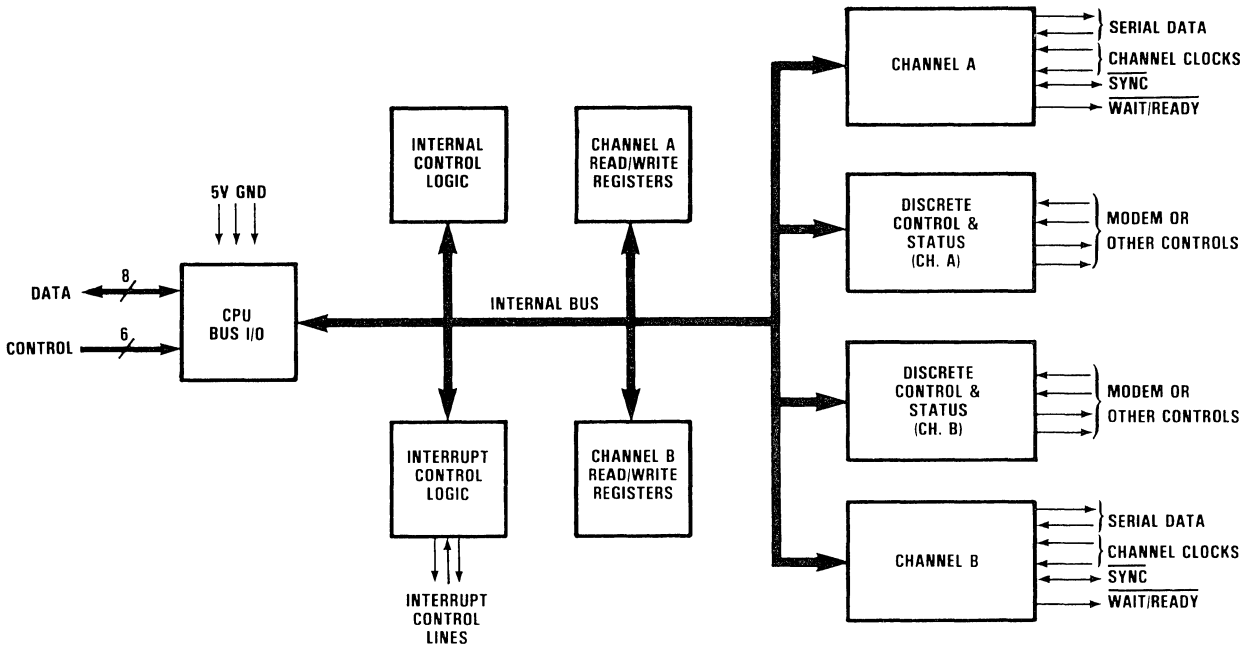


Figure 3-3. USART INTERNAL STRUCTURE

The USART's internal structure includes a CPU interface, internal control and interrupt logic, and two full duplex channels. Each channel contains read and write registers, and discrete control and status logic that provides the interface to modems or other external devices (see Figure 3-3).

The read and write register group includes five 8-bit control registers, two sync character registers and two status registers. The ZSIO interrupt vector capability is not used. All interrupt vectors are provided by the 8259A PIC. The registers for both channels are designated in the text as follows:

WR0-WR7 -- write registers 0 through 7

RR0-RR2 -- read registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Paragraps 3.4.1 and 3.4.2 on the following page list the functions assigned to each read or write register.

3.4.1 READ REGISTER FUNCTIONS

RR0 Transmit/receive buffer status, interrupt status and external status

RR1 Special receive condition status

RR2 Modified interrupt vector (Channel "B" only)

3.4.2 WRITE REGISTER FUNCTIONS

WR0 Register pointers, CRC initialize, initialization commands for the various modes, etc.

WR1 Transmit/receive interrupt and data transfer mode definition

WR2 Interrupt vector (Channel "B" only)

WR3 Receive parameters and control

WR4 Transmit/receive miscellaneous (parameters and modes)

WR5 Transmit parameters and controls

WR6 Sync character or SDLC address field

WR7 Sync character or SDLC flag

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs Clear to Send (CTS) and Data Carrier Detect (DCD) are monitored by the discrete control logic under program control. Strapping options permit on-board emulation of the modem control signals. The automatic interrupt vectoring capability of the ZSIO is not used. An attempt to use the ZSIO generated interrupt vectors will cause an indeterminate result.

Both channels contain command registers that must be programmed prior to operation. The controlFORTH monitor initializes Channel "A and B" of the ZSIO.

3.4.3 PROGRAMMING THE WRITE REGISTERS

The Z80-SIO contains eight registers (WR0-WR7) in each channel that are programmed separately by the system program to configure the functional personality of the channels. With the exception of WR0, programming the write register requires two bytes. The first byte contains three bits (D0-D2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z80-SIO. (See Figure 3-5.)

WR0 is a special case in that all the basic commands (CMD0-CMD2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D0-D2 to point to WR0.

3.4.4 PROGRAMMING THE READ REGISTERS

The Z80-SIO contains three registers, RR0-RR2 (Figure 3-1) that can be read to obtain the status information for each channel (except for RR2 -- Channel "B" only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the user program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then by executing an input instruction, the contents of the addressed read register can be read.

The status bits of RR0 and RR1 are grouped to simplify status monitoring. This enables the user to read all the appropriate error bits from one register (RR1).

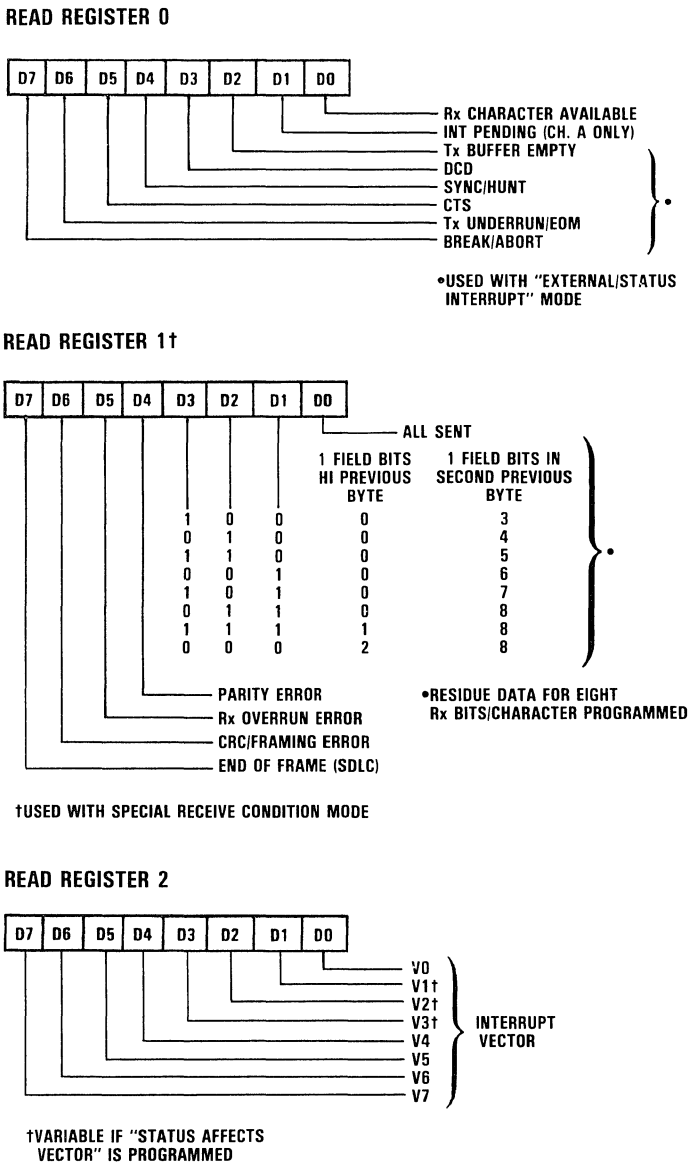
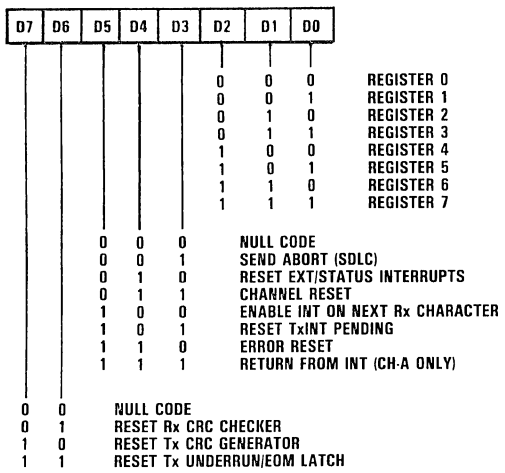
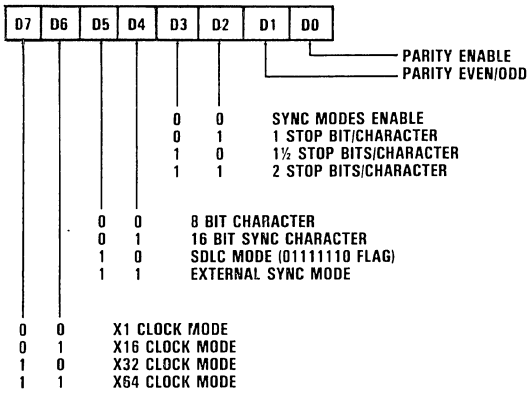


Figure 3-4. READ REGISTER BIT FUNCTIONS

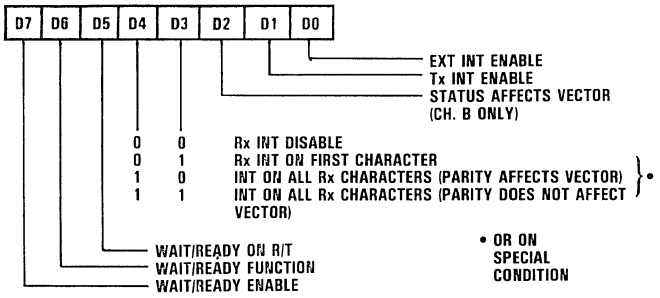
WRITE REGISTER 0



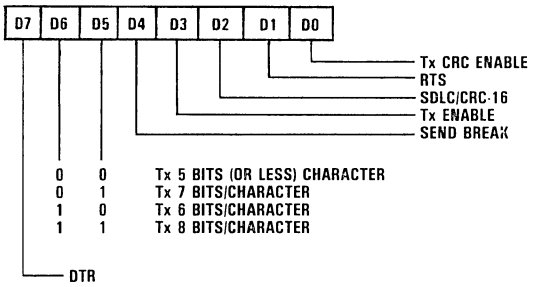
WRITE REGISTER 4



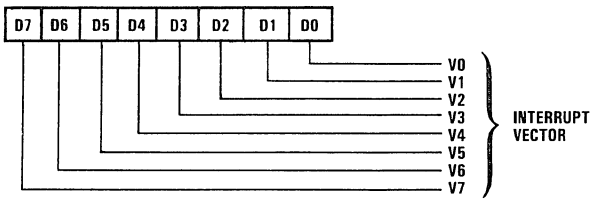
WRITE REGISTER 1



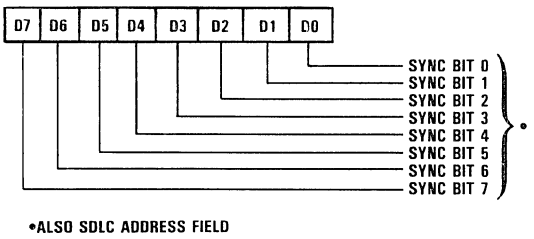
WRITE REGISTER 5



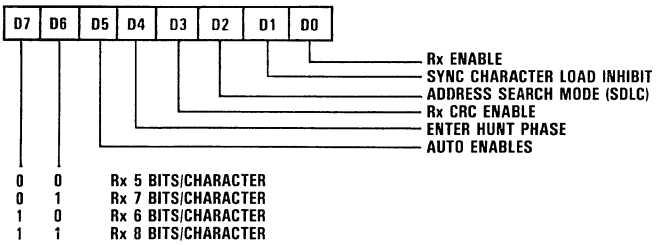
WRITE REGISTER 2 (CHANNEL B ONLY)



WRITE REGISTER 6



WRITE REGISTER 3



WRITE REGISTER 7

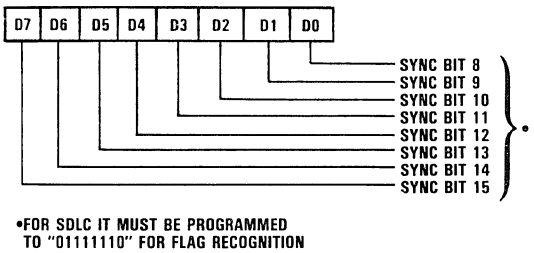


Figure 3-5. WRITE REGISTER BIT FUNCTIONS

3.5 INTERRUPT CONTROL

The 8259A programmable interrupt controller is in local I/O space at Hex address 0008. It can be programmed using the P@ and P! commands in the same manner as the serial communications ports.

When an interrupt request is generated and presented to one of the 8259A interrupt request lines, the interrupt controller will evaluate the interrupt request and, if appropriate, generate an interrupt request to the 8086. If interrupts are enabled, the processor will complete execution of the current instruction and enter the interrupt acknowledge machine cycle. The processor status line S2 being low indicates either an I/O operation or an interrupt machine cycle. The I/O bus will be enabled. The local 8288 bus controller generates an interrupt acknowledge signal (INTA) which is used to precondition the 8259A interrupt controller. No other activity takes place during this processor cycle. The second processor cycle duplicates the first up to issuing the INTA. The second INTA causes the 8259A to issue a vector byte to the 8086 via the I/O data bus transceiver. The vector byte is used to generate an address where the 8086 loads a new code segment and instruction pointer.

The base address of the interrupt controller is Hex 0008. Address line 1 is used to indicate the first word of either an initialized command word or an operational command word. For full programming information see the Intel Component Data Catalog, or the 8086 User's Guide.

SECTION 4.0

BREADBOARD INTERFACE

4.1 MEMORY ADDRESS BUS

The memory address bus is available at two locations. The full 20-bit address is available at pads FA0 through FA13. The lower 8-bits of the address lines are available on pads at board location A0 also. Refer to Figure 4-1 for pad location.

Memory addresses are always presented at these locations even when the address is not within the on-board address range.

4.2 I/O ADDRESS BUS

The I/O address bus is the lower 8-bits of the address bus. The I/O address bus is always active when any address is output from the 8086.

The addresses are available at board area G2. See Figure 4-2 for pad layout and numbering.

4.3 DATA LINES

Two separate sets of data transceivers are available. One is active during memory references or memory mapped I/O operations. The other is only active during input, output or interrupt acknowledge processor cycles.

4.3.1 MEMORY DATA BUS

Memory data lines are available to the user at pads FD0 through FDF. During an on-board memory write cycle, data will be valid on these pads during T3, Tw . . Tw. Tw is governed by strap settings at board location D1 (see Wait State Timing Section). During an on-board memory read, data should be presented during T3, Tw Tw, Tw. See Intel 8086 product specification for exact timings.

During off-board operations, the on-board transceivers will be tri-stated, allowing pads FD0 through FDF to float. The low order byte of the data bus is also available at board location A1. Refer to Figure 4-1 for pad location.

4.3.2 I/O DATA BUS

The I/O data bus is the low order byte of the data bus. It is only active during an input, output or interrupt acknowledge processor cycle. Timing is the same as the memory data bus. Wait states are set by straps at D2 (see Section on Wait State Timing).

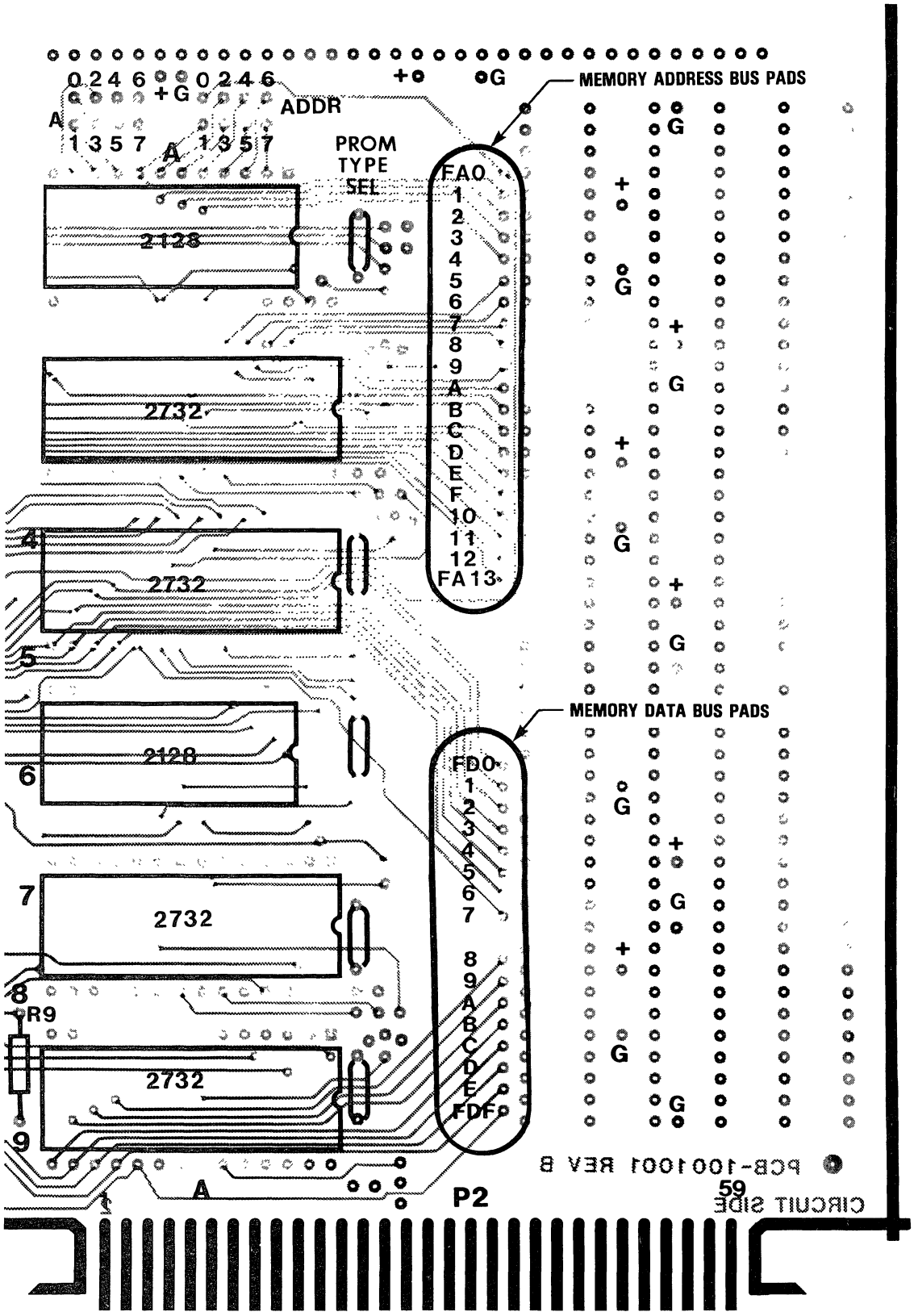
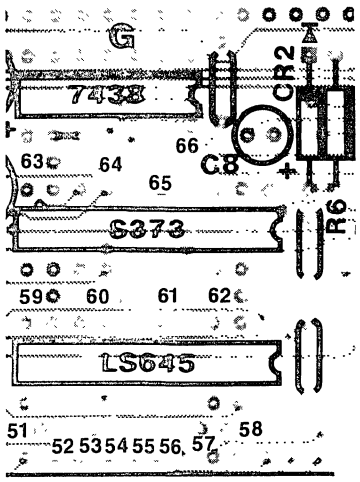


Figure 4-1. MEMORY ADDRESS AND DATA BUS'S



PAD#	I/O ADDRESS (Hex)
51	ØDBØ
52	ØDB1
53	ØDB2
54	ØDB3
55	ØDB4
56	ØDB5
57	ØDB6
58	ØDB7
59	ØABØ
60	ØAB2
61	ØAB4
62	ØAB6
63	ØAB1
64	ØAB3
65	ØAB5
66	ØAB7

Figure 4-2. I/O ADDRESS PADS

4.4 I/O SELECT LINES

All I/O addresses in the range of Hex 00 to Hex 53 are automatically considered on-board addresses even though not all of the I/O addresses are decoded.

The I/O select lines are decoded on 8 byte boundaries. Address line 0 is not used. All I/O port addresses must be even addresses. Address lines 2 and 1 are available for port addressing within the selected chip. Select lines 0 and 1 are used for the ZSIO and 8259A respectively. Refer to Figure 4-3 for I/O select line information.

4.5 CHIP SELECT DECODING

Chip select lines are provided for RAM, EPROM and I/O. Spare lines are available for user added components.

4.5.1 EPROM CHIP SELECTS

ERPOM chip select lines are available on pads at board location B2. Select lines are decoded on 8 Kbyte boundaries. If the user adds PROM's using the spare select lines they must be connected to the opposite pad in order to be within the on-board address space. Only the top 4 chip select lines may be used, allowing the user to put a maximum of 32 Kbytes of EPROM on-board.

One chip select line is provided for 2764 EPROMs. This is derived from OR'ing the top two chip select lines on pads 119 and 121. If 2764s are to be used, pads 119 and 121 must be strapped to 120 and 122 respectively. Refer to Figure 4-4 for pad location.

BASE ADDRESS	PAD	TO PAD
In Hex		
0E000 or FE000	121	122
0C000 or FC000	119	120
0A000 or FA000	123	124
08000 or F8000	125	126

I/O ADDRESSES	CHIP SELECT PAD NUMBER	SELECTED DEVICE
In Hex		
0000-0007	--	ZSIO
0008-000F	--	8259A
0010-0017	91	SPARE
0018-001F	90	SPARE
0020-0027	89	SPARE
0028-002F	88	SPARE
0030-0037	87	SPARE
0038-003F	92	SPARE

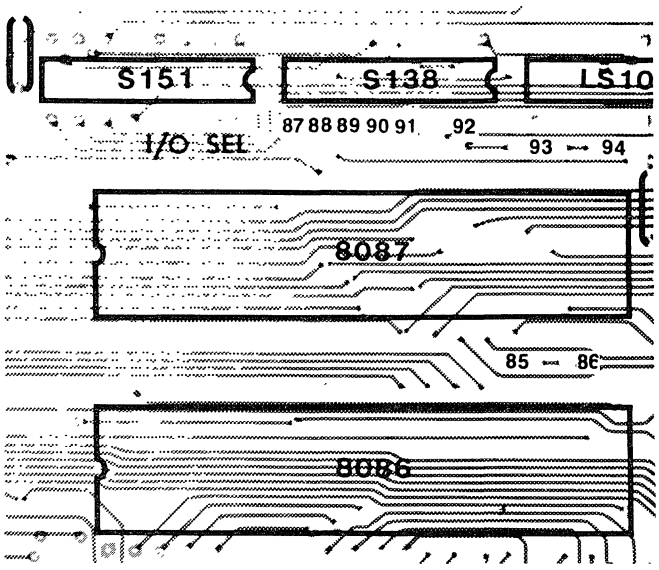


Figure 4-3. I/O SELECT PADS

4.5.2 RAM SELECT LINES

The RAM select lines are decoded on 4 Kbyte boundaries. One select line is used to select the 2 Kbit by 8 RAM chips at board locations 1A and 6A.

Seven select lines are avilable for use options. To be included in the on-board RAM space the user must connect the appropriate pad to the pad opposite it in addition to wiring the select signal to the RAM chip. Refer to Figure 4-4 for pad location.

BASE ADDRESS OF SELECT LINE (Hex)	CHIP SELECT PAD NUMBER	CONNECT TO PAD NUMBER
00000 or F0000	117	118
01000 or F1000	115	116
02000 or F2000	113	114
03000 or F3000	111	112
04000 or F4000	109	110
05000 or F5000	107	108
06000 or F6000	103	104
07000 or F7000	105	106

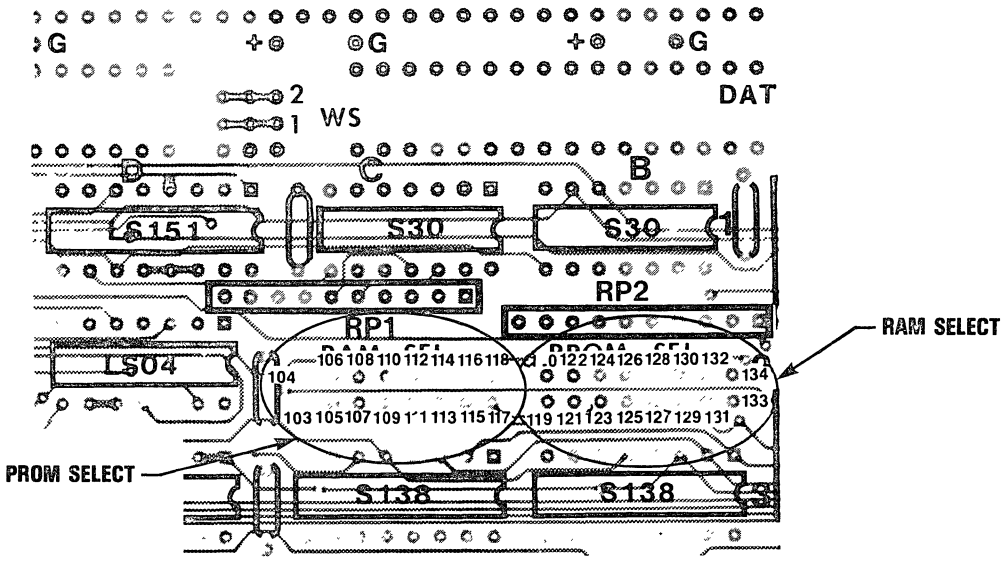


Figure 4-4. EPROM AND RAM CHIP SELECT LINES

SECTION 5.0

FIRMWARE

5.1 CONTROLFORTH

ControlFORTH is supplied in EPROM and is an implementation of FORTH derived from fig-FORTH and with certain extensions. The extensions are a general set of monitor commands to assist the user in adding, debugging and testing LSI devices. This Section explains the monitor enhancements of controlFORTH in detail. The glossary in Appendix A contains descriptions of the other FORTH words in controlFORTH. For further information on FORTH, Appendix B is a bibliography of FORTH works.

In the following text underlines indicate one or more spaces and ← means the RETURN key. With the exception of the substitute command all addresses, counts and value fields may be entered in decimal, octal or hexadecimal depending on the base selected prior to entering the command. Commands may be strung together, separated with spaces, on the same line with a carriage return (←) at the end.

Once a base is selected it will remain in effect until another base command is entered.

Base commands:

HEX_ ←	Sets hexadecimal base
DECIMAL_ ←	Sets decimal base
OCTAL_ ←	Sets octal base

The base selected will affect the number of characters displayed in any numeric field. All command formats shown use the hexadecimal base. See the glossary for more information.

5.2 MONITOR COMMANDS - Implied Segment Value

5.2.1 STUFF

FORMAT	AAAA_CCCC_VV_STUFF
AAAA	Source address
CCCC	Byte count
VV	Value to be inserted

The STUFF command executes using the implied segment value. Memory will be filled with the specified value VV starting at address AAAA for the number of bytes specified in CCCC. See 5.4.1 for information on setting or viewing the implied segment value. The 16-bit implied segment value and 16-bit addresses are used together to generate the 20-bit addresses of the 8086. See the biblography in Appendix B for more information the 8086.

EXAMPLE: HEX_C00_10_55_STUFF_ ←

This example fills 16 decimal (hexadecimal 10) memory locations starting at hexadecimal address C00 with hexadecimal 55.

5.2.2 SUBSTITUTE

FORMAT	AAAA_SUBSTITUTE
AAAA	Starting address

DISPLAY FORMAT

SSSS_AAAA_HH

This command executes on successive byte memory locations. If the contents of a byte location do not need to be changed, pressing carriage return will not alter the current value HH and fetch the next byte location. To change a value enter the required value VV and press carriage return. The entered value VV will replace the original value HH.

To terminate the SUBSTITUTE command press the Q key and carriage return. SUBSTITUTE operates in hexadecimal but it preserves and restores the base in use prior to using SUBSTITUTE.

5.2.3 MOVE

FORMAT	AAAA_BBBB_CCCC_MOVE
AAAA	Source address
BBBB	Destination address
CCCC	Number of bytes to moved

This command moves the specified number of bytes from locations starting at AAAA to locations starting at BBBB within the implied segment. The move is always to higher memory locations.

5.2.4 MATCH

FORMAT	AAAA_BBBB_CCCC_MATCH
AAAA	Source address 1
BBBB	Source address 2
CCCC	Byte count to compare

DISPLAY FORMAT FOR MISMATCH

IIII_AAAA_HH__IIII_BBBB_VV

Where IIII is the implied segment value, AAAA and BBBB are the addresses within the segment and HH and VV are the unequal values at those locations.

MATCH compares two given strings of bytes. The start addresses are AAAA and BBBB. This command will terminate either on completion of the count CCCC or after a given number of mismatches have been displayed. The number of errors tolerated before termination can be altered, see Section 5.4.3. This default is set to 10 mismatches.

5.2.5 P! and WP!

FORMAT	VV_DD_P! and VV_DD_WP!
VV	Value to be output to the port
DD	Destination port address

The commands P! and WP! write data to output ports. P! works with 8-bit ports and WP! is for 16-bit output ports. If the destination port address is within the local (on-board) I/O address space these commands will terminate normally even if there is no port at that address. If the destination port is in the off-board I/O address space, i.e., via the Multibus, two conditions can occur. If the destination port responds, the commands will terminate normally. If the destination port doesn't respond, the deadman timer will expire and terminate the command. In the second case there will be a delay of at least 1/10th of a second. Using a "TIMES . . . RUN" command it is simple to detect a non-responsive port.

5.2.6 P@ and WP@

FORMAT	DD_P@ and DD_WP@
DD	Input port address

The commands P@ and WP@ read data from a given input port. Again, P@ is for 8-bit ports and WP@ is for 16-bit ports. These commands will exhibit exactly the same symptoms as the OUT command if the addressed port does not respond. The data value displayed will be indeterminate if the port does not respond. To see the value read from the port type the FORTH print command . "dot" **or U.** See glossary for more information about dot.

EXAMPLE: to view the value from an I/O device at Hex port #2E type:

2E_P@_ . _ ←

5.2.7 GO

FORMAT	AAAA_GO
AAAA	Address of first instruction to execute

This command transfers control to the program whose first instruction is located at address AAAA and with the implied segment value.

5.2.8 RECEIVE

FORMAT	AAAA_CCCC_DD_RECEIVE
AAAA	Address where input data is to be loaded
CCCC	Count of bytes to be loaded
DD	Input port address

5.2.9 SEND

FORMAT	AAAA_CCCC_DD_SEND
AAAA	Address where output data begins
CCCC	Number of memory bytes to be transferred
DD	Output port address

TRANSMISSION FORMAT

HHHHHHHH etc.

Where each H is an ASCII character containing 4 of the 8-bits in a byte. To SEND n bytes requires transmission of 2 n characters.

The command pair RECEIVE and SEND receive and send binary data via a selected serial input/output port. The binary data is broken into 4-bit nibbles and converted to form the ASCII Hex characters 0 through 9F and A0 through F for transmission over a serial link. The format for both commands is simply one of a long string of ASCII characters. The command SEND assembles the ASCII from the data bytes and transmits it. The RECEIVE command accepts an ASCII string, strips the ASCII, reassembles the original data bytes, and places them in memory at the given address.

5.2.10 DUMP

FORMAT	AAAA_CCCC_DUMP
AAAA	Starting address
CCCC	Byte count

DISPLAY FORMAT

AAAA_n_n_n_n_n_n_n_n_n_n_n_n_n_n_n_n

5.3 MONITOR COMMANDS WITH EXPLICIT SEGMENT ADDRESSES

5.3.1 SMOVE

FORMAT	AAAA_SSSS_BBBB_DDDD_CCCC_SMOVE
AAAA	Source address within source segment
SSSS	Source segment address
BBBB	Destination address within destination segment
DDDD	Destination segment address
CCCC	Count of bytes to be moved

The command SMOVE functions in the same way as MOVE except that the segment values must be explicitly specified.

5.3.2 SMATCH

FORMAT	AAAA_SSSS_BBBB_DDDD_CCCC_SMATCH
AAAA	First source address within source segment
SSSS	First source segment register address
BBBB	Second source address within segment
DDDD	Second source segment register address
CCCC	Count of bytes to be compared

DISPLAY FORMAT

SSSS_AAAA_HH_ _ _ _DDDD_BBBB_VV

Command terminates either on completion of count or upon a given number of mismatches, see Section 5.4.3.

5.4 MONITOR CONTROL COMMANDS

5.4.1 SEGMENT

FORMAT	SSSS_SEGMENT
SSSS	Value to be loaded into the implied segment register. The contents of the implied segment may be views by typing: IMPLIED_?_ ← . (Note: If the number appears negative type: IMPLIED_@_U._ ←).

5.4.2 TIMES . . . RUN

FORMAT	NNNN_TIMES_COMMAND_RUN
NNNN	Loop counter value
COMMAND	One or more of the monitor commands using either implied or explicit segment register.

The TIMES . . . RUN command pair must be used as a pair and in the order given above. The commands that go between TIMES and RUN will be executed at least once even if a zero is given for the count. The maximum count is 32,767. The number of commands that go in between has no practical limit and may extend over several lines. Each command must be able to be found in controlFORTH's dictionary or an error will result.

5.4.3 TOLERATED

To set the number of mismatches tolerated before termination of MATCH or SMATCH type:

```
n_TOLERATED_ ←
```

To view the number tolerated type:

```
#TOLERATED_?_ ←
```

APPENDIX A

CONTROLFORTH GLOSSARY

This glossary contains the definition of all words in the controlFORTH vocabulary. The definitions are presented in ASCII sort order.

Stack Notation

The first line of each entry shows a symbolic description of the action of the procedure on the parameter stack. The symbols on the left indicate the order in which input parameters have been placed on the stack. Three dashes "---" indicate the execution point; any parameters left on the stack after execution are listed on the right. In this notation, the top of the stack is to the right.

Symbol Definition

addr,adrl, . . .	Memory address
b	8-bit (with high eight bits zero)
c	7-bit ASCII character (with high nine bits zero)
d,di, . . .	32-bit signed double integer, most significant portion with sign on top of stack
flag	Boolean flag (Ø = false, non-zero = true)
ff	Boolean false flag (value = Ø)
n,nl, . . .	16-bit signed integer number
u,ul, . . .	16-bit unsigned integer number
ud,udi, . . .	32-bit unsigned number
tf	Boolean true flag (value - non-zero)

Pronunciation

The natural language pronunciation of controlFORTH names is given in double quotes (").

Integer Format

Unless otherwise noted, all references to numbers are for 16-bit signed integers. For 32-bit signed double numbers, the most significant part (with the sign) is on top.

All arithmetic is implicitly 16-bit signed integer math, with error and underflow indication unspecified.

Capitalization

Word names as used within the glossary are conventionally written in upper case characters. Lower case is used when reference is made to the run-time machine codes, not directly accessible. (i.e. VARIABLE is the user word to create a varible.) Each use of that variable makes use of a code sequence 'variable' which executes the function of the particular variable.

Attributes (ATTR)

Capital letters show definition characteristics:

C	May only be used within a colon definition. A digit indicates number of memory addresses used, if other than one.
E	Intended for execution only.
I	Indicates that the word is IMMEDIATE and will execute during compilation, unless special action is taken.
P	Has precedence bit set. Will execute even when compiling.
U	A user variable.

Group Key Words (GROUP)

The following key words identify the functional groups that each word is most related to.

STACK	Stack Manipulation
NUMERIC	Numeric Representation
ARITHMETIC	Arithmetic and Logical
COMPARISON	Comparison Operators
CONTROL	Control Structures
MEMORY	Memory
I/O	Input/Output
FORMAT	Output Formatting
COMPILER	Compiler - Text Interpreter
DICTIONARY	Dictionary Control
DEFINING	Defining Words
VOCABULARY	Vocabularies
MASS	Mass Storage
MISC	Miscellaneous
SECURITY	Security/Error Detection
PRIMITIVE	Primitives
ASSEMBLER	Assembler Dictionary
PARAMETER	Parameter Used in ControlFORTH

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

! — **n addr ---** — **"store"** — MEMORY

Stores 16-bit number n into addr.

!CSP — **---** — **"store CSP"** — SECURITY

Stores the stack position in CSP . Used as part of the compiler security. See CSP .

— **udl --- ud2** — **"sharp"** — FORMAT

Generates the next ASCII character placed in an output string from udl. Result ud2 is the quotient after division by BASE, and is maintained for further processing. Use between <# and #> . See #S .

#> — **d --- addr n** — **"sharp-greater"** — FORMAT

Terminates numeric output conversion by dropping d, leaving the text address and character count n suitable for TYPE .

#S — **ud --- 0 0** — **"sharp-s"** — FORMAT

Converts all digits of a ud adding each to the pictured numeric output text, until the remainder is zero. A single zero is added to the output string if the number was initially zero. Use only between <# and #> .

' — **--- addr** — **"tick"** — DICTIONARY I

Use in the form:

' <name>

If executing, leaves the parameter field address of the next word accepted from the input stream. If compiling, compiles this address as a literal; later execution will place this value on the stack.

If the word is not found after a search of CONTEXT and FORTH vocabularies an error message is displayed.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
(	**"paren"**	MISC

Used in the form:

(cccc)

Accepts and ignores comment characters from the input stream, until the next right parenthesis. As a word, the left parenthesis must be followed by one blank. It may be freely used while executing or compiling. An error condition exists if the input stream is exhausted before the right parenthesis.

(.") PRIMITIVE

The run-time procedure, compiled by ." , which transmits the following in-line text to the selected See ." .

(;CODE) PRIMITIVE

The run-time procedure, compiled by ;CODE , that rewrites the code field of the most recently defined word to point to the following machine code sequence. See ;CODE .

(+LOOP) PRIMITIVE

The run-time procedure compiled by +LOOP , which increments the loop index by n and tests for loop completion. See +LOOP .

(ABORT) PRIMITIVE

Executes after an error when WARNING is -1. This word normally executes ABORT , but may be altered (with care) to a user's alternative procedure. See ABORT .

(DO) **limit+1 start ---** PRIMITIVE

The run-time procedure, compiled by DO , which moves the loop control parameters to the return stack. See DO .

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
(FIND)	**addr1 addr2 --- pfa byte tf (found)** **addr1 addr2 --- ff (not found)**	PRIMITIVE
	Searches the dictionary starting at the name field address addr2, matching to the text at addr1. Returns parameter field address, length of name field byte and Boolean true for a good match. If no match is found, only a Boolean false is left. See -FIND .	
(LOOP)		PRIMITIVE C
	The run-time procedure, compiled by LOOP, which increments the loop index and tests for loop completion. See LOOP .	
(NUMBER)	**d1 addr1 --- d2 addr2**	PRIMITIVE
	Converts the ASCII text beginning at addr1+1 with regard to BASE . The new value is accumulated into d1, being left as d2. addr2 is the address of the first unconvertable digit. See NUMBER .	
*	**n1 n2 --- n3** **"times"**	ARITHMETIC
	Multiples n1 by n2 and leaves the product n3.	
*/	**n1 n2 n3 --- n4** **"times-divide"**	ARITHMETIC
	Multiplies n1 by n2, divides the result by n3 and leaves the quotient n4. n4 is rounded toward zero. The product of n1 times n2 is maintained as in intermediate 32-bit value for a greater precision than the otherwise equivalent sequence:	
	n1 n2 * n3 /	
*/MOD	**n1 n2 n3 --- n4 n5** **"times-divide-mod"**	ARITHMETIC
	Multiplies n1 by n2, divides the result by n3 and leaves the remainder n4 and quotient n5. A 32-bit intermediate product is used as for */ . The remainder has the same sign as n1.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

+ **n1 n2 --- n3**
"plus" ARITHMETIC

Adds n1 to n2 and leaves the arithmetic sum n3.

+! **n addr ---**
"plus store" MEMORY

Adds n to the 16-bit value at the address, by the convention given for +.

+- **n1 n2 --- n3**
"plus-minus" ARITHMETIC

Applies the sign of n2 to n1, which is left as n3.

+LOOP CONTROL IC

n1 --- (run-time)
addr n2 --- (compile-time)
"plus loop"

Used in a colon-definition in the form:

DO . . . n1 +LOOP

At run-time, +LOOP selectively controls branching back to the corresponding DO based on n1, the loop index and the loop limit. The signed increment n1 is added to the index and the total compared to the limit. The branch back to DO occurs until the new index is equal to or greater than the limit (n1 > Ø), or until the new index is equal to or less than the limit (n1 < Ø). Upon exiting the loop, the parameters are discarded and execution continues. Index and limit are signed integers in the range -32,768 to 32,767.

At compile-time, +LOOP compiles the run-time word (+LOOP) and computes the branch offset from HERE to the address left on the stack by DO . n2 is used for compile time error checking.

n ---
"comma" DICTIONARY

Stores n into the next available dictionary memory cell, advancing the dictionary pointer.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

- — ARITHMETIC

n1 n2 --- n3
"minus"

Substracts n2 from n1 and leaves the difference n3.

-FIND — DICTIONARY

--- pfa byte tf (found)
--- ff (not found)
"dash-find"

Accepts the next text word (delimited by blanks) in the input stream to HERE , and searches the CONTEXT and then CURRENT vocabularies for a matching entry. If found, the dictionary entry's parameter field address, its length byte, and a Boolean true is left. Otherwise, only a Boolean false is left.

-TRAILING — FORMAT

addr n1 --- addr n2
"dash-trailing"

Adjusts the character count n1 of a text string beginning address to suppress the output of trailing blanks. The characters at addr+n1 to addr+n2 are blanks. An error condition exists if n1 is negative.

. — INPUT/OUTPUT

n ---
"dot"

Displays the number on the top of a stack. The number is converted from a signed 16-bit two's complement value according to the numberic BASE . The sign is displayed only if the value is negative. A trailing blank is displayed after the number. Also see D. .

." — INPUT/OUTPUT I

"dot-quote"

Used in the form:

." cccc"

Accepts the following text from the input stream, terminated by " (double-quote). If executing, transmits this text to the selected output device. If compiling, compiles so that later execution will transmit the text to the selected output device. At least 127 characters are allowed in the text. If the input stream is exhausted before the terminating double-quote, an error condition exists.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
.R	n1 n2 --- "dot-R"	FORMAT
	Displays number n1 right justified n2 places. No trailing blank is printed.	
.S	"dot-S"	STACK
	Displays the contents of the stack without altering the stack. This word is very useful in determining the stack contents during debugging programs and learning FORTH.	
/	n1 n2 --- n3 "divide"	ARITHMETIC
	Divides n1 by n2 and leave the quotient n3. n3 is rounded toward zero. The remainder is lost.	
/MOD	n1 n2 --- n3 n4 "divide-mod"	ARITHMETIC
	Divides n1 by n2 and leaves the quotient n3 and remainder n4. n3 has the same sign as n1.	
0	--- 0 "zero"	NUMERIC
	The number zero is placed on top of the stack.	
0<	n --- flag "zero-less"	COMPARISON
	Leaves a true flag (1) if the number is less than zero (negative), otherwise leaves a false flag (0) . The number is lost.	
0=	n --- flag "zero-equals"	COMPARISON
	Leaves a true flag (1) if the number is equal to zero, otherwise leaves a false flag (0). The number is lost.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
0BRANCH	flag --- "zero-branch"	PRIMITIVE C
	The run-time procedure to conditionally branch. If the flag is false (zero), the following in-line parameter is added to the interpretive pointer to branch ahead or back. Compiled by IF , UNTIL , and WHILE .	
1	--- 1 "one"	NUMERIC
	The number one is placed on top of the stack.	
1+	n --- n+1 "one-plus"	ARITHMETIC
	Increments n by one according to the operation of +.	
1-	n --- n-1 "one-minus"	ARITHMETIC
	Decrements n by one according to the operation of -.	
2	--- 2 "two"	NUMERIC
	The number two is placed on top of the stack.	
2+	n --- n+2 "two-plus"	ARITHMETIC
	Increments n by two according to the operation of +.	
2-	n --- n-2 "two-minus"	ARITHMETIC
	Decrements n by two, according to the operation of -.	
2DROP	d --- or n1 n2 --- "two-drop"	STACK
	Drops the top double number on the stack.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

2DUP — d --- d d — STACK
or n1 n2 --- n1 n2 n1 n2
"two-dup"

Duplicates the top double number on the stack.

3 — --- 3 — NUMERIC
"three"

The number three is placed on top of the stack.

: — DEFINING E
"colon"

A defining word used in the form:

: <name> . . . ;

Selects the CONTEXT vocabulary to be identical to CURRENT . Creates a dictionary entry for name in CURRENT , and sets the compile mode. Words thus defined are called 'colon-definitions'. The compilation addresses of subsequent words from the input stream which are not immediate words are stored into the dictionary to be executed when <name> is later executed. IMMEDIATE words are executed as encountered.

If a word is not found after a search of the CONTEXT and FORTH vocabularies conversion and compilation of a literal number is attempted, with regard to the current BASE ; that failing, an error condition exists.

; — DEFINING I
"semi-colon"

Terminates a colon-definition and stops further compilation. If compiling from an external source and the input stream is exhausted before encountering ; an error condition exists.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

;CODE — DEFINING I

"semi-colon-code"

Used in the form:

: <name> . . . . ;CODE <assembly code>
END-CODE

Stops compilation and terminates a new defining word name by compiling (;CODE) . The assembly code is put into place by putting bytes on the stack and using C, and , to emplace the opcodes in line. Example:

0_1 C, 0_2 C, n, ,

When <name> is later executed in the form:

<name> <namex>

To define the new <namex>, the code field address of <namex> will contain the address of the code sequence following the ;CODE in <name> . Execution of any <namex> will cause this machine code sequence to be executed.

;S — COMPILER

"semi-colon-S"

Stops interpretation of an input stream. ;S is also the run-time word compiled at the end of a colon-definition which returns execution to the calling procedure.

< — **n1 n2 --- flag** — COMPARISON

"less-than"

Leaves a true flag (1) if n1 is less than n2; otherwise leaves a false flag (Ø).

<# — **d --- d** — FORMAT

"less-than-sharp"

Initializes the pictured numeric output format using the words:

<# # #S HOLD SIGN #>

specifies the conversion of a double-precision number into an ASCII character string stored in right-to-left order, producing text at PAD .

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

<BUILDS — DEFINING

Used within a colon-definition:

: <name> <BUILDS . . . DOES> . . . ;

Each time <name> is executed, <BUILDS defines a new word with a high-level execution procedure. Executing <name> in the form:

<name> <namex>

Uses <BUILDS to create a dictionary entry for <namex> with a call to the DOES> part for <namex>. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES> in <name>. <BUILDS and DOES> allows run-time procedures to be written in high-level rather than in assembler code (as required by ;CODE).

> — **n1 n2 --- flag** **"greater-than"** — COMPARISON

Leaves a true flag (1) if n1 is greater than n2; otherwise a false flag (Ø).

>R — **n ---** **"to-R"** — STACK

Removes a number from the computation stack and places it as the most accessible number on the return stack. Use should be balanced with R> in the same definition.

? — **addr --** **"question-mark"** — STACK

Displays the value contained at the address on the top of the stack in free format according to the current BASE. Uses the format of . .

?COMP — **"question comp"** — SECURITY

Issues error message if not compiling.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
?CSP	**"question c s p"** Issues error message if stack position differs from value saved in CSP .	SECURITY
?DUP	**n1 --- n1 (if zero)** **n1 --- n1 n1 (non-zero)** **"question-dup"** Reproduces n1 only if it is non-zero. This is usually used to copy a value just before IF, to eliminate the need for an ELSE clause to drop it.	STACK
?ERROR	f n --- **"question error"** Issues error message n if the Boolean flag is true.	SECURITY
?EXEC	**"question exec"** Issues an error message if not executing.	SECURITY
?PAIRS	**n1 n2 ---** **"question pairs"** Issues error message #19 (CONDITIONALS NOT PAIRED) if n1 does not equal n2. The message indicates that compiled conditionals do not match.	SECURITY
?STACK	**"question stack"** Issues error message #7 (FULL STACK) if the stack is out of bounds.	SECURITY
?TERMINAL	**--- flag** **"question terminal"** Tests the terminal keyboard for actuation of any key. Generates a Boolean value. A true flag (1) indicates actuation, whereas a false flag (Ø) indicates non-actuation.	INPUT/OUTPUT

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
@	**addr --- n** **"fetch"**	MEMORY
	Leaves the 16-bit contents of the address on top of the stack.	
ABORT	**"abort"**	SECURITY
	Clears the stacks and enters the execution state. Returns control to the active I/O port.	
ABS	**n --- u** **"absolute"**	ARITHMETIC
	Leaves the absolute value of n as u.	
AGAIN	**addr n --- (compile-time)** **"again"**	CONTROL
	Used in a colon-definition in the form: **BEGIN . . . AGAIN** At run-time, AGAIN forces execution to return to the corresponding BEGIN . There is no effect on the stack. Execution cannot leave this loop (unless R> DROP is executed one level below). At compile-time, AGAIN compiles BRANCH with an offset from HERE to addr. n is used for compile-time error checking.	
ALLOT	**n ---** **"allot"**	COMPILER
	Adds the signed number to the dictionary pointer DP . May be used to reserve dictionary space or re-origin memory. n is the number of bytes.	
AND	**n1 n2 --- n3** **"and"**	ARITHMETIC
	Leaves the bitwise logical AND of n1 and n2 as n3.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

BASE — **--- addr** — **"base"** — NUMERIC U

Leaves the address of the variable containing the current number base used for input and output conversion. The range of BASE is 2 through 7Ø.

BEGIN — **--- addr n (compile-time)** — **"begin"** — CONTROL

Occurs in a colon-definition in form:

```
BEGIN  . . .  Flag UNTIL
BEGIN  . . .  AGAIN
BEGIN  . . .  flag WHILE  . . .  REPEAT
```

At run-time, BEGIN marks the start of a word sequence for repetitive execution.

A BEGIN-UNTIL loop will be repeated until flag is true. A BEGIN-WHILE-REPEAT loop will be repeated until flag is false. The words after UNTIL or REPEAT will be executed when either loope is finished. flag is always dropped after being tested. The BEGIN-AGAIN loop executes indefinitely.

At compile-time, BEGIN leaves its return address and n for compiler error checking.

BL — **--- char** — **"blank"** — INPUT/OUTPUT

A constant that leaves the ASCII character value for "blank", i.e. Hex 20.

BLANKS — **addr n ---** — **"blanks"** — MEMORY

Fills an area of memory beginning at addr with the ASCII value for "blank", the number of bytes specified by count n will be blanked.

BLK — **--- addr** — **"b-l-k"** — MASS U

Leaves the address of a user variable containing the number of the mass storage block being interpreted as the input stream. If the content is zero, the input stream is taken from the terminal. This variable is used internally and is included so that later mass storage words can be added.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
BRANCH	**"branch"**	CONTROL
	The run-time procedure to unconditionally branch. An in-line offset is added to the interpretive pointer IP to branch ahead or back. BRANCH is compiled by ELSE , AGAIN , and REPEAT .	
C!	**n addr ---** **"c-store"**	MEMORY
	Stores the least significant 8-bits of n into the byte at the address.	
C,	**n ---** **"c-comma"**	DICTIONARY
	Stores the least significant 8-bits of n into the next available dictionary byte, advancing the dictionary pointer.	
C/L	**--- n** **"characters/line"**	INPUT/OUTPUT
	Leaves the number of characters (default value = 8Ø) per input line.	
C@	**addr --- byte** **"c-fetch"**	MEMORY
	Leaves the 8-bit contents of the byte at the address on the top of the stack in the low order byte. The high order byte is zero.	
CFA	**pfa --- cfa** **"c-f-a"**	MISC
	Converts the parameter field address (pfa) of a definition to its code field address (cfa).	
CLIT	**--- b** **"c-lit"**	STACK
	Compiled within system object code to indicate that the next byte is a single character literal (i.e. in range Ø-255). Used only in system code (not by application program, i.e. user). Application programs use LITERAL , which uses CLIT or LIT as appropriate.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

CMOVE — **addr1 addr2 n ---** — **MEMORY**

"c-move"

Moves n bytes from memory area beginning at address addr1 to memory area starting at addr2. The contents of addr1 is moved first proceeding toward high memory. If n is zero or negative, nothing is moved.

COLD — **MONITOR**

"cold"

The cold start procedure to adjust the dictionary pointer to the minimum standard and restart via ABORT . May be called from the terminal to remove application programs and restart. Performs the same functions as entering control-FORTH by a reset or power on sequence.

COMPILE — **COMPILER**

"compile"

When the word containing COMPILE executes, the compilation address of the next non-immediate word following COMPILE is copied (compiled) into the dictionary. This allows specific compilation situations to be handled in addition to simply compiling an execution address (which the interpreter already does).

CONSTANT — **n --- <name> (compile-time)** — **DEFINING**

<name> --- n (run-time)

"constant"

A defining word used in the form:

n CONSTANT <name>

To create a dictionary entry for <name>, leaving n in its parameter field. When <name> is later executed, it will push the value of n to the stack.

CONTEXT — **--- addr** — **DICTIONARY**

"context"

Leaves the address of a user variable pointing to the vocabulary in which dictionary searches are made, during interpretation of the input stream.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

COUNT — **addr --- addr+1 n** — **"count"** — FORMAT

Leaves the addr+1 and the character count n of text beginning at addr. The first byte at addr must contain the character count n. The actual text starts with the second byte. The range of n is Ø-255. Typically COUNT is followed by TYPE .

CR — **"carriage-return"** — INPUT/OUTPUT

Transmits a carriage return (CR) and line feed (LF) to the active output device.

CREATE — **"create"** — DICTIONARY

A defining word used in the form:

CREATE <**name**>

Creates a dictionary entry for <name> without allocating any parameter field memory. When <name> is subsequently executed, the address of the first byte of <name>'s parameter field is left on the stack. The code field contains the address of the word's parameter field. The new word is created in the CURRENT vocabulary.

CSP — **--- addr** — **"c-s-p"** — SECURITY U

Leaves the address of a user variable temporarily storing the check stack pointer (CSP) position, for compilation error checking.

CURRENT — **--- addr** — **"current"** — DICTIONARY

Leaves the address of a user variable pointing to the vocabulary into which new word definitions are to be entered.

D+ — **d1 d2 --- d3** — **"d-plus"** — ARITHMETIC

Adds double precision numbers d1 and d2 and leaves the double precision number sum d3.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

D+- — ARITHMETIC

d1 n --- d2
"d-plus"

Applies the sign of n to the double precision number d1 and leaves it as double precision number d2.

D. — FORMAT

d ---
"d-dot"

Displays a signed double-precision number from a 32-bit two's complement value. The high-order 16-bits are most accessable on the stack. Conversion is performed according to the current BASE . A blank follows.

D.R — FORMAT

d n ---
"d-dot-r"

Displays a signed double-precision number d right aligned in a field n characters wide. No blank follows.

DABS — ARITHMETIC

d --- ud
"d-abs"

Leaves the absolute value ud of a double number.

DECIMAL — NUMERIC

"decimal"

Sets the numeric conversion BASE to decimal (base 1Ø) for input-output.

DEFINITIONS — VOCABULARY

"definitions"

Used in the form:

cccc DEFINITIONS

Sets CURRENT to the CONTEXT vocabulary so that subsequent definitions will be created in the vocabulary previously selected at CONTEXT . In the example, executing vocabulary name cccc made it the CONTEXT vocabulary and executing DEFINITIONS made both specify vocabulary cccc .

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

DIGIT — NUMERIC

char n1 --- n2 tf (valid conversion)
char n1 --- ff (invalid conversion)
"digit"

Converts the ASCII character (using BASE n1) to its binary equivalent n2, accompanied by a true flag (1). If the conversion is invalid, leaves only a false flag (Ø).

DLITERAL — COMPILER

d --- d (executing)
d --- (compiling)
"d-literal"

If compiling, compiles a stack double number into a literal. Later execution of the definition containing the literal will push it to the stack.

If executing, the number will remain on the stack.

DNEGATE — ARITHMETIC

d1 --- d1
"d-negate"

Leaves the two's complement of a double precision number.

DO — CONTROL

n1 n2 --- (run-time)
addr n --- (compile-time)

Occurs in a colon-definition in form:

DO . . . LOOP
DO . . . +LOOP

At run-time, DO begins a sequence with repetitive execution controlled by a loop limit n1 and an index with initial value n2. DO removes these from the stack. Upon reaching LOOP the index is incremented by one. At the +LOOP the index is modified by a positive or negative value. Until the new index equals or exceeds the limit, execution loops back to just after DO ; otherwise the loop parameters are discarded and execution continues ahead. Both n1 and n2 are determine at run-time and may be the result of other operations.

Loops may be nested. Within a loop I will copy the current value of the index to the stack. See I , LOOP , +LOOP , LEAVE .

At compile-time within the colon-definition, DO compiles (DO) and leaves the following addr and n for later error checking.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

DOES> DEFINING

"does"

Defines the run-time action within a high-level defining word.

Used in the form:

: <name> . . . (BUILDS . . .
DOES> . . . ;
and the <name> <namex>.

Marks the termination of the defining part of the defining word <name> and begins the definition of the run-time action for words that will later be defined by <name>.

DOES> alters the code field and first parameter of the new word to execute the sequence of compiled word addresses following DOES>. Used in combination with <BUILDS . The execution of the DOES> part begins with the address of the first parameter of the new word <namex> on the stack. Upon execution of <name> the sequence of words between DOES> and ; will be executed, with the address of <namex>'s parameter field on the stack. This allows interpretation using this area or its contents.

Typical uses include a FORTH assembler, multi-dimensional arrays, and compiler generation.

DP **--- addr** COMPILER U
"d-p"

Leaves the address of user variable, the dictionary pointer, which points to address the next free memory address above the dictionary. The value may be read by HERE and altered by ALLOT .

DPL **--- addr** FORMAT U
"d-p-l"

Leaves the address of user variable containing the number of digits to the right of the decimal on double integer input. It may also be used to hold the output column location of a decimal point, in user generated formatting. The default value on single number input is -1.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

DROP — **n ---** **"drop"** — STACK

Drops the number on top of the stack from the stack.

DUMP — **addr n ---** **"dump"** — INPUT/OUTPUT

Displays the contents of n memory locations beginning at addr. Both addresses and contents are shown in the current numeric base.

DUP — **n --- n n** **"dup"** — STACK

Duplicates the value on the stack.

ELSE — **addr1 n1 --- addr2 n2 (compiling)** **"else"** — CONTROL I

Occurs within a colon-definition in the form:

IF . . . ELSE . . . THEN

At run-time, ELSE executes after the true part following IF . ELSE forces execution to skip over the following false part and resumes execution after the THEN . It has no stack effect.

At compile-time, ELSE emplaces BRANCH reserving a branch offset, leaves the address addr2 and n2 for error testing. ELSE also resolves the pending forward branch from IF by calculating the offset from addr1 to HERE and storing at addr1. See IF and THEN .

EMIT — **char ---** **"emit"** — INPUT/OUTPUT

Transmits an ASCII character to the active output device. See KEY .

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
ENCLOSE	**addr char --- addr n1 n2 n3** **"enclose"**	PRIMITIVE

The text scanning primitive used by WORD. From the text address addr and an ASCII delimiting character, is determined the byte offset to the first non-delimiter character n1, the offset to the first delimiter after the text n2, and the offset to the first character not included n3. This procedure will not process past an ASCII 'null', treating it as an unconditional delimiter.

ERASE	**addr n ---** **"erase"**	MEMORY

Clears a region of memory to zero from addr over n addresses.

ERROR	**line --- in blk** **"error"**	SECURITY

Executes error notification and restart of system. WARNING is first examined. If WARNING = 1, the text of line n, relative to screen 4 of drive Ø is printed. This line number may be positive or negative, and beyond just screen 4. If WARNING = Ø, n is just printed as a message number (non-disk installation). If WARNING = -1, the definition (ABORT) is executed, which executes the system ABORT . The user may cautiously modify this execution by altering (ABORT) . ControlFORTH saves the contents of IN and BLK to assist in determining the location of the error. Final action is execution of QUIT .

EXECUTE	**addr ---** **"execute"**	COMPILER

Executes the definition whose code field address is on the stack. The code field address is also called the compilation address.

EXPECT	**addr count ---** **"expect"**	INPUT/OUTPUT

Transfers characters from the terminal beginning at addr, upwards until a "return" or the count of n characters has been received. Takes no action for n = zero or less. One or more nulls are added at the end of the text.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

FENCE --- **addr** SECURITY U
"fence"

Leaves the address of a user variable containing an address below which FORGETting is trapped. To forget below this point the user must alter the contents of FENCE .

FORGET DICTIONARY
"forget"

Executes in the form:

FORGET <**name**>

Delete from the dictionary <name> (which is in the CURRENT vocabulary) and all words added to the dictionary after <name>, regardless of their vocabulary. An error message will occur if the CURRENT and CONTEXT vocabularies are not currently the same. Failure to fine <name> in CURRENT or FORTH is an error condition.

FORTH VOCABULARY
"forth"

The name of the primary vocabulary. Execution makes FORTH the CONTEXT vocabulary.

New definitions become a part of FORTH until a differing CURRENT vocabulary is established.

User vocabularies conclude by "chaining" to FORTH , so it should be considered that FORTH is 'contained' within each user's vocabulary.

HERE --- **addr** DICTIONARY
"here"

Leaves the address of the next available dictionary location.

HEX NUMERIC
"hex"

Sets the numeric conversion BASE to sixteen (hexadecimal).

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

HLD — **--- addr** — **"hold"** — FORMAT

Leaves the address of user variable which holds the address of the latest character of text during numeric output conversion.

HOLD — **char ---** — **"hold"** — FORMAT

Used between <# and #> to insert an ASCII character into a pictured numeric output string.

I — **--- n** — **"i"** — CONTROL

Used within a DO-LOOP to copy the loop index from the return stack to the stack.

ID. — **nfa ---** — **"i-d-dot"** — INPUT/OUTPUT

Print a definition's name from its name field address. See NFA .

IF — **flag --- (run-time)** / **--- addr n (compile)** — **"if"** — CONTROL

Used in a colon-definition in form:

```
IF ... THEN
IF ... ELSE ... THEN
```

At run-time, IF selects execution based on a Boolean flag. If flag is true, the words following IF are executed and the words following ELSE are skipped. The ELSE part is optional.

If flag is false, the words between IF and ELSE , or between IF and THEN (when no ELSE is used), are skipped. IF-ELSE-THEN conditionals may be nested.

At compile-time, IF compiles ØBRANCH and reserves space for an offset at addr . addr and n are used later for resolution of the offset and error testing.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

IMMEDIATE — COMPILER

"immediate"

Marks the most recently made dictionary entry as a word which will be executed when encountered rather than being compiled.

IN — **--- addr** — INPUT/OUTPUT U

"in"

Leaves the address of user variable containing the byte offset within the current input text buffer (terminal or disk) from which the next text will be accepted. WORD uses and moves the value of IN .

INTERPRET — COMPILER

"interpret"

The outer text interpreter which sequentially executes or compiles text from the input stream (terminal or mass storage) depending on STATE . If the word name cannot be found after a search of CONTEXT and then CURRENT it is converted to a number according to the current BASE . That also failing, an error message echoing the <name> with a "?" will be given.

Text input will be taken according to the convention for WORD . If a decimal point is found as part of a number, a double number value will be left. The decimal point has no other purpose than to force this action. See NUMBER .

KEY — **--- char** — INPUT/OUTPUT

"key"

Leaves the ASCII value of the next avaiable character from the active input device.

LATEST — **--- addr** — COMPILER

"latest"

Leave the name field address of the top most word in the CURRENT vocabulary.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

LEAVE — CONTROL

"leave"

Forces termination of a DO-LOOP at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains unchanged, and execution proceeds normally until LOOP or +LOOP is encountered.

LFA — **pfa --- fa** — DICTIONARY

"f-a"

Converts the parameter field address (pfa) of a dictionary definition to its link field address (fa).

LIMIT — **--- n** — MISC

Leaves the highest address plus one available in the data (or mass storage) buffer. Usually this is the highest contiguous system memory.

LIT — **--- n** — COMPILER

"lit"

Within a colon-definition, LIT is automatically compiled before each 16-bit literal number encountered in input text. Later execution of LIT causes the contents of the next dictionary address to be pushed to the stack.

LITERAL — **n --- (compiling)** — COMPILER

"literal"

If compiling, then compile the stack value n as a 16-bit literal, which when later executed will leave n on the stack. This definition is immediate so that it will execute during a colon definition. The intended use is:

: xxx [calculation] **LITERAL ;**

Compilation is suspended for the compile time calculation of a value. Compilation is then resumed and LITERAL compiles this value into the definition.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

LOOP — **addr n --- (compiling)** — **"loop"** — CONTROL I

Occurs in a colon-definition in form:

DO . . . LOOP

At run-time, LOOP selectively controls branching back to the corresponding DO based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to DO occurs until the index equals or exceeds the limit; at that time, the parameters are discarded and execution continues ahead.

At compile-time, LOOP compiles (LOOP) and uses addr to calculate an offset to DO . n is used for error testing.

M* — **n1 n2 --- d** — **"m-times"** — ARITHMETIC

A mixed magnitude math operation which leaves the double number signed product of two signed number.

M/ — **d n1 --- n2 n3** — **"m-divide"** — ARITHMETIC

A mixed magnitude math operator which leaves the signed remainder n2 and signed quotient n3, from a double number dividend d and divisor n1. The remainder takes its sign from the dividend.

M/MOD — **ud1 u2 --- u3 ud4** — **"m-divide-mod"** — ARITHMETIC

An unsigned mixed magnitude math operation which leaves a double quotient ud4 and remainder u3, from a double dividend ud1 and single divisor u2.

MAX — **n1 n2 --- max** — **"max"** — ARITHMETIC

Leaves the greater of two numbers.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
MESSAGE	**n ---** **"message"**	SECURITY
	If WARNING is positive, executes the word whose CFA is in UWARN. n may be positive or negative. If WARNING is zero, the message will simply be displayed as a number (no mass storage).	
MIN	**n1 n2 --- n3** **"min"**	ARITHMETIC
	Leaves the smaller number n3 of two numbers, n1 and n2.	
MOD	**n1 n2 --- n3** **"mod"**	ARITHMETIC
	Leaves the remainder n3 of n1 divided by n2, with the same sign as n1.	
NEGATE	**n --- -n** **"negate"**	ARITHMETIC
	Leaves the two's complement of a number, i.e. the difference of Ø less n.	
NFA	**pfa --- nfa** **"n-f-a"**	DICTIONARY
	Converts the parameter field address (pfa) of a definition to its name field address (nfa).	
NOT	**flag ---** **"not"**	COMPARISON
	Leaves a true flag (1) if the number is equal to zero, otherwise leaves a false flag. Same as Ø = .	
NUMBER	**addr --- d** **"number"**	FORMAT
	Converts a character string left at addr with a preceeding count, to a signed double precision number, using the current number BASE . If a decimal point is encountered in the text, its position will be given in DPL , but no other effect occurs. If numeric conversion is not possible, an error message will be given.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
OR	**n1 n2 --- n3** **"or"**	ARITHMETIC
	Leaves the bit-wise logical or of two 16-bit values.	
OVER	**n1 n2 --- n1 n2 n1** **"over"**	STACK
	Copies the second stack value, placing it as the new top of stack.	
PAD	**--- addr**	DICTIONARY
	Leaves the address of a scratch area used to hold character strings for intermediate processing. The maximum capacity is 64 characters.	
PFA	**nfa --- pfa** **"p-f-a"**	DICTIONARY
	Converts the name field address (nfa) of a dictionary definition to its parameter field address (pfa).	
PICK	**n --- nth** **"pick"**	STACK
	Returns the contents of the nth stack value, not counting n itself. An error conditions results for n less than one. 2 PICK is equivalent to OVER .	
QUERY	**"query"**	INPUT/OUTPUT
	Accepts input of up to 8Ø characters of text, (or until a 'return') from the keyboard into the terminal input buffer (TIB) . WORD may be used to accept text from this buffer as the input stream, by setting IN and BLK to zero.	
QUIT	**"quit"**	MISC
	Clears the return stack, stops compilation, and returns control to the entire input. No message is given.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
R@	--- n "r-fetch"	STACK

Copies the top of the return stack to the computation stack.

R>	--- n "r-from"	STACK

Removes the top value from the return stack and leaves it on the computation stack. See >R and R@ .

RØ	--- addr "r-zero"	PRIMITIVE U

Leaves the address of user variable containing the initial value of the return stack pointer. See RP! .

REPEAT	addr n --- (compiling) "repeat"	CONTROL

Used within a colon-definition in the form:

BEGIN . . . WHILE . . . REPEAT

At run-time, REPEAT forces an unconditional branch back to just after the corresponding BEGIN .

At compile-time, REPEAT compiles BRANCH and the offset from HERE to addr. n is used for error testing.

ROT	n1 n2 n3 --- n2 n3 n1 "rote"	STACK

Rotates the top three values on the stack, bringing the third to the top.

RP!	"r-p-store"	PRIMITIVE

Initializes the return stack pointer from user variable RØ .

WORD	STACK NOTATION/DEFINTION	GROUP ATTR
RP@	--- addr "r-p-fetch"	STACK
	Leaves the address of a variable containing the return stack pointer.	
S->D	n --- d "s-to-d"	ARITHMETIC
	Extends the sign of single number n to form double number d.	
SØ	--- addr "s-zero"	PRIMITIVE
	Leaves the address user variable that contains the initial value for the parameter stack pointer. See SP! .	
SCR	--- addr "s-c-r"	MASS
	Leaves the address of user variable containing the screen number most recently referenced.	
SIGN	n d --- d "sign"	FORMAT
	Inserts the ASCII "-" (minus sign) into the pictured numeric output string if n is negative. n is discarded, but double number d is maintained. Must be used between <# and #> .	
SMUDGE	"smudge"	DICTIONARY
	Used during word definition to toggle the "smudge bit" in a definitions name field. This prevents an uncompleted definition from being found during dictionary searches, until compiling is completed without error.	
SP!	"s-p-store"	STACK
	Initializes the stack pointer from SØ .	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
SP@	--- addr "s-p-fetch"	STACK
	Returns the address of the top of the stack as it was before SP@ was executed.	
SPACE	"spaces"	INPUT/OUTPUT
	Transmits an ASCII blank to the active output device.	
SPACES	n --- "spaces"	INPUT/OUTPUT
	Transmit n ASCII blanks to the active output device.	
STATE	--- addr "state"	COMPILER U
	Leaves the address of user variable containing the compilation state. A non-zero value indicates compilation.	
SWAP	n1 n2 --- n2 n1 "swap"	STACK
	Exchanges the top two values on the stack.	
TASK	"task"	DICTIONARY
	A no-operation word which can mark the boundary between applications. By forgetting TASK and re-compiling, an application can be discarded in its entirety. Its definition is : TASK ; .	
THEN	"then"	CONTROL
	Used within a colon-definition, in the form: IF ... ELSE ... THEN or IF ... THEN	
	THEN is the point where execution resumes after ELSE or IF (when no ELSE is present).	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
TIB	--- addr "t-i-b"	INPUT/OUTPUT U
	Leaves the address of user variable containing the starting address of the terminal input buffer.	
TOGGLE	addr b --- "toggle"	MEMORY
	Complements the contents of addr by the 8-bit pattern byte.	
TYPE	addr n --- "type"	INPUT/OUTPUT
	Transmits n characters beginning at addr to the active output device. No action takes place for n less than one.	
U*	un1 un2 --- ud "u-times"	ARITHMETIC
	Performs and unsigned multiplication of un1 by un2, leaving the unsigned double number product of two unsigned numbers.	
U/	ud u1 --- u2 u3 "u-divide"	ARITHMETIC
	Performs the unsigned division of double number ud by u1, leaving the unsigned remainder u2 and unsigned quotient n3 from the unsigned double dividend ud and unsigned divisor u1.	
U<	un1 un2 --- flag "u-less-than"	ARITHMETIC
	Leaves the flag representing the magnitude comparison of un1 < un2 where un1 and un2 are treated as 16-bit unsigned integers.	
UABORT	--- addr "u-abort"	PARAMETER U
	Leaves the address of the user variable containing the code field address of the ABORT word.	

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
UC/L	--- addr "u-characters-per-line"	PARAMETER U

Leaves the address of the user variable containing the number of characters per line.

UEMIT	--- addr "u-emit"	PARAMETER U

Leaves the address of the user variable containing the code field address of the EMIT output word.

UKEY	--- addr "u-key"	PARAMETER U

Leaves the address of the user variable containing code filed address of the KEY input word.

ULIMIT	--- addr "u-limit"	PARAMETER U

Leaves the address of the user variable containing the last address plus one of the data (or mass storage) buffer.

UNTIL	flag --- (run-time) addr n --- (compile-time) "until"	CONTROL

Occurs within a colon-definition in the form:

BEGIN . . . UNTIL

At run-time, if flag is true, the loop is terminated. If flag is false, execution returns to the first word after BEGIN . BEGIN-UNTIL structures may be nested.

At compile-time, UNTIL compiles ØBRANCH and an offset from HERE to addr. n is used for error tests.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

USER — **n ---** — **"user"** — DEFINING

A defining word used in the form:

n USER <name>

Which creates a user variable <name>. The parameter field of <name> contains n as a fixed offset relative to the user pointer register UP for this user variable. When <name> is later executed, it places the sum of its offset and the user area base address on the stack as the storage address of that particular variable.

VARIABLE — DEFINING

n --- <name> (compute-time)
<name> --- (run-time)
"variable"

A defining word executed in the form:

n VARIABLE <name>

to create a dictionary entry for <name> and allot two bytes for storage in the parameter field. When <name> is later executed, it will place the storage address on the stack.

VOC-LINK — **--- addr** — **"voc-link"** — VOCABULARY U

Leaves the address of user variable containing the address of a field in the definition of the most recently created vocabulary. All vocabulary names are linked by these fields to allow control for FORGETting through multiple vocabularies.

VOCABULARY — **"vocabulary"** — VOCABULARY

A defining word used in the form:

VOCABULARY <name>

to create (in the CURRENT vocabulary) a dictionary entry for <name>, which specifies a new ordered list of word definitions. Subsequent exe-

WORD	STACK NOTATION/DEFINITION	GROUP ATTR

VOCABULARY (Cont.)

cution of <name> will make it the CONTEXT vocabulary. When <name> becomes the CURRENT vocabulary (See DEFINITIONS), new definitions will be created in that list.

New vocabularies 'chain' to FORTH . This is, when all of a dictionary search through a vocabulary is exhausted, FORTH will be searched.

VLIST — VOCABULARY

"v-list"

Lists the names of the definitions in the CONTEXT vocabulary. Depression of any key will terminate the listing.

WARNING — **--- addr** — SECURITY U

"warning"

Leaves the address of user variable containing a value controlling messages. If value = 1 mass storage is present and screen 4 of drive Ø is the base location for messages. If value = Ø, no disk is present and messages will be presented by number. If value = -1, execute (ABORT) for a user specified procedure. See MESSAGE and ERROR .

WHILE — **flag --- (run-time)** — CONTROL

addr1 n1 --- addr1 n1 addr2 n2 (compile-time)

"while"

Occurs in a colon-definition in the form:

BEGIN . . . WHILE (tp) . . . REPEAT

At run-time, WHILE selects conditional execution based on Boolean flag. If flag is true (non-zero), WHILE continues execution of the true part through to REPEAT , which then branches back to BEGIN . If flag is false (zero), execution skips to just after REPEAT , exiting the structure.

At compile-time, WHILE emplaces (ØBRANCH) and leaves addr2 of the reserved offset. The stack values will be resolved by REPEAT .

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
WIDTH	--- addr "width"	SECURITY U

Leaves the address of user variable containing the maximum number of letters saved in the compilation of a definitions name. It must be 1 through 31, with a default value of 31. The name character count and its natural characters are saved, up to the value in WIDTH . The value may be changed at any time within the above limits.

WORD	char --- addr "word"	COMPILER

Receives characters from the input stream until the non-zero delimiting character in the stack is encountered or the input stream is exhausted, ignoring leading delimiters. The characters are stored as a packed string with the character count in the first character position. The actual delimiter encountered (char or null) is stored at the end of the text but not included in the count. If the input stream was exhausted as WORD is called, then a zero length will result. The address of the beginning of this packed string is left on the stack.

XOR	n1 n2 --- n3 "x-or"	ARITHMETIC

Leaves the bitwise logical exclusive or of two values.

[	"left-bracket"	COMPILER

Ends the compilation mode. The text from the input stream is subsequently executed. See] .

[COMPILE]	"bracket compile"	COMPILER

Used in a colon-definition in form:

[COMPILE] <name>

Forces compilation of the following word. This allows compilation of an IMMEDIATE word when it would otherwise be executed.

WORD	STACK NOTATION/DEFINITION	GROUP ATTR
]	**"right bracket"**	COMPILER

Sets the compilation mode. The text from the input stream is subsequently compiled. See [.

APPENDIX B

RECOMMENDED READING

BRODIE, L. (FORTH, INC.). Starting FORTH. Prentice-Hall. Englewood Cliffs, New Jersey 07632.

HARRIS, K. FORTH Extensibility, or How to Write a Compiler in 25 Words or Less. BYTE Magazine. August 1980. pp. 164-184.

INTEL CORPORATION. The 8086 Family User's Manual. Santa Clara, CA 95051.

INTEL CORPORATION. The 8086 Family User's Manual Numerics Supplement. Santa Clara, CA 95051.

JAMES, J. S. What is FORTH? A Tutorial Introduction. BYTE Magazine. August 1980. pp. 100-126.

APPENDIX C - FT-86C PIN ASSIGNMENTS

Pin Assignment of Bus Signals on 796 Bus Board Connector P1

	PIN	(COMPONENT SIDE) MNEMONIC	DESCRIPTION
Power Supplies	1	GND	Signal GND
	3	+5V	+5 Vdc
	5	+5V	+5 Vdc
	7	+12V	+12 Vdc
	9		Reserved, Bussed
	11	GND	Signal GND
Bus Controls	13	BCLK*	Bus Clock
	15	BPRN*	Bus Priority In
	17	BUSY*	Bus Busy
	19	MRDC*	Memory Read Command
	21	IORC*	I/O Read Command
	23	XACK*	XFER Acknowledge
Bus Controls and Address	25	LOCK*	Lock
	27	BHEN*	Byte High Enable
	29	CBRQ*	Common Bus Request
	31	CCLK*	Constant Clock
	33	INTA*	Intr. Acknowledge
Interrupts	35	INT6*	Parallel
	37	INT4*	Interrupt
	39	INT2*	Requests
	41	INT0*	
Address	43	ADRE*	
	45	ADRC*	
	47	ADRA*	Address
	49	ADR8*	Bus
	51	ADR6*	
	53	ADR4*	
	55	ADR2*	
	57	ADR0*	
Data	59	DATE*	
	61	DATC*	
	63	DATA*	Data
	65	DAT8*	Bus
	67	DAT6*	
	69	DAT4*	
	71	DAT2*	
	73	DAT0*	
Power Supplies	75	GND	Signal GND
	77		Reserved, Bussed
	79	-12V	-12 Vdc
	81	+5V	+5 Vdc
	83	+5V	+5 Vdc
	85	GND	Signal GND

All reserved pins are reserved for future use and should not be used if upwards compatibility is desired.

Pin Assignment of Bus Signals on 796 Bus Board Connector P1

(CIRCUIT SIDE)

PIN	MNEMONIC	DESCRIPTION
2	GND	Signal GND
4	+5V	+5 Vdc
6	+5V	+5 Vdc
8	+12V	+12 Vdc
10		Reserved, Bussed
12	GND	Signal GND
14	INIT*	Initialize
16	BPRO*	Bus Priority Out
18	BREQ*	Bus Request
20	MWTC*	Memory Write Command
22	IOWC*	I/O Write Command
24	INH1*	Inhibit 1 (Disable RAM)
26	INH2*	Inhibit 2 (Disable PROM or ROM)
28	AD10*	
30	AD11*	Address
32	AD12*	Bus
34	AD13*	
36	INT7*	Parallel
38	INT5*	Interrupt
40	INT3*	Request
42	INT1*	
44	ADRF*	
46	ADRD*	
48	ADRB*	Address
50	ADR9*	Bus
52	ADR7*	
54	ADR5*	
56	ADR3*	
58	ADR1*	
60	DATF*	
62	DATD*	
64	DATB*	Data
66	DAT9*	Bus
68	DAT7*	
70	DAT5*	
72	DAT3*	
74	DAT1*	
76	GND	Signal GND
78		Reserved, Bussed
80	-12V	-12 Vdc
82	+5V	+5 Vdc
84	+5V	+5 Vdc
86	GND	Signal GND

J1 CONNECTIONS

PIN #	SIGNAL	PIN #	SIGNAL
1	Ground	26	Spare
2	Ground	27	Spare
3	Spare	28	Spare
4	External Reset	29	Spare
5	Spare	30	Spare
6	Spare	31	Spare
7	Spare	32	Spare
8	Spare	33	Transmit Data Channel "B"
9	Spare	34	Spare
10	Spare	35	Data Terminal Ready Channel "A"
11	Spare	36	Spare
12	Spare	37	Data Terminal Ready Channel "B"
13	Spare	38	Receive Data Channel "B"
14	Spare	39	Request to Send Channel "A"
15	Spare	40	Data Carrier Detect Channel "B"
16	Spare	41	Transmit Data Channel "A"
17	Spare	42	Clear to Send Channel "B"
18	Spare	43	Request to Send Channel "B"
19	Spare	44	Transmit Clock Channel "B"
20	Spare	45	Clear to Send Channel "A"
21	Spare	46	Receive Clock Channel "B"
22	Spare	47	Data Carrier Detect Channel "A"
23	Spare	48	Receive Data Channel "A"
24	Spare	49	Receive Clock Channel "A"
25	Spare	50	Transmit Clock Channel "A"

Pin Assignment of Bus Signals cn 796 Bus Board Connector P2

(CIRCUIT SIDE)

PIN	MNEMONIC	DESCRIPTION
2		Reserved, Not Bussed
4		Reserved, Not Bussed
6		Reserved, Not Bussed
8		Reserved, Not Bussed
10		Reserved, Not Bussed
12		Reserved, Not Bussed
14		Reserved, Not Bussed
16		Reserved, Not Bussed
18		Reserved, Not Bussed
20		Reserved, Not Bussed
22		Reserved, Not Bussed
24		Reserved, Not Bussed
26		Reserved, Not Bussed
28		Reserved, Not Bussed
30		Reserved, Not Bussed
32		Reserved, Not Bussed
34		Reserved, Not Bussed
36		Reserved, Not Bussed
38		Reserved, Not Bussed
40		Reserved, Not Bussed
42		Reserved, Bussed
44		Reserved, Bussed
46		Reserved, Bussed
48		Reserved, Bussed
50		Reserved, Bussed
52		Reserved, Bussed
54		Reserved, Bussed
56	ADR17*	Address Bus
58	ADR15*	
60		Reserved, Bussed

J1 CONNECTIONS

PIN #	SIGNAL	PIN #	SIGNAL
1	Ground	26	Spare
2	Ground	27	Spare
3	Spare	28	Spare
4	External Reset	29	Spare
5	Spare	30	Spare
6	Spare	31	Spare
7	Spare	32	Spare
8	Spare	33	Transmit Data Channel "B"
9	Spare	34	Spare
10	Spare	35	Data Terminal Ready Channel "B"
11	Spare	36	Spare
12	Spare	37	Data Terminal Ready Channel "A"
13	Spare	38	Receive Data Channel "B"
14	Spare	39	Request to Send Channel "A"
15	Spare	40	Data Carrier Detect Channel "B"
16	Spare	41	Transmit Data Channel "A"
17	Spare	42	Clear to Send Channel "B"
18	Spare	43	Request to Send Channel "B"
19	Spare	44	Transmit Clock Channel "B"
20	Spare	45	Clear to Send Channel "A"
21	Spare	56	Receive Clock Channel "B"
22	Spare	47	Data Carrier Detect Channel "A"
23	Spare	48	Receive Data Channel "A"
24	Spare	49	Receive Clock Channel "A"
25	Spare	50	Transmit Clock Channel "A"