

SIGNETICS
8X300
EVALUATION KIT
MANUAL

\$3.00

TABLE OF CONTENTS

8X300KT100SK DATA SPECIFICATIONS	3	
8X300 DATA SPECIFICATIONS	10	
INTERFACE COMPONENTS		
8T26A/8T28	24	
8T31	27	
8T32/33/35/36	30	
MEMORY COMPONENTS		
PROM	37	
82S114/115	39	
RAMs	43	
82S16/17/116/117	46	
SSI/MSI COMPONENTS		47
54/7400	49	
54/7427	49	
54/7474	50	
54/74157	52	
APPLICATIONS MEMOS		53
AH3 Input/Output Design	54	
AH4 8X300 Applications	56	
MP3 Understanding the 8X300 Instruction Set	71	
SP1 The 8X300 Cross Assembly Program	78	
8X300 A Fast Microprocessor for Control Applications	85	
SMS DATA SPECIFICATIONS		
SMS 3000 Microcontroller Simulator	91	
SMS 3300 Microcontroller Monitor	93	
Sales Office List	95	

DESCRIPTION

The Signetics 8X300 Fixed Instruction Bipolar Microprocessor is a high performance microprocessor optimized for control and data movement applications.

The unique features of the 8X300 IV bus and instruction set permit 8-bit parallel data to be rotated or masked before undergoing arithmetic or logical operations. Then, the data may be shifted and merged into any field of from 1 to 8 contiguous bits at the destination. The entire process of input, shifting, processing and output is accomplished in one instruction cycle time. The 250ns cycle time makes the 8X300 suited for high speed applications.

The evaluation board contains all the elements which a designer needs to familiarize himself with the 8X300 for his systems applications. Included with the 8X300 are 4 I/O ports for external device interface, 256 bytes of temporary (working) data storage and 512 words of program storage, all properly

connected to the 8X300 to allow immediate exercising of the board. For this purpose, the PROMs are preprogrammed with the I/O control, RAM control and RAM integrity diagnostic programs. With the remaining PROM space, the designer may enter his own benchmark, test or development routines.

The board design allows complete flexibility in access to the address, instruction and IV busses as well as all controls and signals of the 8X300. The IV bus, I/O port user connection, clock signals, control lines, address bus and instruction bus are wired to output pins, the board edge connector and flat cable connectors.

The board layout permits variations and/or expansion of the basic design. In addition to the access to all signals for transfer off the board, a wire wrap area is provided so that the designer may add to the board circuitry as he desires. The addition may include memory, additional interfaces, or special circuits which meet specific user requirements.

Controls are also provided for diagnostic and instructional purposes by allowing various operating modes. In the WAIT mode, the program may be single stepped for ease of checkout. The one-shot instruction jamming allows control of the program start location, changes of program flow, changing or examining the internal registers, or testing of simple sequences. The repeated instruction jamming provides a means of repetitive execution of an instruction so that the I/O bus and the control lines may be examined without software changes. In both of these jam cases, the jammed instruction is selected by board-mounted switches.

Auxiliary Circuits

The 8X300 can be used with any bipolar (or TTL-compatible) ICs. It can directly address 8192 program instruction locations and up to 512 I/O ports. Memory paging may be employed for larger working storage. Typical auxiliary circuits include:

DEVICE	MFG	PART NUMBER	DESCRIPTION	QTY
U14	Signetics	N8X300I	Microprocessor	1
U2-U9		N82S116B	RAM (256X1)	8
U15-16		N82S115I	PROM (512X8)	2
U13, U21, U26, U27		N8T32N	I/O Port (Tri-state)	4
U1		N8T31N	Latch	1
U10-U11		N8T26AB	Transceiver	2
U17-U20		N74LS157B	Multiplexer	4
U23, U25		N74LS74A	Flip-Flop	2
U12		N74LS27A	NOR Gate	1
U22, U24	Signetics	N74LS00A	NAND Gate	2
Q1		2N5320	Transistor	1
Q2		2N2222	Transistor	1
RN1-RN3	Dale	CSP-10E-01-102-K	Resistor network	3
R1, R3, R4		1kΩ 1/4W	Resistor	3
R5		4kΩ 1/4W	Resistor	1
R2		8kΩ 1/4W	Resistor	1
C2-C20, C22-C30		0.1μF	Capacitor	29
C1, C21		22μF	Capacitor	2
Z1		8.00 MHz	Crystal	1
J104	Berg	65616-134	Connector	1
J101	Berg	65616-150	Connector	1
J105	Berg	65616-120	Connector	4
	Robinson-Nugent	ICN-163-53	Socket	2
	Robinson-Nugent	ICN-246-54	Socket	7
	Robinson-Nugent	SB-25	Strip socket	2
S1, S2	Amp	435640-5	Dip switch	2
S5, S6	Alco	MSTS-104D	Toggle switch	2
S3, S4	Digitast	ST	Momentary switch	2
	Smith	230 (red)	Binding post	1
	Smith	230 (black)	Binding post	1
	SAE	CPH 8100-100	50/100 edge connector	1
			PC board	1
			Packing case	1
			Assembly manual	1

Table 1 KIT CONTENTS

Program Storage
82S115 (512X8 PROM)

I/O
8T32/33 (8-Bit Synchronous Bidirectional I/O Port)
8T35/36 (8-Bit Asynchronous Bidirectional I/O Port)
8T31 (8-Bit Bidirectional I/O Port)
8T39 (Quad Bus Extender)

Working Storage
82S116 (256X1 RAM)

KIT ASSEMBLY

The kit is designed to be assembled by a skilled technician. To aid assembly as well as the evaluation, the major board areas are identified and part placement is indicated directly on the PC board. The board has been solder masked so that it may be wave soldered and to avoid solder bridges if the board is hand soldered. Sockets are provided to mount the following parts: 8X300, 8T32 (4), 8T31, 82S115 (2), and the DIP Switches (2). Kit assembly is straightforward and may be accomplished in about four hours.

Figure 4 shows the component side of the board and the position and orientation of all parts. Assembly can be accomplished using this figure, the notes and observing the board markings. Figure 5 provides the schematic.

SOFTWARE

The PROMs furnished with the kit are programmed with diagnostics in the first 50 locations to assist in verifying that the assembled kit is working correctly. The remaining 462 locations may be programmed by the user.

The diagnostics are separated into two parts: Locations 00-23 contain tests of the I/O ports and the interface to the RAM, locations 24-47 write all combinations of bits into all locations of RAM and test the data read back. Locations 50-61 contain a delay routine which is used by the memory test to allow monitoring of the test with an external LED array or logic analyzer.

Execution of the first part should be checked in Single Step. Figures 1, 2 and 3 are flow diagrams for the diagnostic coding.

CONTROLS

Run/Wait

With this switch in the WAIT position, the processor halts execution of instructions and holds all buses in their current state (MCLK will still be active). When the switch is returned to the RUN position, the processor will remain in the halted state until the STEP key is depressed. The processor then begins running normally and subsequent depressions of the STEP key have no effect.

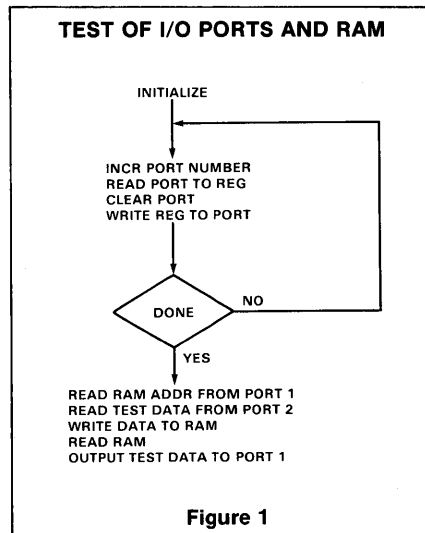


Figure 1

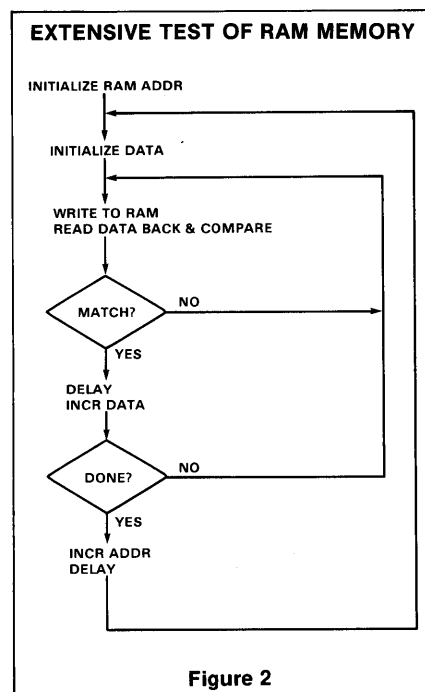


Figure 2

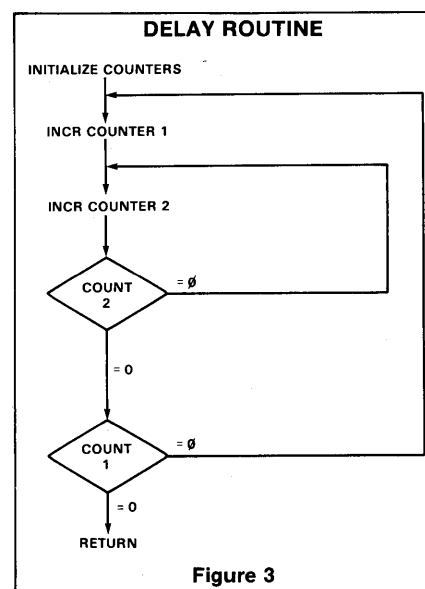


Figure 3

Single Step

Instructions may be executed one at a time by depressing the STEP key while the processor is halted. Instructions can be read from either the PROM or the Instruction switches.

Instruction Insertion

Although instructions are normally fetched from the PROMs, the bank of DIP switches provides an alternate instruction source. The command that is set in the switches is jammed on the instruction bus for either 1 cycle or indefinitely, depending on the setting on the SINGLE/REPEAT switch.

With the switch in the SINGLE position, the instruction encoded in the switches is placed on the bus for 1 cycle time when the INSERT key is pressed, then control returns to the PROMs. With the switch in the REPEAT position, the instruction will be executed repeatedly until the switch is returned to the SINGLE position. This feature allows examination of the control signals without changing the software.

Connection to the I/O Ports

The I/O ports are 8T32s programmed with addresses 1 through 4. It is recommended that \overline{BOC} (J105 pin 18) be tied low for most tests and that \overline{BIC} (J105 pin 20) be pulled low when data is to be entered to the port from an external device. This scheme allows output from the port to be monitored constantly yet data can be entered at any time since \overline{BIC} overrides \overline{BOC} (see 8T32 data sheet).

USAGE

By external means, a test pattern is loaded into each of the I/O ports. As the program is stepped through, each of the ports is read, cleared, then restored to verify proper connection of the control and data buses. If the connections are properly made, the test pattern that the user entered into Port 1 will be cleared after 10 steps and restored on the 11th. The other 3 ports are tested in sequence after 6 more steps per port. The memory is tested in a similar manner for continuity of the control signals by reading an address from Port 1 and a test pattern from Port 2, writing the pattern to the addressed location, reading the data back and writing it to Port 1. If the data in Port 1 matches the data in Port 2, the control interface to the RAM is verified. The flow diagram for this test is given in Figure 1.

The next portion of the test exercises each location in the RAM with all possible combination of bits. If the pattern read back does not match the data written, the test will be repeated with the same address and pattern until successful. The RAM address being written is output to Port 1, the pattern to Port 2. S5 should be in the RUN mode for this test. The program includes a delay loop so that its progress may be monitored by a simple LED display connected to the 8T32 outputs (Ports 1 and 2).

COMPONENT SIDE OF 8X300 PC BOARD

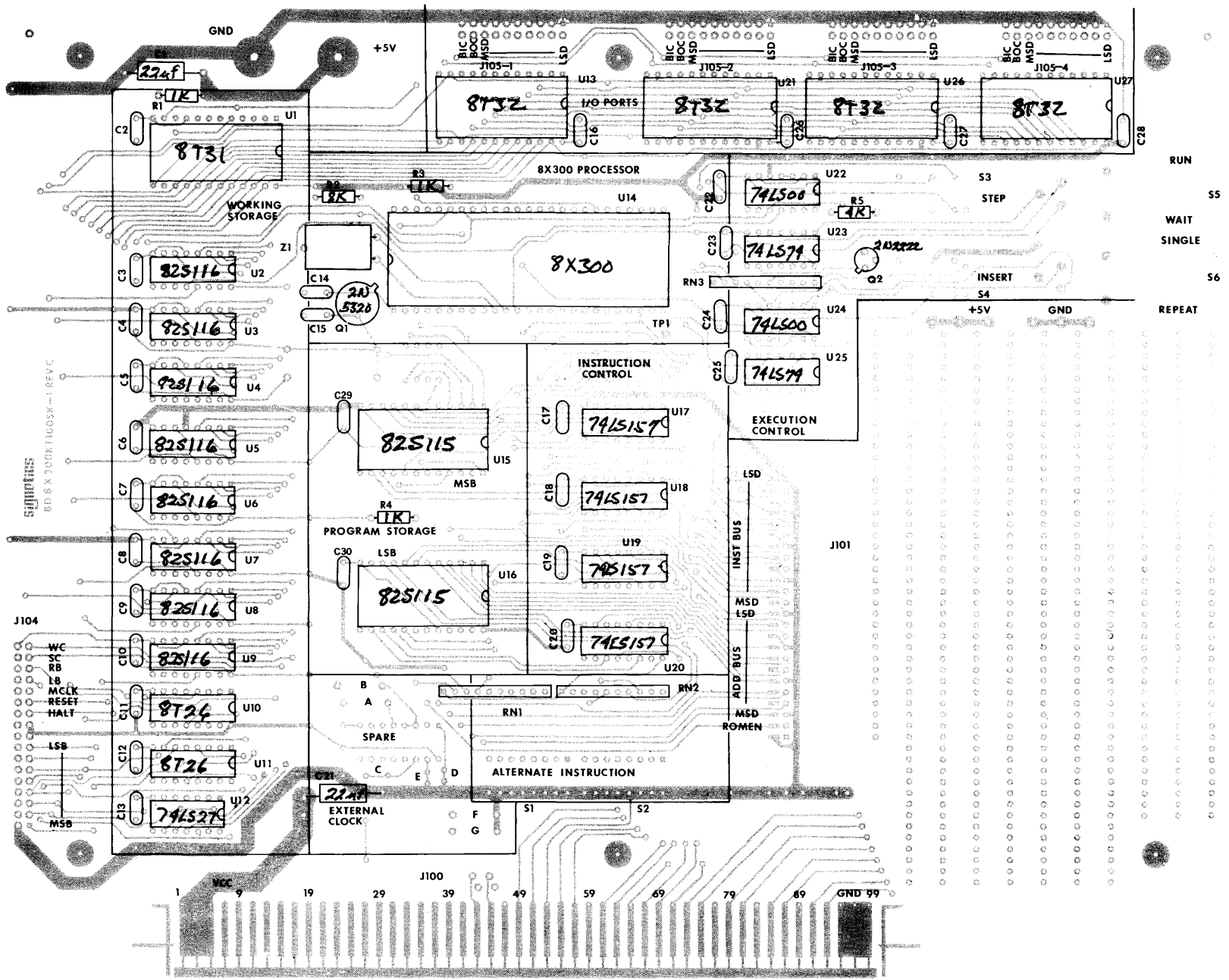
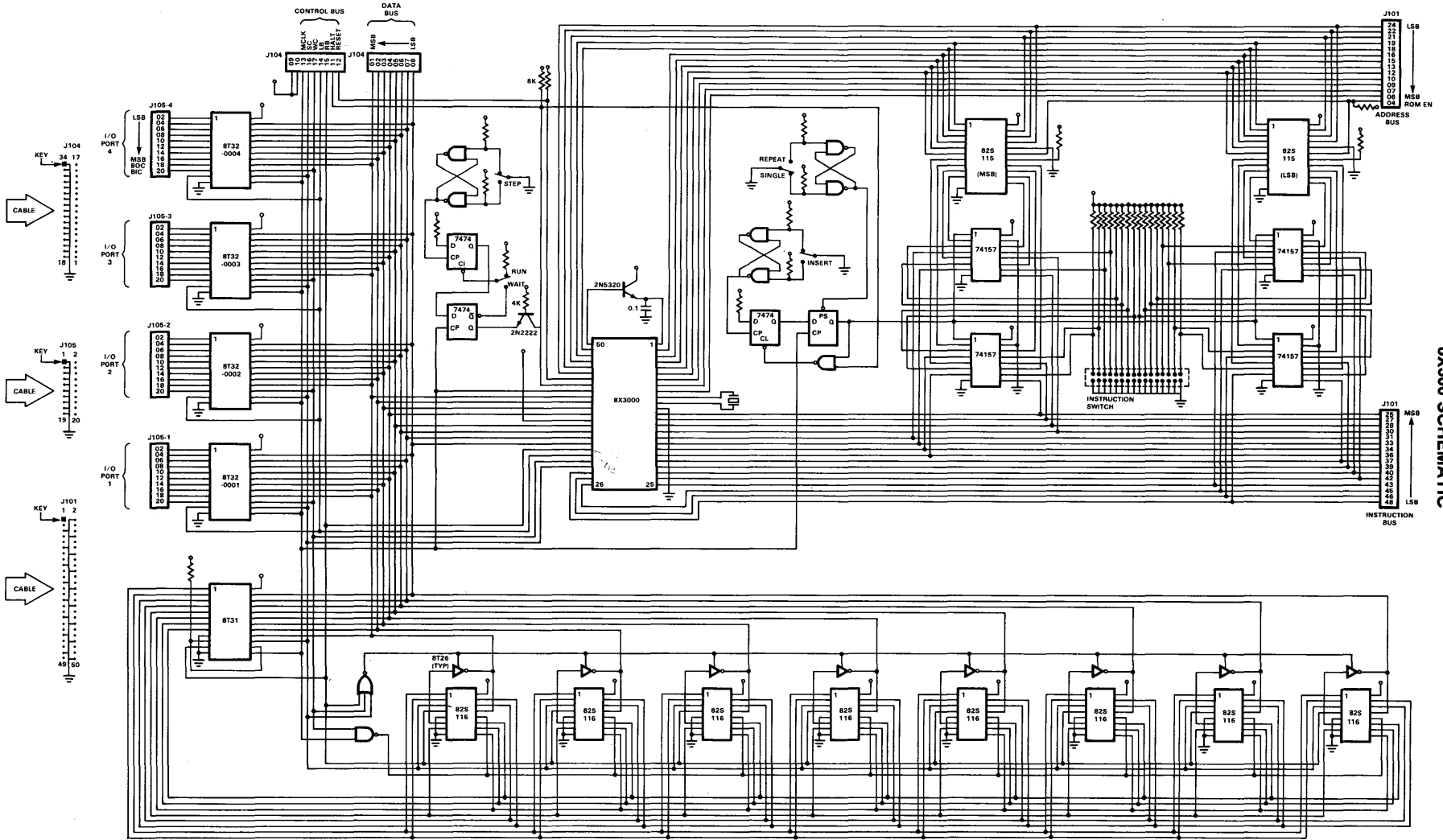


Figure 4

signetics

Figure 5



8300 SCHEMATIC

PIN NO.	FUNCTION	PIN NO.	FUNCTION
1	VCC	2	VCC
3	VCC	4	VCC
5	VCC	6	VCC
7	_____	8	D07 (MSB)
9	_____	10	D06
11	_____	12	D05
13	_____	14	D04
15	_____	16	D03
17	_____	18	D02
19	_____	20	D01
21	_____	22	D00 (LSB)
23	_____	24	RESET
25	_____	26	HALT
27	_____	28	MCLK
29	_____	30	LB
31	_____	32	RB
33	_____	34	SC
35	_____	36	WC
37	GND	38	GND
39	_____	40	CLK OUT
41	_____	42	CLK IN
43	SPARE	44	SPARE
45	SPARE	46	SPARE
47	GND	48	GND
49	I00	50	_____
51	I01	52	_____
53	GND	54	GND
55	I04	56	A00
57	I05	58	A01
59	GND	60	GND
61	I08	62	A04
63	I09	64	A05
65	GND	66	GND
67	I12	68	A08
69	I13	70	A09
71	GND	72	GND
73	I15	74	A12
75	I14	76	ROM EN
77	GND	78	GND
79	I11	80	A11
81	I10	82	A10
83	GND	84	GND
85	I07	86	A07
87	I06	88	A06
89	GND	90	GND
91	I03	92	A03
93	I02	94	A02
95	GND	96	GND
97	GND	98	GND
99	GND	100	GND

Table 2 J100 CONNECTIONS

To begin the test sequence, the DIP switches are loaded with 700000₈ (a jump to address 0), as shown in Figure 6. Switches S5 and S6 are placed in the WAIT and SINGLE positions respectively. After power is applied to the board, S4 must be pressed to insert the jump to 0. The system is now ready to be stepped through the diagnostic programs. A listing of the diagnostic coding is given in Table 3.

External Clock Synchronization

The 8X300 board provides for clock synchronization with external logic by means of the "spare" IC location at the lower center part of the board.

To drive the 8X300 from an external TTL level clock source, install an 8T98 driver in the "spare" location, place 100Ω resistors at points A and B and 47Ω resistors at points F and G. Trace C may be cut to reduce the loading on MCLK is desired. Signal is input on J100 pin 42.

To synchronize external logic to the 8X300 with MCLK, install an 8T98 driver in the "spare" location. No traces need to be cut or jumpered. Signal is available on J100 pin 40.

Use with MCSIM*

Remove 8X300 from PC board and plug MCSIM cable into J104. Set on-board controls to RUN. When power is applied, MCSIM is ready to run according to the usual MCSIM procedures.

Use with SMS Monitor*

The 8X300 remains in the PC board when the SMS monitor is used. The panel connects to J104 and J101. Set the on-board controls to RUN and SINGLE, the procedure according to the monitor operating instructions. If the board is halted, depress the STEP switch to initiate.

JUMPERS FOR 82S114/82S115 PROMs

The 8X300 Evaluation Board will accommodate either 82S114 or 82S115 PROMs. Determine which memory was included in this kit and connect the circuit according to the appropriate drawing in Figure 7. Use insulated wires approximately 0.5" long with all but 0.3" insulation stripped off. Put wires in place, solder and trim.

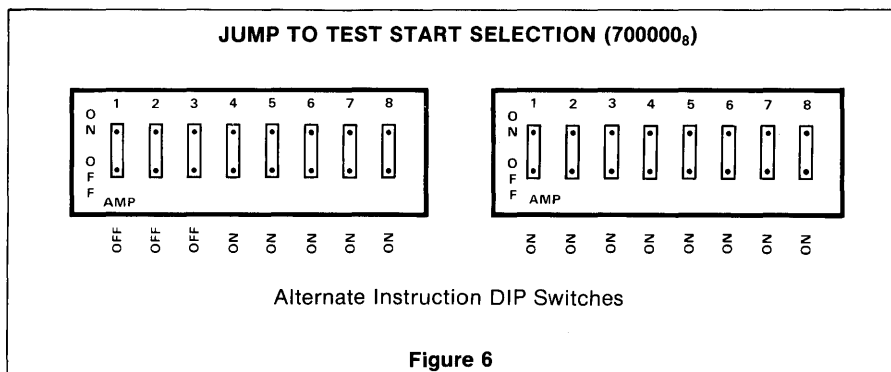


Figure 6

*Scientific Microsystems, 520 Clyde Avenue, Mt. View, Calif. 94043

ADDRESS	CODE	INSTRUCTION	COMMENT
TEST OF I/O PORTS & RAM			
000	6 06374	XMIT -4, R6	Initialize Counter
001	6 11000	XMIT 0, R11	Initialize Port Register
002	6 00001	XMIT 1, AUX	Load Increment
003	1 06006	Q1 ADD R6, R6	Incr Count
004	1 11011	ADD R11, R11	Incr Port Number
005	0 11007	MOVE R11, IVL	Select Port
006	0 27001	MOVE LB, R1	Read Port
007	6 27000	XMIT 0, LB	Clear Port
010	0 01027	MOVE R1, LB	Write Port
011	5 06003	NZT R6, Q1	All Done ?
012	6 07001	XMIT 1, IVL	Address Port 1
013	0 27001	MOVE LB, R1	Read RAM Addr
014	6 07002	XMIT 2, IVL	Addr Port 2
015	0 27002	MOVE LB, R2	Read RAM Data
016	0 01017	MOVE R1, IVR	Addr RAM
017	0 02037	MOVE R2, RB	Write RAM
020	6 02000	XMIT 0, R2	Clear R2
021	6 07001	XMIT 1, IVL	Addr Port 1
022	0 37002	MOVE RB, R2	Read RAM
023	0 02027	MOVE R2, LB	Output Data to Port 1
BEGINNING OF MEMORY TEST			
024	6 01000	XMIT 0, R1	Initialize RAM Addr
025	6 07001	Q2 XMIT 1, IVL	Addr Port 1
026	0 01027	MOVE R1, LB	Output RAM Addr
027	0 27017	MOVE LB, IVR	Addr RAM
030	6 02000	XMIT 0, R2	Initialize RAM Data
031	6 07002	XMIT 2, IVL	Addr Port 2
032	0 02027	Q3 MOVE R2, LB	Output RAM Data
033	0 02037	MOVE R2, RB	Write to RAM
034	0 02000	MOVE R2, AUX	Move Data to AUX
035	3 37000	XOR RB, AUX	Compare
036	5 00032	NZT AUX, Q3	Test
037	6 11000	XMIT 0, R11	Set up Return Indicator
040	7 00050	JMP DELAY	Delay for display purposes
041	6 00001	XMIT 1, AUX	Load Incr
042	1 02002	ADD R2, R2	Incr Data
043	5 02032	NZT R2, Q3	Done?
044	1 01001	ADD R1, R1	Incr Addr
045	6 11001	XMIT 1, R11	Set up Return Indicator
046	7 00050	JMP DELAY	Delay
047	7 00025	JMP Q2	Repeat
050	6 03000	DELAY XMIT 0, R3	
051	6 04000	XMIT 0, R4	
052	6 00001	XMIT 1, Aux	
053	1 03003	D1 ADD R3, R3	
054	1 04004	D2 ADD R4, R4	
055	5 04054	NZT R4, D2	
056	5 03053	NZT R3, D1	
057	4 11060	RTN XEC *+1, R11	
060	7 00041	JMP 041	
061	7 00047	JMP 047	

Table 3 SYSTEM TEST

JUMPERS FOR 82S114/115 PROMS

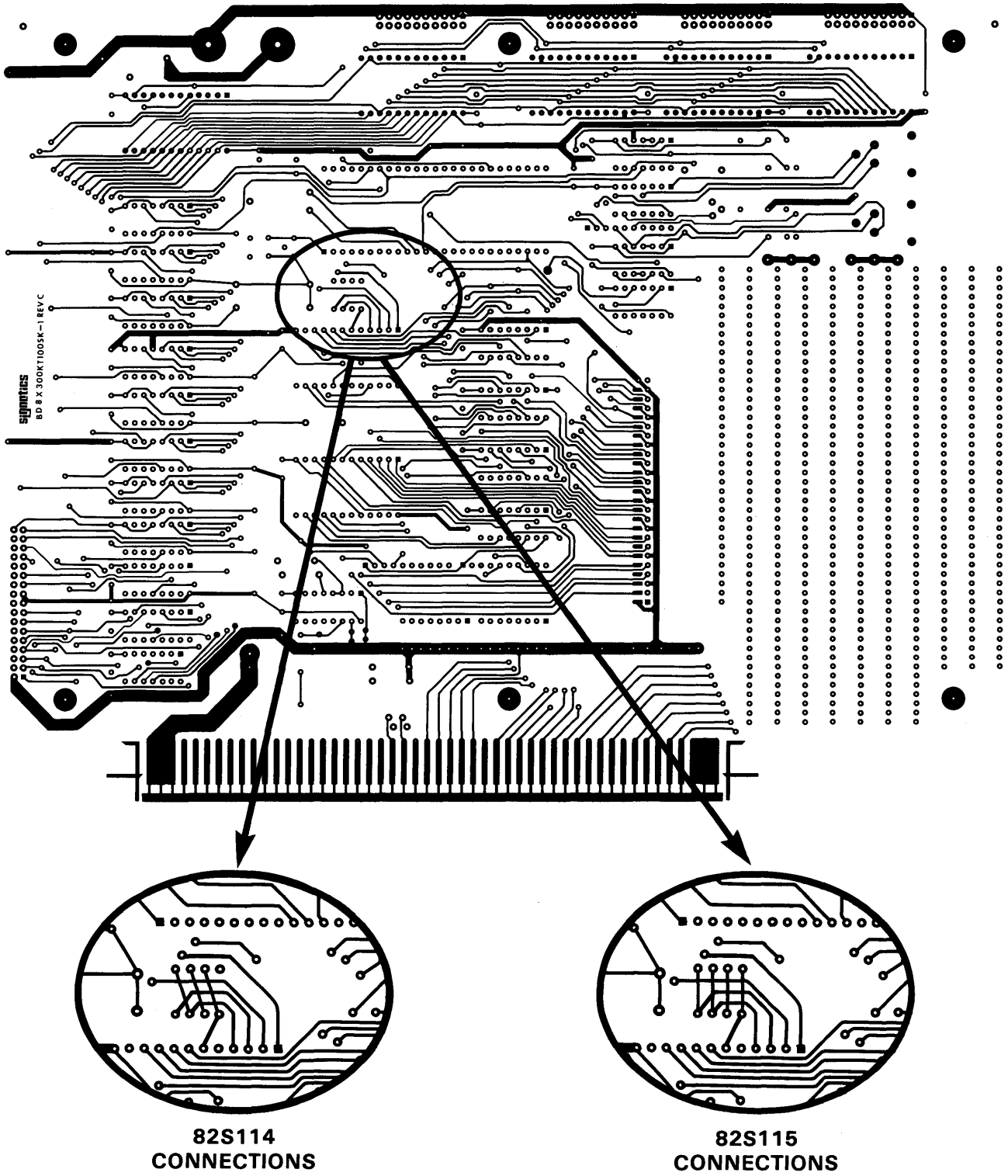


Figure 7

INTRODUCTION

A Microcomputer Designed for Control

The 8X300 is a microcomputer designed for control. It features:

Execution Speed

- 250ns instruction execution time
- Direct address capability—up to 8192 16-bit words of program memory
- Eight 8-bit general purpose registers
- Simultaneous data transfer and data edit in a single instruction cycle time
- n-way branch or n-entry table lookup in 2 instruction cycle times
- 8X300 instructions operate with equal speed on 1-bit, 2-bit, 3-bit, 4-bit, 5-bit, 6-bit, 7-bit, or 8-bit data formats

The 8X300 instruction set features control-oriented instructions which directly access variable length input/output and internal data fields. These instructions provide very high performance for moving and interpreting data. This makes the 8X300 ideal in switching, controlling, and editing applications.

Direct Processing of External Data

The 8X300 I/O system is treated as a set of internal registers. Therefore data from external devices may be processed (tested, shifted, added to, etc.) without first moving them to internal storage. In fact, the entire

concept is to treat data at the I/O interface no differently than internal data. This concept extends to the software which allows variables at the input/output system to be named and treated in the same way as data in storage.

Separate Program Storage and Data Storage

The storage concept of the 8X300 is to separate program storage from data storage. Program storage is implemented in read-only memory in recognition of the fact that programs for control applications are fixed and dedicated. The benefits of using read-only memory are that great speeds may be obtained at lower cost than if read/write memory were used, and that program instructions reside in a non-volatile medium and cannot be altered by system power failures.

8X300 Architecture

Figure 1 of the 8X300 data sheet illustrates the 8X300 architecture. The 8X300 contains an Arithmetic Logic Unit (ALU), Program Counter, and an Address Register. Eight 8-bit general purpose registers are also pro-

vided, including 7 working registers and an auxiliary register which performs as a working register and also provides an implied operand for many instructions. The 8X300 registers are shown in Figure 1 of the 8X300 data sheet and are summarized below:

Control Registers include:

- Instruction—A 16-bit register containing the current instruction
- Program Storage Address Register (AR)—A 13-bit register containing the address of the current instruction being accessed from Program Storage
- Program Counter (PC)—A 13-bit register containing the address of the next instruction to be read from Program Storage

Data Registers include:

- Working Registers (WR)—Seven 8-bit registers for data storage
- Overflow (OVF)—A 1-bit register that retains the most significant bit position carry from ALU addition operation. Arithmetically treated as 2^0 .
- Auxiliary (AUX)—An 8-bit register. Source of implied operand for arithmetic and logical instructions. May be used as a working register.

A crystal external to the CPU is used to generate the CPU system clock. The CPU executes 8 instruction types.

DESCRIPTION

The Signetics 8X300 Interpreter is a monolithic, high-speed microprocessor implemented with bipolar Schottky technology. As the central processing unit, CPU, it allows 16-bit instructions to be fetched, decoded and executed in 250ns. A 250ns instruction cycle requires maximum memory access of 65ns, and maximum I/O device access of 35ns.

Interpreter instructions operate on 8-bit, parallel data. Logic is distributed along the data path within the Interpreter. Input data can be rotated and masked before being subject to an arithmetic or logical operation; and output data can be shifted and merged with the input data, before being output to external logic. This allows 1- to 8-bit I/O and data memory fields to be accessed and processed in a single instruction cycle.

PROGRAM STORAGE INTERFACE

Program Storage is typically connected to the A0-A12 (A12 is least significant bit) and I0-I15 signal lines. An address output on A0-A12 identifies one 16-bit instruction word in program storage. The instruction word is subsequently input on I0-I15 and defines the interpreter operations which are to follow.

The Signetics 82S115 PROM, or any TTL compatible memory, may be used for program storage.

I/O DEVICES INTERFACE

An 8-bit I/O bus, called the Interface Vector (IV) data bus, is used by the Interpreter to communicate with 2 fields of I/O devices. The complementary LB and RB signals identify which field of the I/O devices is selected.

Both I/O data and I/O address information can be output on the IV bus. The SC and WC signals are typically used to distinguish between I/O data and I/O address information as follows:

SC WC

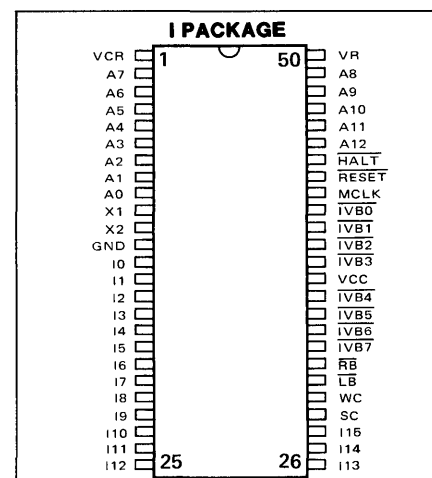
1	0	I/O address is being output on IV bus
0	1	I/O data is being output on IV bus
0	0	I/O data is expected on the IV, bus, as input to the Interpreter
1	1	Not generated by the Interpreter

The Signetics 82SXXX series RAM, and the 8T32/33 may be attached to the IV bus.

FEATURES

- 185ns instruction decode and execute delay (with Signetics 8T32/33 I/O port)
- Eight 8-bit working registers
- Single instruction access to 1-bit, 2-bit, 3-bit . . . or 8-bit field on I/O bus
- Separate instruction address, instruction, and I/O data busses
- On-chip oscillator
- Bipolar Schottky technology
- TTL inputs and outputs
- Tri-state output on I/O data bus
- +5 volt operation from 0° to 70° C

PIN CONFIGURATION



PIN DESCRIPTION

PIN	SYMBOL	NAME AND FUNCTION	TYPE
2-9, 45-49	A0-A12:	Instruction address lines. A high level equals "1." These outputs directly address up to 8192 words of program storage. A12 is least significant bit.	Active high
13-28	I0-I15:	Instruction lines. A high level equals "1." Receives instructions from Program Storage. I ₁₅ is least significant bit.	Active high
33-36, 38-41	$\overline{IVB0}$ - $\overline{IVB7}$	Interface Vector (IV) Bus. A low level equals "1." Bidirectional tri-state lines to communicate with I/O devices. $\overline{IVB7}$ is least significant bit.	Three-state Active low
42	\overline{MCLK} :	Master Clock. Output to clock I/O devices, and/or provide synchronization for external logic	
30	WC:	Write Command. High level output indicates data is being output on the IV Bus.	Active high
29	SC:	Select Command. High level output indicates that an address is being output on the IV Bus.	Active high
31	\overline{LB} :	Left Bank. Low level output to enable one of two sets of I/O devices (\overline{LB} is the complement of RB).	Active low
32	\overline{RB} :	Right Bank. Low level output to enable one of two sets of I/O devices (\overline{RB} is the complement of \overline{LB}).	Active low
44	\overline{HALT} :	Low level is input to stop the Interpreter.	Active low
43	\overline{RESET} :	Low level is input to initialize the Interpreter.	Active low
10-11	X1, X2:	Inputs for an external frequency determining crystal. May also be interfaced to logic or test equipment.	
50	VR	Reference voltage to Pass Transistor.	
1	VCR	Regulated output voltage from Pass Transistor.	
37	VCC:	5V power connection.	
12	GND:	Ground.	

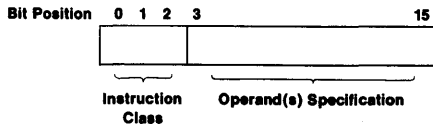
INSTRUCTION CYCLE

Each interpreter operation is executed in 1 instruction cycle, which may be as short as 250ns. The Interpreter generates MCLK to synchronize external logic to the instruction cycle. Instruction cycles are subdivided into quarter cycles. MCLK is an output during the last quarter cycle.

During the third quarter cycle of an instruction, an address is output on A0-A12, identifying the location in program storage of the next instruction word. This instruction word defines the next instruction, which must be input on I0-I15 during the first quarter cycle of the next instruction cycle (see Table 1).

Instruction Set Summary

The 16-bit instruction word input on I0-I15 is decoded by the instruction decode logic to implement events that are to occur during the remainder of the instruction cycle. Generally the 16-bit instruction word is decoded as follows:



A detailed usage of the 13 "operand(s) specification" bits is given in following sections.

Three operation code bits allow for 8 instruction classes. The 8 instruction classes are summarized in Table 2. Each entry is referred to as an "instruction class" because the unique architecture of the Interpreter allows a number of powerful variations to be specified by the 13 operand(s) specification bits. A complete description of instruction formats and some instruction examples are provided in the Applications Guide.

Data Processing

The Interpreter architecture includes eight 8-bit working registers, an arithmetic logic unit (ALU), an overflow register, and the 8-bit IV Bus. Internal 8-bit data paths connect the registers and IV Bus to the ALU inputs, and the ALU output to the registers and IV Bus. Data processing logic is distributed along these internal 8-bit data paths. Rotate and mask logic precedes the ALU on the data entry path. Shift and merge logic precedes the ALU on the data output path. Shift and merge logic follows the ALU on the data output path. All 4 sets of logic can operate on 8 data bits in a single instruction cycle. (See Figure 2)

When less than 8 bits of data are specified for output to the IV bus by the ALU, the data field (shifted if necessary) is inserted into the prior contents of the IV bus latches. The

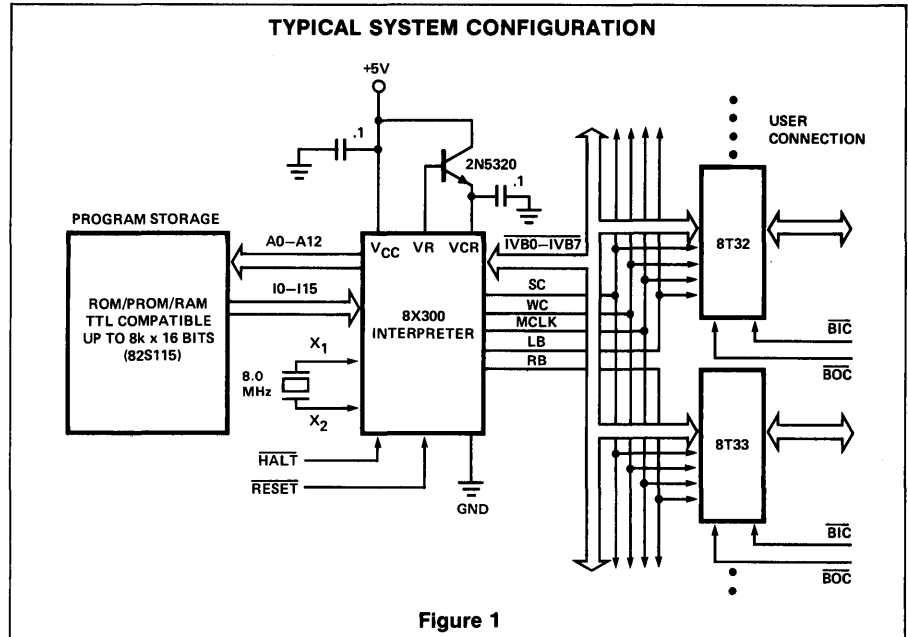


Figure 1

INST. AND IV BUS DATA INPUT	DATA PROCESSING	ADDR. AND IV BUS CHANGING	ADDR. AND IV BUS DATA VALID MCLK = HIGH
← ¼ cycle →	← ¼ cycle →	← ¼ cycle →	← ¼ cycle →

Table 1 INSTRUCTION CYCLE

IV bus latches contain data input at the start of an instruction. This data in the IV bus latches will be specified in the instruction as a) IV bus source data or b) data from an automatic read when the IV bus is specified as a destination. Therefore, IV bus bit positions outside an inserted bit field are unmodified.

Data Addressing

Sources and destinations of data are specified using a 5-bit octal number, as shown in Table 2. The source and/or destination of data to be operated upon is specified in a single instruction word.

Referring to Table 3, the Auxiliary register (address 00) is the implied source of the second argument for ADD, AND or XOR operations.

IVL and IVR are write-only registers used only as a destination. They have addresses and are treated as registers, but in reality they do not exist. When IVL is specified as a destination or the D field = 20-27₈, then LB = 'low', RB = 'high' are generated; when IVR is specified as a destination or the D field = 30-37₈, then RB = low, LB = 'high' are generated.

When IVL or IVR is specified as the destination in an instruction, SC is also activated

and data is placed on the IV bus. If IVL or IVR is specified as a source of data, the source data is all zeroes.

INSTRUCTION SEQUENCE CONTROL

The Address Register and Program Counter are used to generate addresses for accessing an instruction. The Address Register is used to form the instruction address, and in all but 3 instructions (XEC, NZT, and JMP) the address is copied into the Program Counter. The instruction address is formed in 1 of 3 ways:

1. For all instructions but the JMP, XEC, and a satisfied NZT, the Program Counter is incremented by 1 and placed in the Address Register.
2. For the JMP instruction, the full 13-bit address field from the JMP instruction is placed into the Address Register and copied into the Program Counter.
3. For the XEC and NZT instructions, the high order 5- or 8-bits of the Program Counter are combined with 8- or 5-lower-order bits of ALU output (XEC or NZT) and placed in the Address Register. For the NZT instruction, it is also copied into the Program Counter.

INSTRUCTION MNEMONIC	OP CODE	FORMATS	DESCRIPTION	I/O CONTROL SIGNALS	-- INSTRUCTION CYCLE --					
					Instruction Input and Data Processing	Address/IV Bus Output				
MOVE	0	Register to Register 0 23 78 10 11 15 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 25%;">0</td> <td style="width: 25%;">S</td> <td style="width: 25%;">R</td> <td style="width: 25%;">D</td> </tr> </table> S ≠ 07,17,20-37 ₈ D ≠ 10,20-37 ₈	0	S	R	D	(S) → D Move contents of register specified by S to register specified by D. Right rotate contents of register S by R places before operation.	SC = 0 WC = 0 LB/RB = X LB/RB = X	0 0 X X	1 if D = 07,17 0 1 if D = 17 0 if D = 07
		0	S	R	D					
		IV Bus to Register: 0 23 78 10 11 15 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 25%;">0</td> <td style="width: 25%;">S</td> <td style="width: 25%;">L</td> <td style="width: 25%;">D</td> </tr> </table> S = 20-37 ₈ D ≠ 10,20-37 ₈	0	S	L	D	Move right rotated IV bus (source) data specified by S to register specified by D. L specifies the length of source data with most significant bits set to zero.	SC = 0 WC = 0 LB/RB = 0 if S = 20-27 LB/RB = 1 if S = 30-37	0 0 0 if S = 20-27 1 if S = 30-37	1 if D = 07,17 0 1 if D = 17 0 if D = 07
		0	S	L	D					
Register to IV Bus: 0 23 78 10 11 15 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 25%;">0</td> <td style="width: 25%;">S</td> <td style="width: 25%;">L</td> <td style="width: 25%;">D</td> </tr> </table> S ≠ 07,17,20-37 ₈ D = 20-37 ₈	0	S	L	D	Move contents of register specified by S to the IV bus. Before placement on IV bus, data is shifted as specified by D, and L bits merged with destination IV bus data.	SC = 0 WC = 0 LB/RB = 0 if D = 20-27 LB/RB = 1 if D = 30-37	0 0 0 if D = 20-27 1 if D = 30-37	0 1 0 if D = 20-27 1 if D = 30-37		
0	S	L	D							
IV Bus to IV Bus: 0 23 78 10 11 15 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 25%;">0</td> <td style="width: 25%;">S</td> <td style="width: 25%;">L</td> <td style="width: 25%;">D</td> </tr> </table> S = 20-37 ₈ D = 20-37 ₈	0	S	L	D	Move right rotated IV bus data (sources) specified by S to the IV bus. Before placement on IV bus, data is shifted or specified by D and merged with original source data. L specifies the length of source data to be operated on.	SC = 0 WC = 0 LB/RB = 0 if S = 20-27 LB/RB = 1 if S = 30-37	0 0 0 if S = 20-27 1 if S = 30-37	0 1 0 if D = 20-27 1 if D = 30-37		
0	S	L	D							
ADD	1	SAME AS MOVE	(S) plus (AUX) → D Same as MOVE but contents of AUX added to the source data. If carry from most significant bit then OVF = 1, otherwise OVF = 0		SAME AS MOVE					
AND	2	SAME AS MOVE	(S) ^ (AUX) → D Same as MOVE but contents of AUX ANDed with source data.		SAME AS MOVE					
XOR	3	SAME AS MOVE	(S) ⊕ (AUX) → D Same as MOVE but contents of AUX exclusive ORed with source data.		SAME AS MOVE					
XEC	4	Register Immediate: 0 23 78 15 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 25%;">4</td> <td style="width: 25%;">S</td> <td style="width: 25%;">I</td> <td style="width: 25%;"></td> </tr> </table> S ≠ 07,17,20-37 ₈ I = 000-377 ₈	4	S	I		Execute instruction at current page address offset by I + (S).	SC = 0 WC = 0 LB/RB = x	0 0 x	0 0 x
		4	S	I						
IV Bus Immediate: 0 23 78 10 11 15 <table border="1" style="width: 100%; text-align: center;"> <tr> <td style="width: 25%;">4</td> <td style="width: 25%;">S</td> <td style="width: 25%;">L</td> <td style="width: 25%;">I</td> </tr> </table> S = 20-37 ₈ I = 00-37 ₈	4	S	L	I	Execute the instruction at the address determined by concatenating 5 high order bits of PC with the 8 bit sum of I and register specified by S. PC is not incremented.	SC = 0 WC = 0 LB/RB = 0 if S = 20-27 LB/RB = 1 if S = 30-37	0 0 0 if S = 20-27 1 if S = 30-37	0 0 x x		
4	S	L	I							

Table 2 INSTRUCTION SET SUMMARY

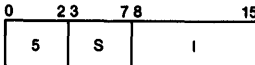
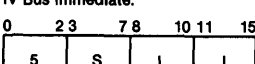
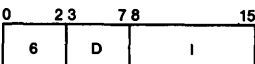
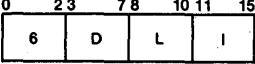
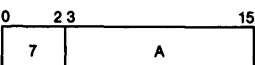
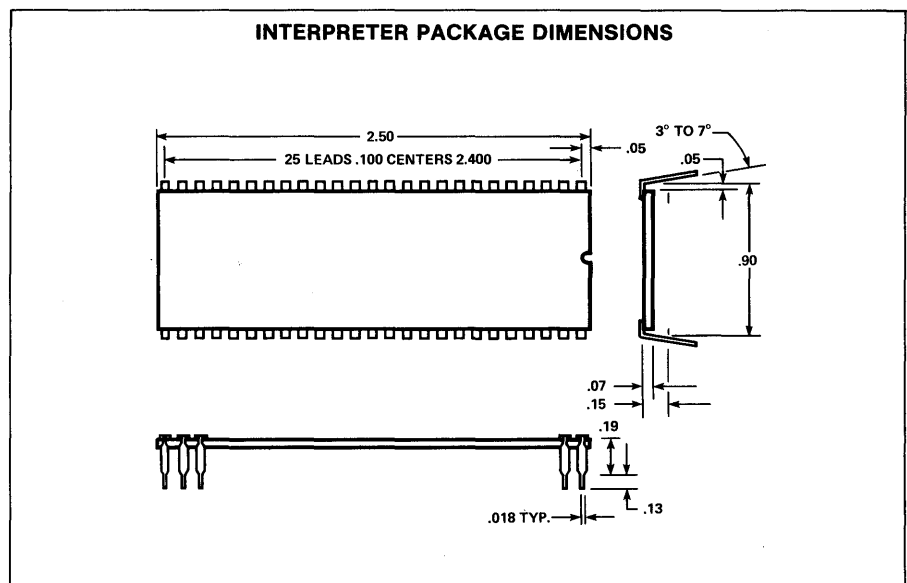
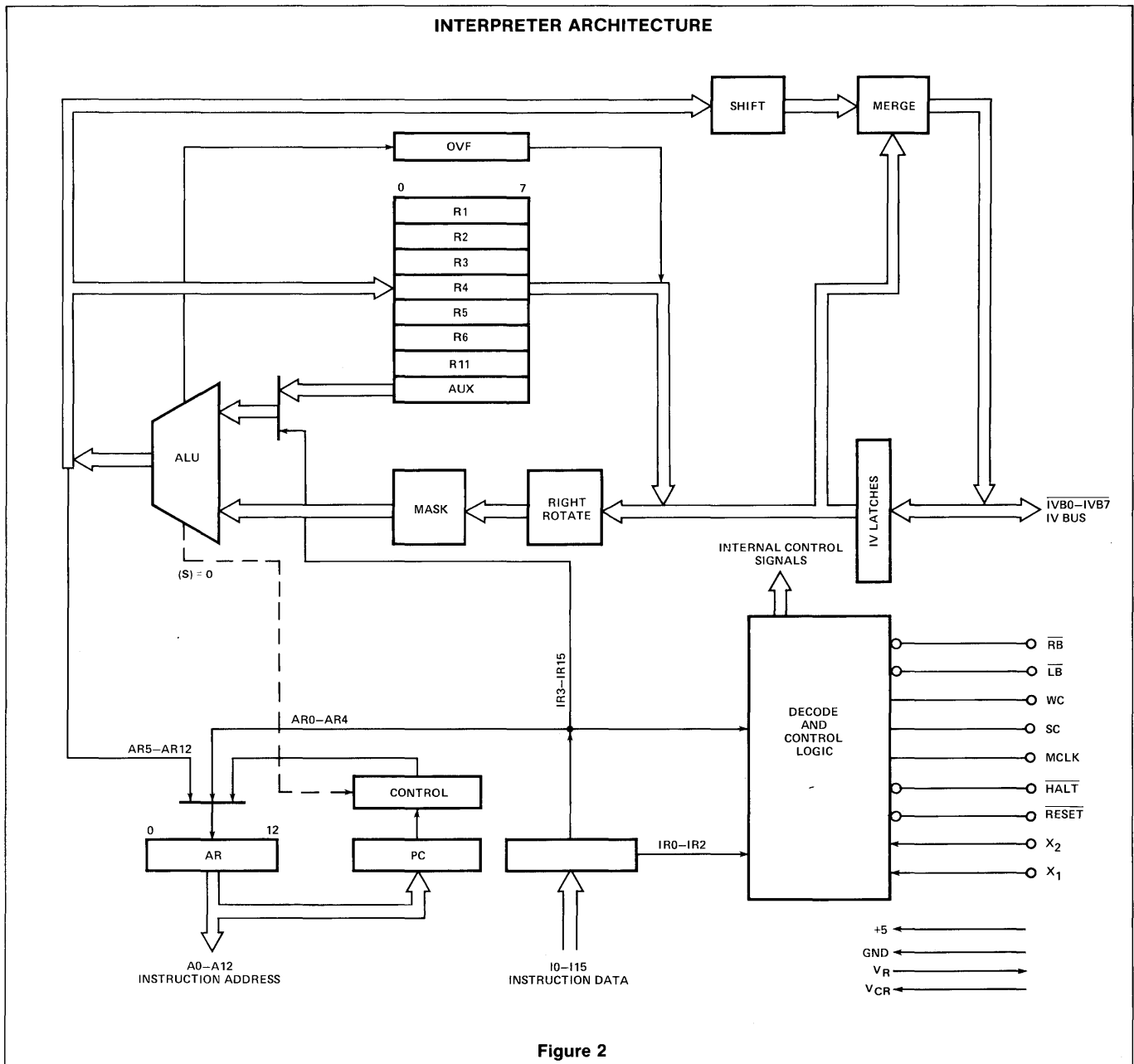
INSTRUCTION MNEMONIC	OP CODE	FORMATS	DESCRIPTION	I/O CONTROL SIGNALS	- INSTRUCTION CYCLE -	
					Instruction Input and Data Processing	Address/IV Bus Output
NZT	5	Register Immediate:  S = 07,17,20-37 ₈ I = 000-377 ₈	If (S) = 0, jump to current page address offset by I; if S = 0, PC + 1 → PC If contents of register specified by S is non zero then transfer to address determined by concatenating 5 high order bits of PC with I; if contents of register specified by S is zero, increment PC.	SC = 0 WC = 0 $\overline{\text{LB}}/\text{RB} = \text{x}$	0 0 x	0 0 x
		IV Bus Immediate:  S = 20-37 ₈ I = 00-37 ₈				
XMIT	6	Register Immediate:  D ≠ 10, 20-37 ₈ I = 000-377 ₈	Transmit I → D Transmit and store 8 bit binary pattern I to register specified by D.	SC = 0 WC = 0 $\overline{\text{LB}}/\text{RB} = \text{x}$ $\overline{\text{LB}}/\text{RD} = \text{x}$	0 0 x x	1 if D = 07,17 0 1 if D = 17 0 if D = 07
		IV BUS IMMEDIATE  D = 20-37 ₈ I = 00-37 ₈				
JMP	7	Address Immediate:  A = 00000-17777 ₈	Jump to Program Address A Jump to program storage address A. A is stored in the address register (AR).	SC = 0 WC = 0 $\overline{\text{LB}}/\text{RB} = \text{x}$	0 0 x	0 0 x

Table 2 INSTRUCTION SET SUMMARY (Cont'd)

NOTE
 1. RB is complement of LB.
 2. "0" = Low voltage
 "1" = High voltage
 x = Don't care





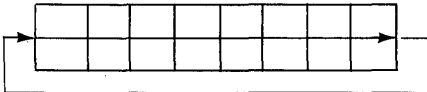

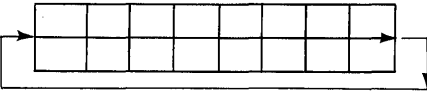
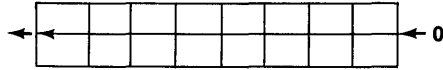
S AND/OR D FIELD SPECIFICATION (OCTAL)	SOURCE/DESTINATION
00 01 to 06 07 10 11 17	Auxiliary Register (AUX) Work registers (R1 to R6) respectively IVL write-only "register" (destination only) Overflow status (OVF)—source only Working register (R11) IVR write-only "register" (destination only)
2N (N = 0,1,2, 3,4,5,6,7)	a. If a source, IV bus data right rotated (7—N) bits and masked (specified by R/L). \overline{LB} = 'low' and \overline{RB} = 'high' generated. <div style="text-align: center;"> <p>IV Bus Source Data</p>  </div> b. If a destination, IV bus data left shifted (7—N) bits and merged (specified by $\overline{R/L}$). LB = 'low' and \overline{RB} = 'high' generated. <div style="text-align: center;"> <p>IV Bus Destination Data</p>  </div>
3N (N = 0,1,2, 3,4,5,6,7)	a. If a source, IV bus data right rotated (7—N) bits and masked (specified by R/L). \overline{LB} = 'high' and \overline{RB} = 'low' generated. <div style="text-align: center;"> <p>IV Bus Source Data</p>  </div> b. If a destination, IV bus data left shifted (7—N) bits and merged (specified by R/L). \overline{LB} = 'high' and \overline{RB} = 'low' generated. <div style="text-align: center;"> <p>IV Bus Destination Data</p>  </div>

Table 3 DATA SOURCE DESTINATION ADDRESS

INTERPRETER INTERNAL REGISTERS
Programmable Registers (all 8 bits):
AUX — General working register. Contains second term for arithmetic or logical operations.
R1 — General working register
R2 — General working register
R3 — General working register
R4 — General working register
R5 — General working register
R6 — General working register
R11 — General working register
Other Registers:
Address Register (AR)
— A 13-bit register containing the address of the current instruction.
OVF — The least-significant bit of this register is used to reflect overflow status resulting from the most recent ADD operation (see Instruction Set Summary).
Program Counter (PC)
— Normally contains the address of the current instruction and is incremented to obtain the next instruction address.
Instruction Register (IR)
— Holds the 16-bit instruction word currently being executed.

Table 4

SYSTEM DESIGN USING THE INTERPRETER

Designing hardware around the 8X300 Interpreter reduces to selecting a program storage device (ROM, PROM, etc.), selecting I/O devices (IV byte, multiplexers, RAM, etc.), selecting clock mode (system driven or crystal controlled) and interfacing the Interpreter to these components, as shown in Figure 3.

System Clock

The Interpreter has an integrated oscillator which generates all necessary clock signals. The oscillator is designed to connect directly to a series resonant quartz crystal via pins X1 and X2. The crystal resonant frequency, f , is related to the desired cycle time, T , by the relationship $f = 2/T$. For a 250ns system, $f = 8.00\text{MHz}$.

In lower speed applications where the cycle time need not be precisely controlled, a

capacitor may be connected between X1 and X2 to drive the oscillator. Approximate capacitor values are given in Table 6. If cycle time is to be varied, X1 and X2 should be driven from complementary outputs of a pulse generator. Figure 4 shows a typical configuration. For systems where the Interpreter is to be driven from a master clock, the X1 and X2 lines may be interfaced to TTL logic as shown in Figure 5.

Type:	Fundamental mode, series resonant
Impedance at Fundamental:	35 ohms maximum
Impedance at harmonics and spurs:	50 ohms minimum

Table 5 CRYSTAL CHARACTERISTICS

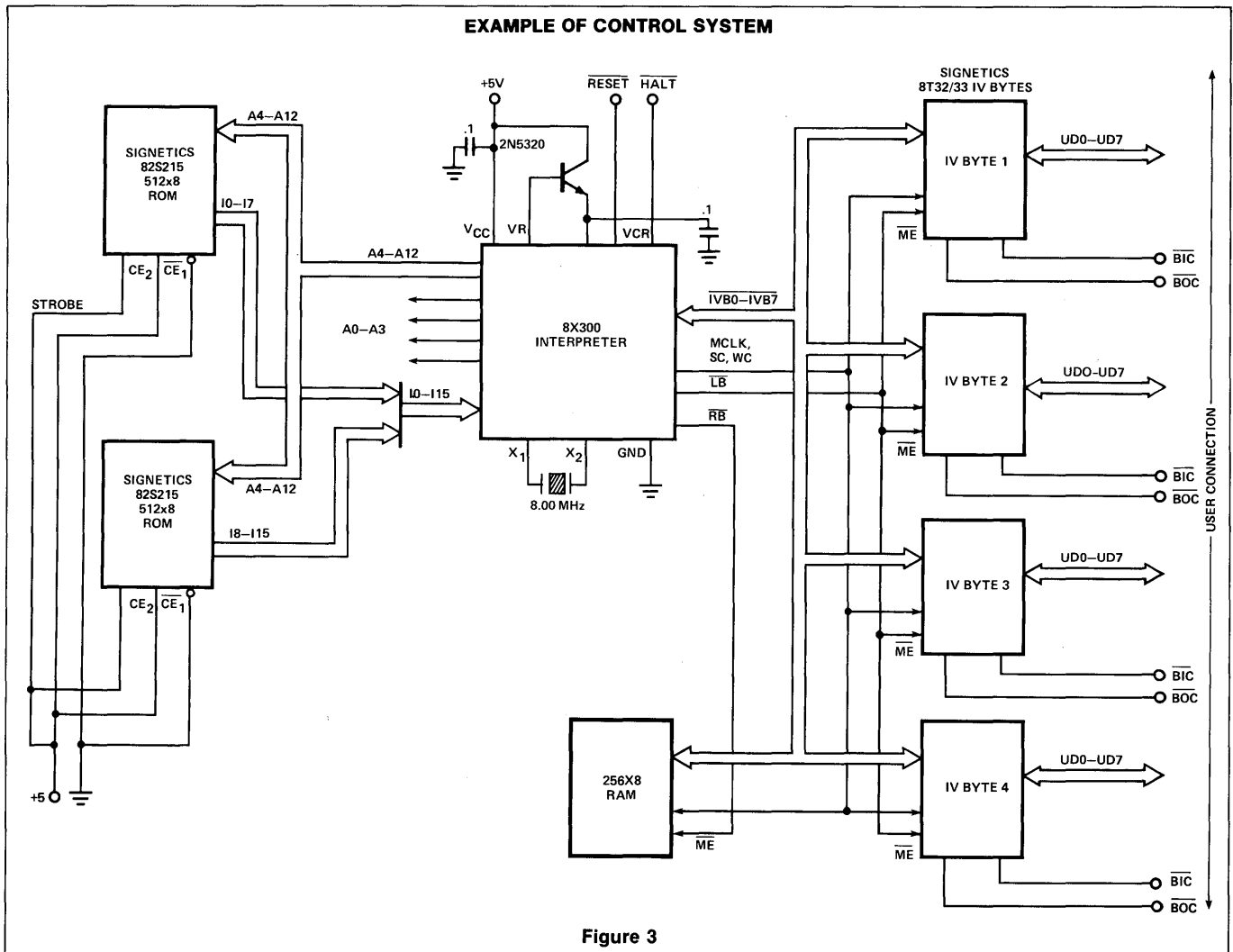


Figure 3

Cx,pF	CYCLE TIME
100	300ns
200	500ns
500	1.1μs
1000	2.0μs

Table 6 CLOCK CAPACITOR VALUES

Halt, Reset Signals

HALT:

A low level at the $\overline{\text{HALT}}$ input stops internal operation of the Interpreter at the start of the next instruction after $\overline{\text{HALT}}$ is applied (quarter cycle after MCLK). Since $\overline{\text{HALT}}$ is sampled at the start of each instruction cycle it is possible to prevent a cycle by applying $\overline{\text{HALT}}$ early in that cycle. $\overline{\text{HALT}}$ does not inhibit MCLK or affect any internal registers. Normal operation begins with the next complete cycle after the $\overline{\text{HALT}}$ input goes high.

RESET:

A low level at the $\overline{\text{RESET}}$ input sets the program counter and address register to zero. While $\overline{\text{RESET}}$ is low MCLK is inhibited. If $\overline{\text{RESET}}$ is applied during the last 2 quarter cycles, the MCLK during that cycle may be shortened. $\overline{\text{RESET}}$ should be applied for 1 full instruction cycle time to assure proper operation. When $\overline{\text{RESET}}$ input goes high an MCLK occurs prior to the resumption of normal processing. $\overline{\text{RESET}}$ does not affect the other internal registers.

EXAMPLE:

A specific example of a control system, using the 8X300 Interpreter—four 8T32/33 IV Bytes, and two 82S215 ROMs is shown in Figure 3. Only 8 components are required to build this system which contains 512 words of program storage, 32 TTL I/O connection points, and operates at a 250ns instruction cycle time.

SYSTEM TIMING

In systems with fast instruction cycle times, most Interpreter delays are strictly determined by internal gate propagation delays. Since some events are constrained to occur in certain quarter cycles, as system cycle times become slower, the delays will appear to increase due to gating with internal clocks. In the table of AC Electrical Characteristics, 2 columns are used: 1 to denote times which occur due to internal clock intervention and 1 to denote minimum delays for fast cycle times.

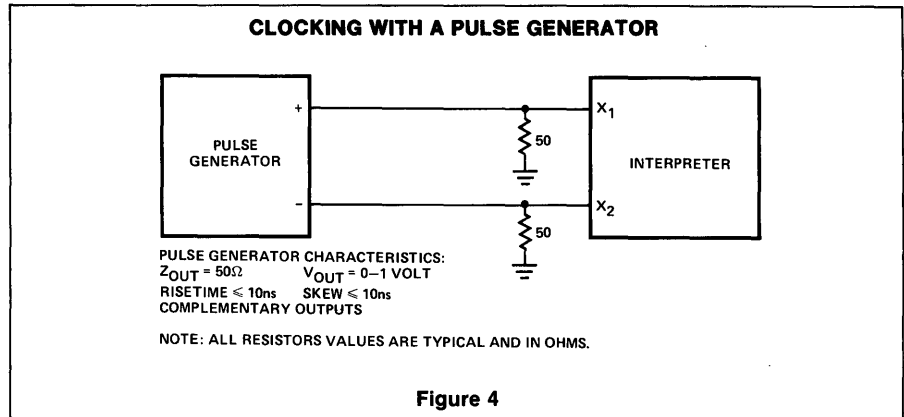


Figure 4

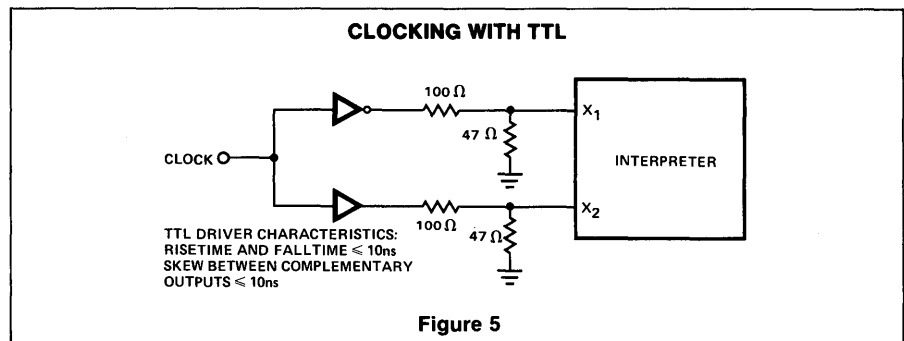


Figure 5

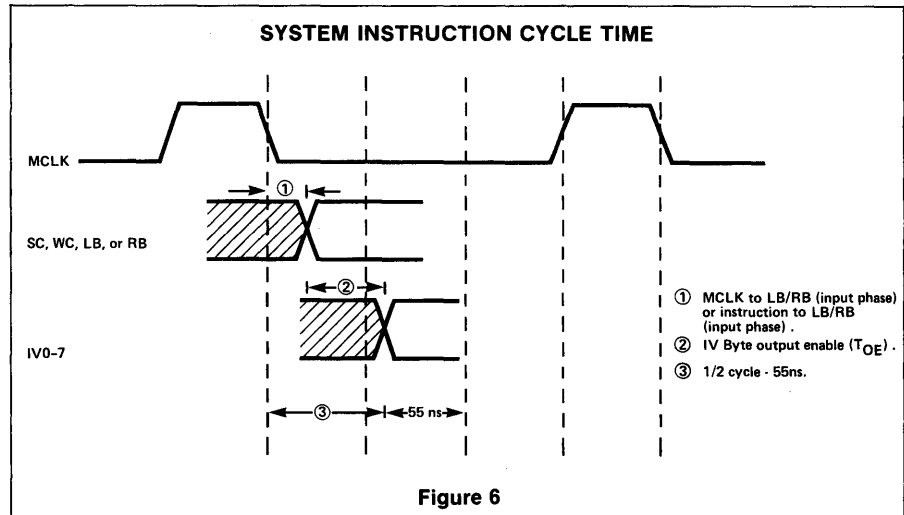


Figure 6

When using Signetics 8T32/33/35/36 IV Bytes, selection of instruction cycle time involves calculating the maximum program storage access time. Assuming the instruction is available when MCLK falls, the I/O control lines are stable 35ns later. Signetics IV Bytes require another 35ns to disable a previously selected byte and enable the desired byte (assumes a change in bank signals). A 10ns margin has been added to the IV Byte enable for this evaluation to reflect the fact that most systems will have more capacitive loading than the 50pF test

condition in the IV Byte specification and to allow for line delays.

The system instruction cycle time for normal systems such as shown in Figure 7 is determined by Interpreter propagation delays, program storage access time, and IV Byte output enable times. Instruction cycle time is normally constrained by one or more of the following conditions:

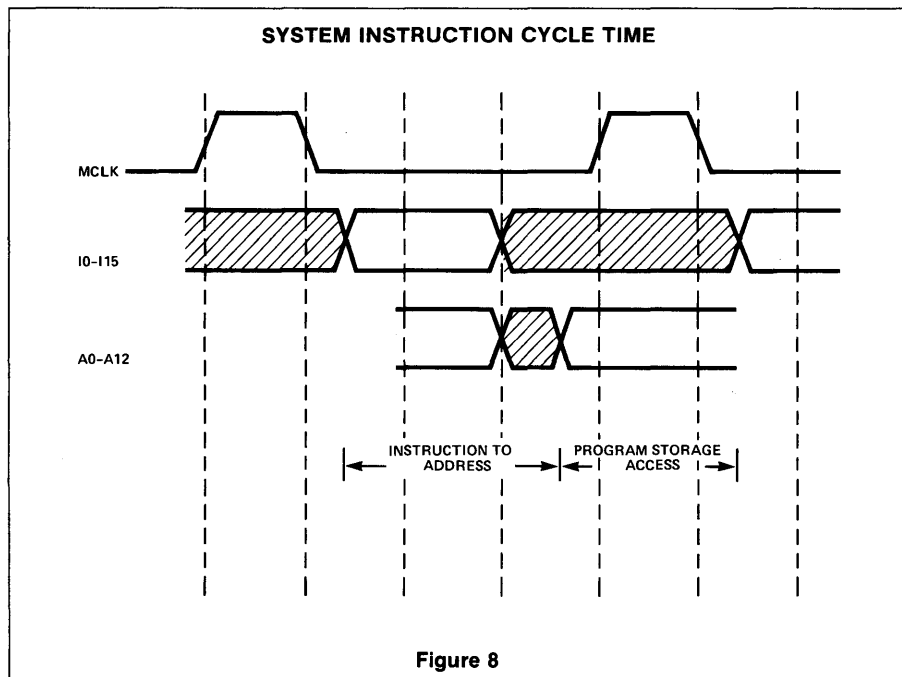
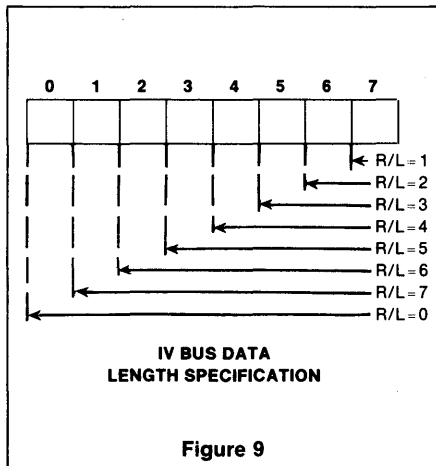
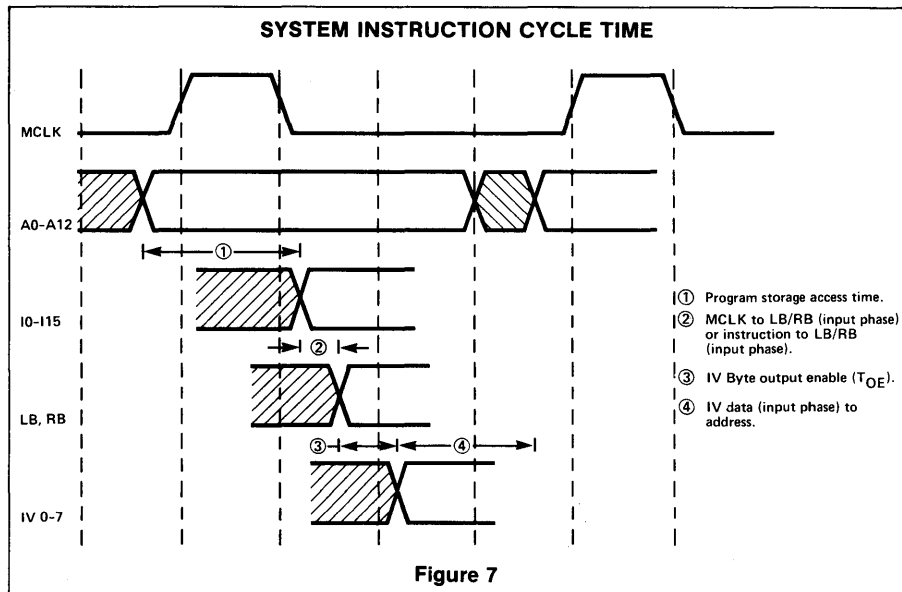
1. Instruction to LB/RB (input phase) and IV Byte output enable:
TOE ≤ 1/2 cycle - 55ns (Figure 6).

2. Program storage access time and instruction to LB/RB (input phase) and IV Byte output enable and IV data (input phase) to address \leq instruction cycle time (Figure 7).
3. Program storage access time and instruction to address \leq instruction cycle time (Figure 8).

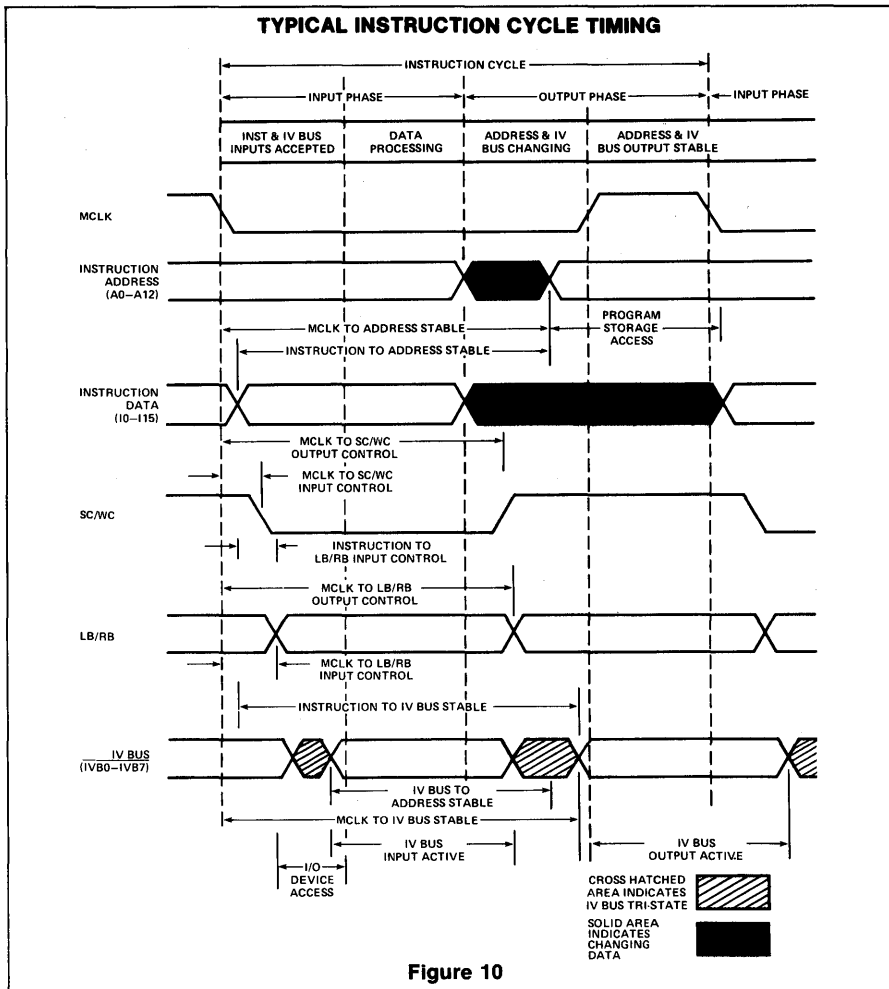
The first constraint can be used to determine the minimum cycle time. Using the inequality $35ns + 35ns \leq \frac{1}{2} \text{ cycle} - 55ns$ implies $\frac{1}{2} \text{ cycle} \geq 125ns$ or an instruction time of 250ns.

Program storage access time for a 250ns instruction cycle can be calculated from the second constraint. Noting that the specification for IV data (input phase) to address is 115ns: Program storage access time + 35ns + 35ns + 115ns \leq 250ns implies program storage access time \leq 65ns.

The third constraint can be used to verify the maximum program storage access time. Noting that the specification for instruction to address is 185ns: Program storage access time + 185ns \leq 250ns confirms that program storage access time 65ns is satisfactory.



ABSOLUTE MAXIMUM RATINGS
 Supply Voltage V_{CC} 7V
 Logic Input Voltage 5.5V
 Crystal Input Voltage 2V



AC ELECTRICAL CHARACTERISTICS $V_{CC} = 5V \pm 5\%$ and $0^{\circ}C \leq T_A < 70^{\circ}C$

DELAY DESCRIPTION	PROPAGATION DELAY TIME	CYCLE TIME LIMIT
X1 falling edge to MCLK (driven from external pulse generator)	75ns	
MCLK to SC/WC falling edge (input phase)	25ns	
MCLK to SC/WC rising edge (output phase)		$\frac{1}{2}$ cycle + 25ns
MCLK to LB/RB (input phase)	35ns	
Instruction to LB/RB output (input phase)	35ns	
MCLK to LB/RB (output phase)		$\frac{1}{4}$ cycle + 35ns
MCLK to IV data (output phase)	185ns	$\frac{1}{2}$ cycle + 60ns
IV data (input phase) to IV data (output phase)	115ns	
Instruction to Address	185ns	$\frac{1}{2}$ cycle + 40ns
MCLK to Address	185ns	$\frac{1}{2}$ cycle + 40ns
IV data (input phase) to Address	115ns	
MCLK to IV data (input phase)		$\frac{1}{2}$ cycle - 55ns
MCLK to Halt falling edge to prevent current cycle		$\frac{1}{4}$ cycle - 40ns
Reset rising edge to first MCLK		0 to 1 cycle

NOTE
 1. Reference to MCLK is to the falling edge when loaded with 300pF.
 2. Loading on Address lines is 150pF.

DC ELECTRICAL CHARACTERISTICS

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Typ	Max	
V _{IH} High level input voltage X1,X2 All others		.6			V
		2			V
V _{IL} Low level input voltage X1,X2 All others				.4	V
				.8	V
V _{CL} Input clamp voltage (Note 1)	V _{CC} = 4.75V I _I = -10mA			-1.5	V
I _{IH} High level input current X1,X2 All others	V _{CC} = 5.25V V _{IH} = .6V		2700		μA
	V _{CC} = 5.25V V _{IH} = 4.5V		<1	50	μA
I _{IL} Low level input current X1,X2 IVBO-7 IO-I15 HALT, RESET	V _{CC} = 5.25V V _{IL} = .4V		-2500		μA
	V _{CC} = 5.25V V _{IL} = .4V		-140	-200	μA
	V _{CC} = 5.25V V _{IL} = .4V		-880	-1600	μA
	V _{CC} = 5.25V V _{IL} = .4V		-230	-400	μA
V _{OL} Low level output voltage A0-A12 All others	V _{CC} = 4.75V I _{OL} = 4.25mA		.35	.55	V
	V _{CC} = 4.75V I _{OL} = 16mA		.35	.55	V
V _{OH} High level output voltage	V _{CC} = 4.75V I _{OH} = 3mA	2.4			V
I _{OS} Short circuit output current (Note 2)	V _{CC} = 5.25V	-30		-140	mA
V _{CC} Supply voltage		4.75	5	5.25	V
I _{CC} Supply current	V _{CC} = 5.25V		300	450	mA
I _{REG} Regulator control	V _{CC} = 5.0V	-14		-21	mA
I _{CR} Regulator current (Note 3)	V _{CR} = 0			290	mA
V _{CR} Regulator voltage (Note 3)	V _{REG} = 0V	2.2		3.2	V

NOTES

- Crystal inputs X1 and X2 do not have clamp diodes.
- Only one output may be grounded at a time.
- (Limits apply for V_{CC} = 5V ± 5% and 0°C < T_A < 70°C unless specified otherwise.)

INTERFACE COMPONENTS

DC ELECTRICAL CHARACTERISTICS

PARAMETER	INPUT VOLTAGE												OUTPUT VOLTAGE								
	V _{IL} (V) LOW LEVEL			V _{IH} (V) HIGH LEVEL			V _{IC} CLAMP VOLTAGE			VOLTAGE RATING			V _{TL} (mV) ²⁰ LOW LEVEL THRESHOLD VOLTAGE			V _{TH} (mV) ²⁰ HIGH LEVEL THRESHOLD VOLTAGE			V _{OL} (V) ⁷ LOW LEVEL		
TEST CONDITIONS							V _{CC} = MIN I _{IN} = -12mA			V _{IN} = 10mA			V _{CC} = MIN V _{IN} = 0.8V I _{OL} = -400μA			V _{CC} = MAX V _{IN} = 0.8V I _{OH} = 16mA			V _{CC} = MIN		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
8T26A		N/A			N/A				-1.0		N/A		0.85					2			Driver I _{OL} = 48mA 0.5 Receiver I _{OL} = 20mA 0.5
8T28		N/A			N/A				-1.0		N/A		0.85					2			Driver I _{OL} = 48mA 0.5 Receiver I _{OL} = 20mA 0.5
8T31		N/A			N/A		I _{IN} = -5mA		-1		N/A			N/A			N/A				I _{OL} = 20mA 0.55
8T32 ¹⁶			.8	2.0			I _{IN} = -5mA		-1		N/A			N/A			N/A				I _{OL} = 16mA 0.55
8T33 ¹⁶			.8	2.0			I _{IN} = -5mA		-1		N/A			N/A			N/A				I _{OL} = 16mA 0.55
8T35 ¹⁶		.8	2.0				I _{IN} = -5mA		-1		N/A			N/A			N/A				I _{OL} = 16mA 0.55
8T36 ¹⁶			.8	2.0			I _{IN} = -5mA		-1		N/A			N/A			N/A				I _{OL} = 16mA 0.55

INTERFACE COMPONENTS

DC ELECTRICAL CHARACTERISTICS (Cont'd)

PARAMETER	INPUT CURRENT						OUTPUT CURRENT						POWER SUPPLY										
	V_{OH} (V) HIGH LEVEL			I_{IL} (mA) LOW LEVEL			I_{IH} (μ A) HIGH LEVEL			I_{CBO} (μ A) LEAKAGE CURRENT			I_{OS} SHORT CIRCUIT CURRENT			I_{CC} POWER/CURRENT CONSUMPTION (mW/mA)				I_{CC} (mA) $V_{IN} = 2.0V$			
	$V_{CC} = \text{MIN}$ $I_{OH} = -160\mu A$			$V_{CC} = \text{MAX}$ $V_{IN} = 0.4V$			$V_{CC} = \text{MAX}$ $V_{IN} = 4.5V$			$V_{IN} = 2.0V$			$V_{CC} = \text{MAX}$ $V_{IN} = 0V$ $V_{OUT} = 0V$			$V_{CC} = \text{MAX}$ $V_{IN} = 0V$ MILITARY COMMERCIAL				$V_{CC} = \text{MAX}$			
TEST CONDITIONS	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Typ	Max	Typ	Max	Min	Typ	Max	
8T26A	Driver $I_{OL} = -10mA$ 2.4 Receiver $I_{OL} = -100\mu A$ 3.5 $I_{OL} = -2.0mA$ 2.4	Driver LOW Level -200 LOW Level (Disabled) -25 Receiver -200	Driver/Receiver 25	HIGH Level $V_{OUT} = 2.4V$ 100 LOW Level $V_{OUT} = 0.5V$ -100 HIGH Level $V_{OUT} = 2.4V$ -50	50 -30	Driver -150 Receiver -75 Driver -150	N/A 457/87 N/A 578/100																
8T28	Driver $I_{OL} = -10mA$ 2.4 Receiver $I_{OL} = -100\mu A$ 3.5 $I_{OL} = -2.0mA$ 2.4	Driver LOW Level -200 LOW Level (Disabled) -25 Receiver -200	Driver/Receiver 25	LOW Level $V_{OUT} = 0.5V$ -100 -10 N/A 10	100 -30 -20	Receiver -75 -200 UD Bus	N/A N/A N/A																150 150
8T31	$I_{OL} = 3.2mA$ 2.4	$V_{IN} = 0.55V$ -500	$V_{IN} = 5.5V$ 100			IV Bus 20																	
8T32 ¹⁶	$I_{OH} = 3.2mA$ 2.4	$V_{IL} = .5V^{17}$ -350 -550	$V_{IH} = 5.25V^{17}$ <10 100	N/A		UD Bus 10 IV Bus 20 UD Bus 10 IV Bus 20	N/A N/A																100 150
8T33 ¹⁶	$I_{OH} = 3.2mA$ 2.4	$V_{IL} = .5V^{17}$ -350 -550	$V_{IH} = 5.25V^{17}$ <10 100	N/A		UD Bus 10 IV Bus 20	N/A N/A																100 150
8T35 ¹⁶	$I_{OH} = -3.2mA$ 2.4	$V_{IL} = .5V^{17}$ -350 -550	$V_{IH} = 5.25V^{17}$ <10 100	N/A		UD Bus 10 IV Bus 20	N/A N/A																100 150
8T36 ¹⁶	$I_{OH} = -3.2mA$ 2.4	$V_{IL} = .5V^{17}$ -350 -500	$V_{IH} = 5.25V^7$ <10 100	N/A																			

NOTES

- All voltage measurements are referenced to the ground terminal. Terminals not specifically referenced are left electrically open.
- All measurements are taken with ground pin tied to zero volts.
- Positive current is defined as into the terminal referenced.
- Precautionary measures should be taken to insure current limiting in accordance with absolute maximum ratings.
- Measurements apply to each gate element independently.
- Output source current is supplied through a resistor to ground.
- Output sink current is supplied through a resistor to V_{CC} .
- Connect an external 1K 1% resistor to the output for this test.
- Not more than one output should be shorted at one time.
- Previous condition is a high level output state.
- Previous condition is a low level output state.
- Test each driver separately.
- For more electrical specifications see data sheet.
- I_{CC} is dependent upon loading. I_{CC} limit specified is for no-load test condition for both drivers.
- With forced output current of $240\mu A$, the output voltage must not exceed 0.15V.
- These limits do not apply during address programming.
- The input current includes the tri-state/open collector leakage current of the output driver on the data lines.
- Output leakage current is supplied through a 2K Ω resistor to 30V.
- Output sink current is supplied through a resistor to 30V.
- The differential input threshold voltage is defined as the maximum dc voltage duration from the reference level necessary to trigger the one shot.
- Common mode voltages that are confined within the dynamic range as specified will not cause false triggering of the one-shot.
- Hysteresis is defined as voltage difference between R input level at which output begins to go from "0" to "1" state and level at which output begins to go from "1" to "0." Refer to Hysteresis test circuit.
- $V_{CC} = +12.6V$, $V_{EE} = 12.6V$.

DESCRIPTION

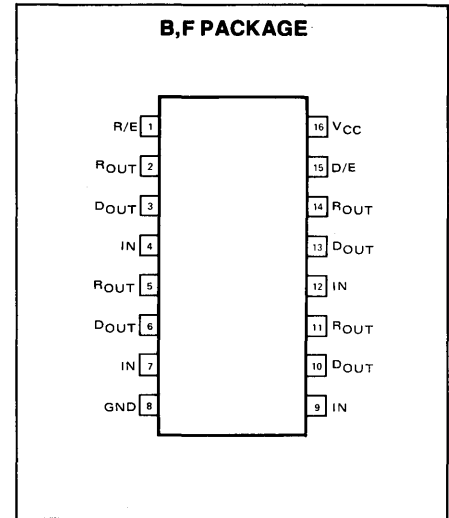
The 8T26A/28 consists of four pairs of Tri-State logic elements configured as Quad Bus Drivers/Receivers along with separate buffered receiver enable and driver enable lines. This single IC Quad Transceiver design distinguishes the 8T26A/28 from conventional multi-IC implementations. In addition, the 8T26/28's ultra high speed while driving heavy bus capacitance (300pF) makes these devices particularly suitable for memory systems and bidirectional data buses.

Both the Driver and Receiver gates have Tri-State outputs and low-current PNP inputs. Tri-State outputs provide the high switching speeds of totempole TTL circuits while offering the bus capability of open collector gates. PNP inputs reduce input loading to 200µA maximum.

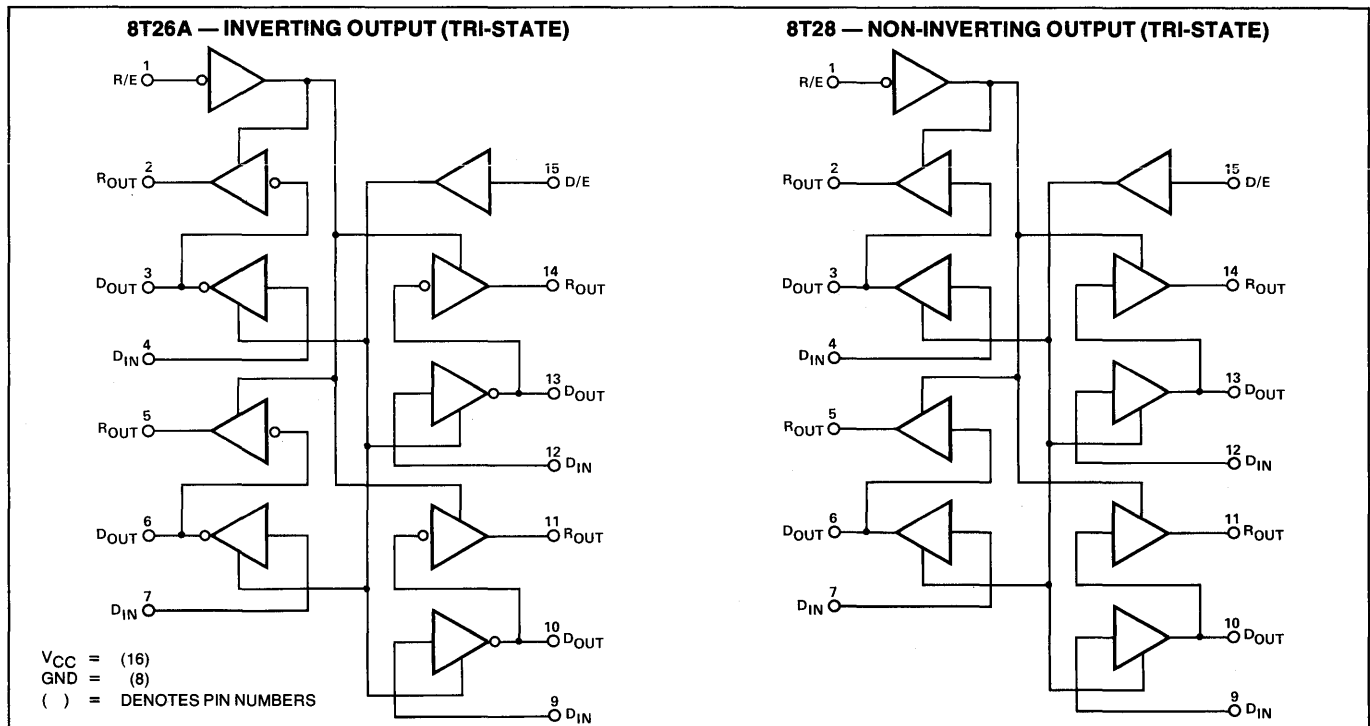
APPLICATIONS

- Half-duplex data transmission
- Memory interface buffers
- Data routing in bus oriented systems
- High current drivers
- MOS/CMOS-to-TTL interface

PIN CONFIGURATION



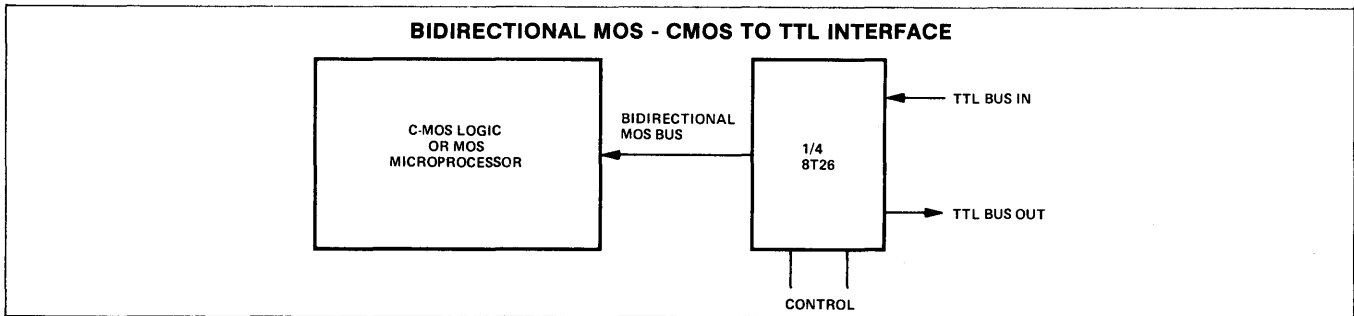
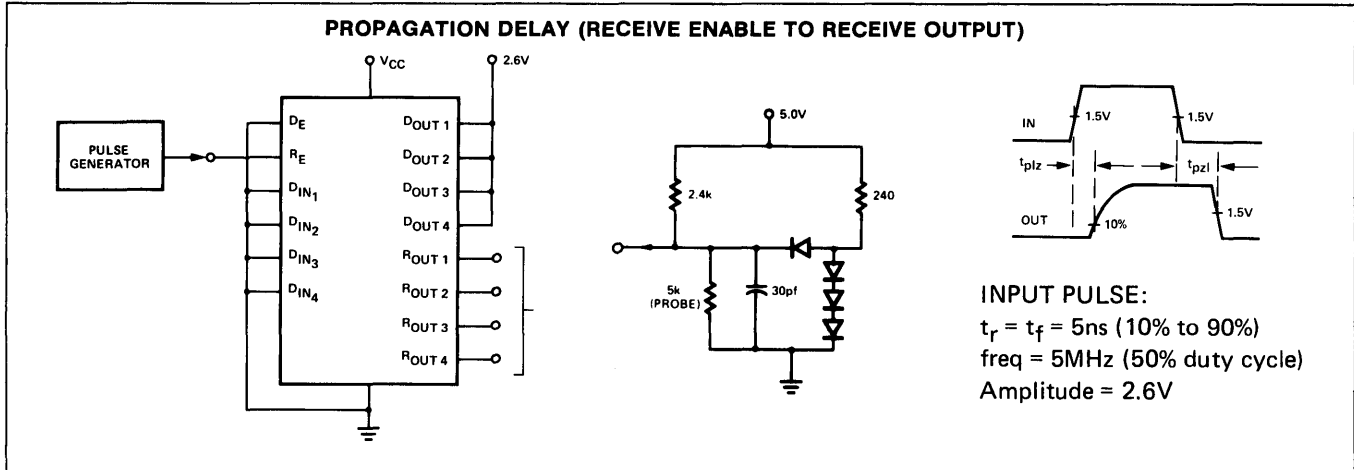
LOGIC DIAGRAM



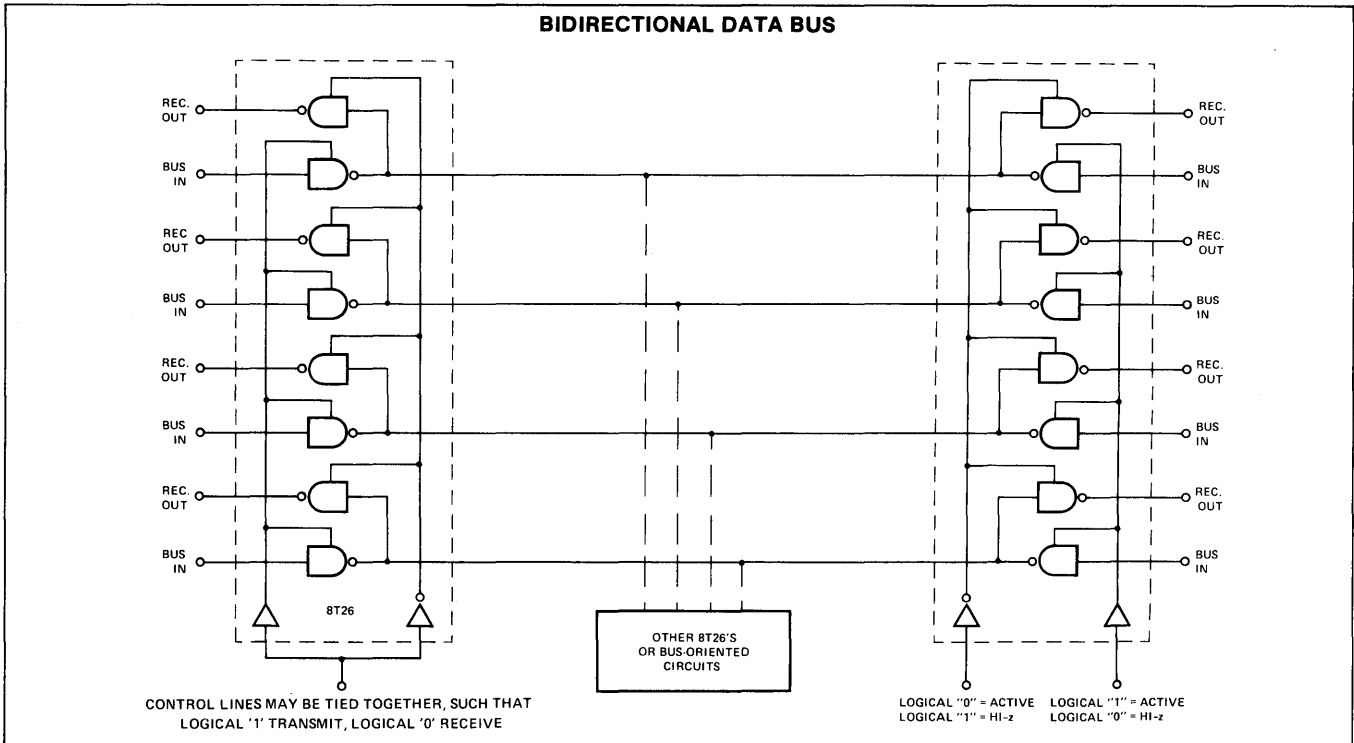
SWITCHING CHARACTERISTICS

PARAMETER	TEST CONDITIONS	8T26A	8T28	UNIT
		Max	Max	
Propagation Delay t_{ON}	DOUT to R_OUT	14	17	ns
	DOUT to R_OUT	14	17	
t_{OFF}	DOUT to R_OUT	14	17	ns
	DIN to D_OUT	14	17	
Data Enable to Data Output t_{PZL}	DIN to D_OUT	25	28	ns
	High Z to O	20	23	
Receiver Enable to Receiver Output t_{PZL}	O to High Z	20	23	ns
	High Z to O	15	18	
t_{PLZ}	O to High Z			

BLOCK DIAGRAM

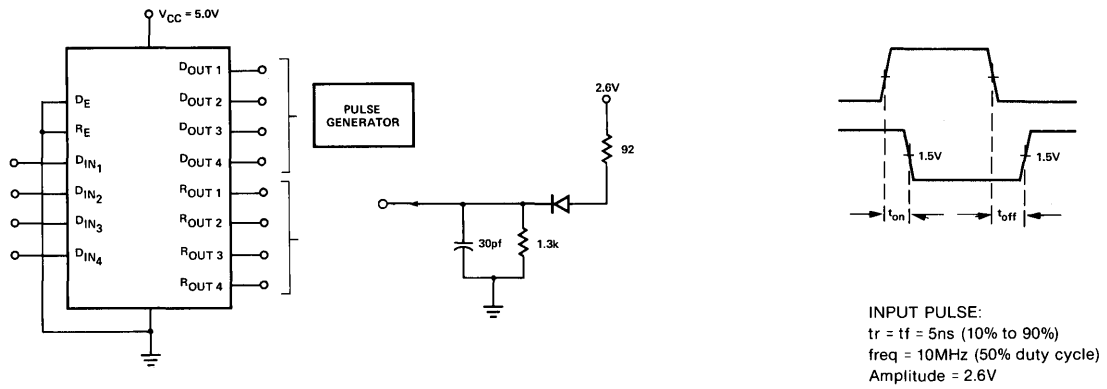


TYPICAL APPLICATIONS

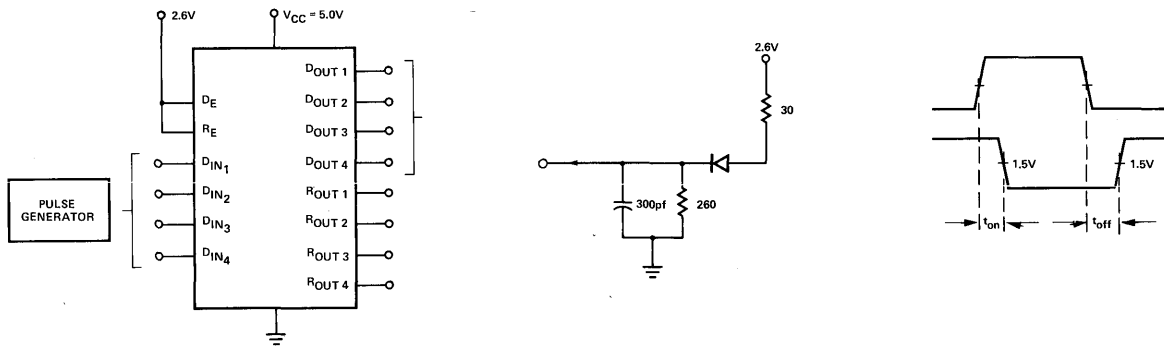


AC TEST CIRCUITS AND WAVEFORMS

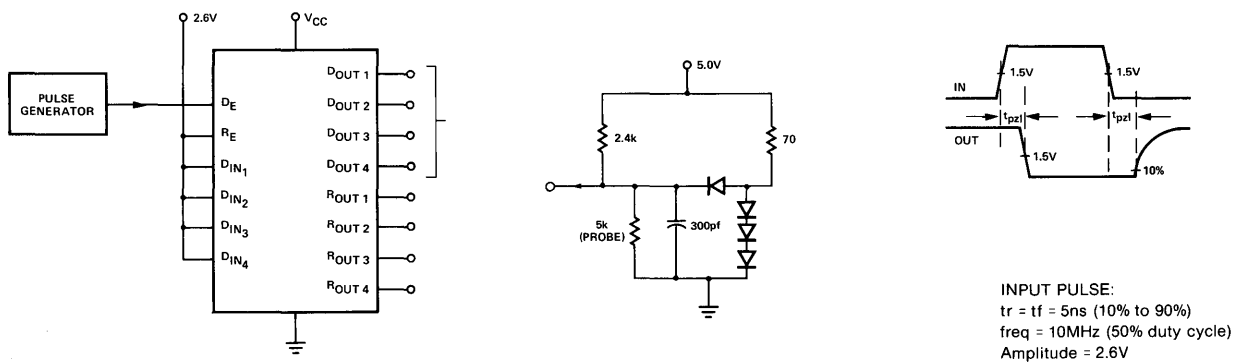
PROPAGATION DELAY (DOUT TO ROUT)



PROPAGATION DELAY (DIN TO DOUT)



PROPAGATION DELAY (DATA ENABLE TO DATA OUTPUT)



OBJECTIVE SPECIFICATION

8T31 N,F

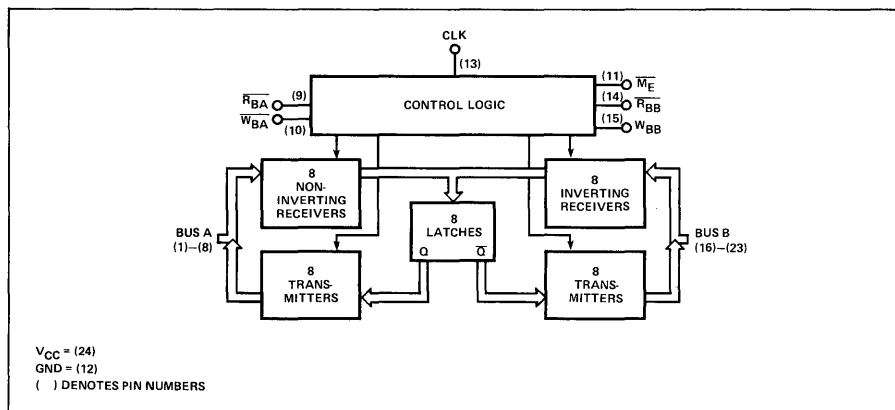
DESCRIPTION

The 8T31 8-bit Bidirectional I/O Port is designed to function as a general purpose I/O interface element in minicomputers, microcomputers and other bus oriented digital systems. It consists of 8 clocked latches with two sets of bidirectional inputs/outputs, Bus A (BA0-BA7) and Bus B (BB0-BB7). Each Bus has a write control line and a read control line. The two buses operate independently except for the case where the user is attempting to write data in from each bus simultaneously. In that case, the data on Bus A will be written into the latches while Bus B will be forced into a high impedance state. Data written into one Bus will appear inverted at the other Bus.

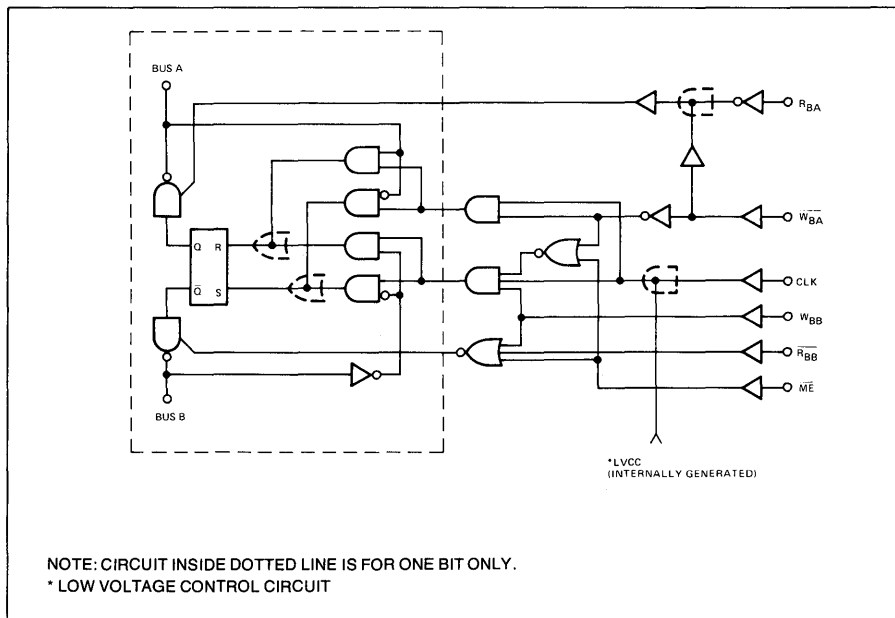
A unique feature of the 8T31 is its ability to start up in a predetermined state. If the clock is maintained at a voltage less than .8V until the power supply reaches 3.5V, Bus A will always be all logic 1 levels, while Bus B will be all logic 0 levels.

A master enable (ME) is provided that enables or disables Bus B regardless of the state of the other inputs.

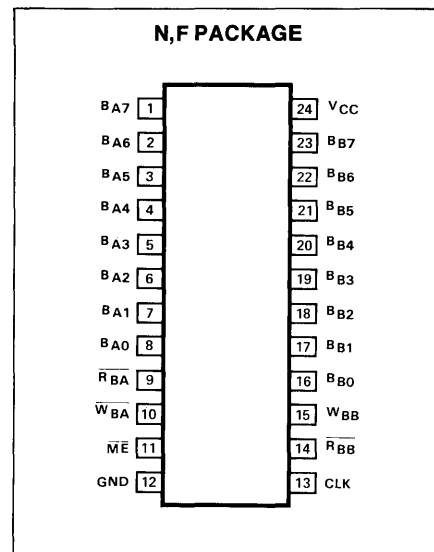
BLOCK DIAGRAM



SCHEMATIC



PIN CONFIGURATION



FUNCTION TABLE

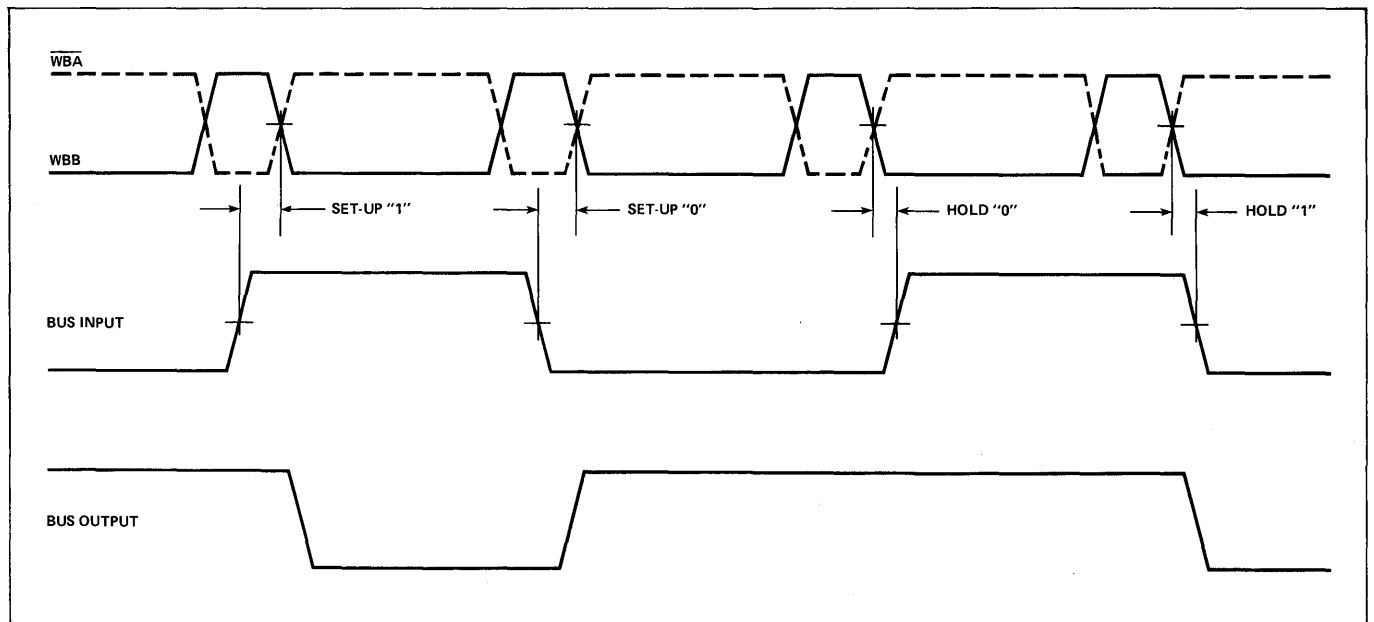
BUS A					
R _{BA}	W _{BA}	CLK			
X	0	1	WRITE (INPUT)		
0	1	X	READ (OUTPUT)		
1	1	X	HI-Z		
BUS B					
R _{BB}	W _{BB}	W _{BA}	CLK	M _E	
X	X	X	X	1	HI-Z
1	0	X	X	0	HI-Z
X	1	0	X	0	HI-Z
0	0	X	X	0	READ (OUTPUT)
X	1	1	1	0	WRITE (INPUT)

SWITCHING CHARACTERISTICS

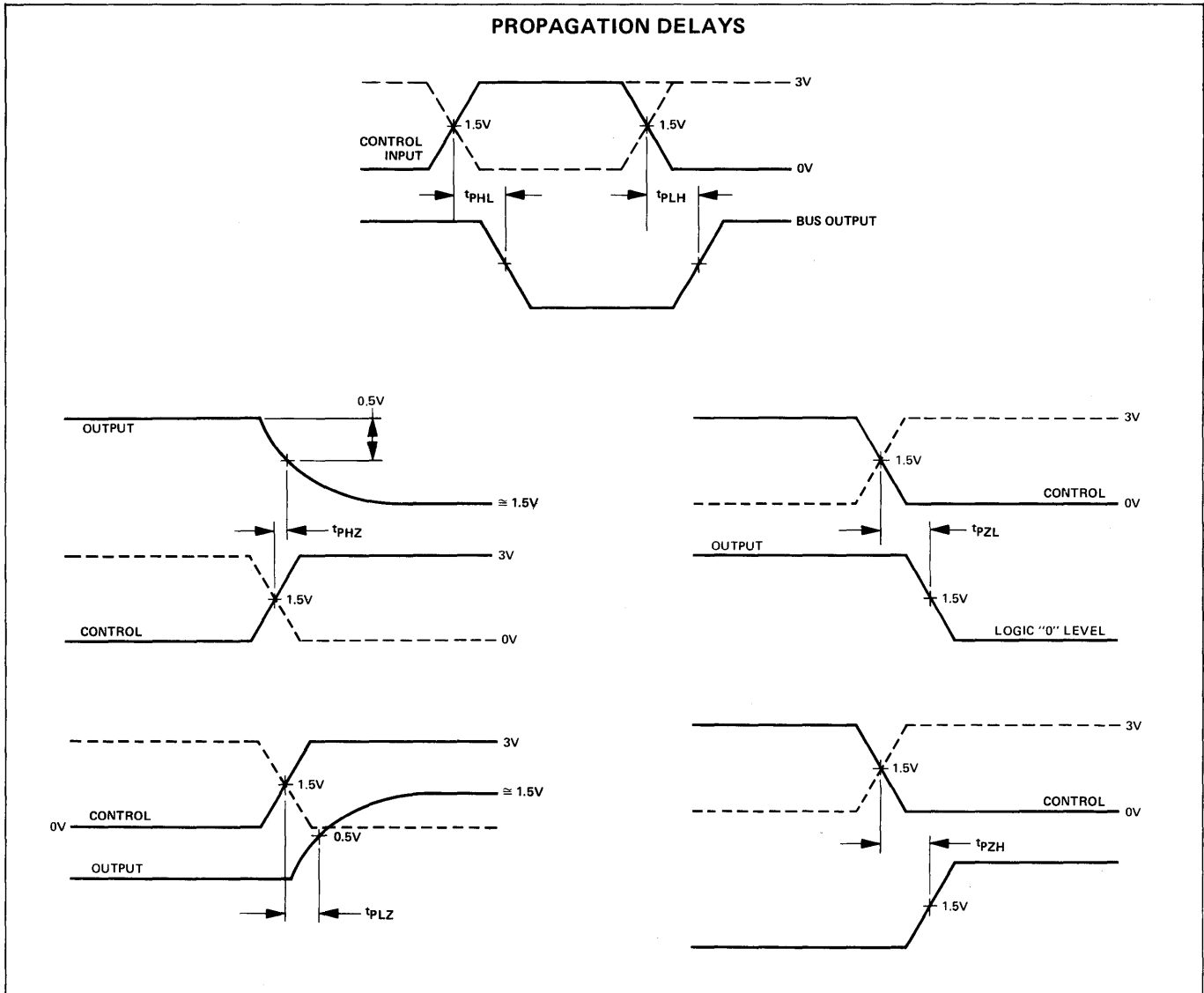
PARAMETER	TEST CONDITIONS	LIMITS			UNIT	
		Min	Typ	Max		
tZL	CL = 300pF CL = 300pF CL = 30pF CL = 30pF CL = 30pF CL = 30pF		27	45	ns	
tZH			29	50	ns	
tZL		Propagation Delay From Read (RBB), Write (WBB) and Master Enable (ME) to Bus B		17	30	ns
tZH				14	25	ns
tLZ				13	20	ns
tHZ				17	30	ns
tSETUP	Bus A Data Setup and Hold Times	0	-10		ns	
tHOLD1		10	4		ns	
tHOLD0		25	16		ns	
tSETUP	Bus A Write Setup and Hold Times	30	20		ns	
tHOLD		0	-30		ns	
tSETUP	Bus B Data Setup and Hold Times	*			ns	
tHOLD		0			ns	
CIN	Input Capacitances Control Data			6	pF	
		VIN = 0V		12	pF	
		VIN = 3V		9	pF	

*The Bus B Data Setup Time is equal to the clock pulse width.

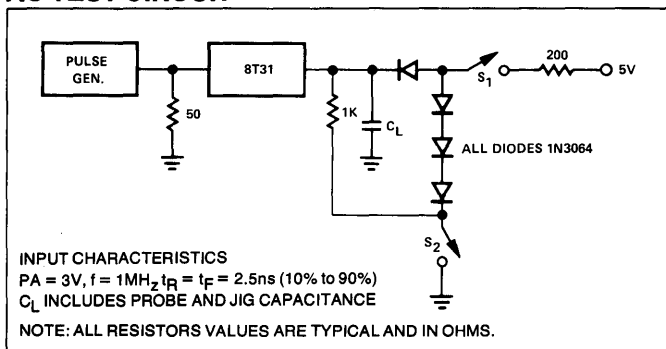
CLOCK



AC WAVEFORMS



AC TEST CIRCUIT



TEST TABLE

	S ₁	S ₂
t _{PHL}	Closed	Closed
t _{PLH}	Closed	Closed
t _{PL}	Closed	Closed
t _{PHZ}	Closed	Closed
t _{PZL}	Closed	Open
t _{PZH}	Open	Closed

TYPES

- 8T32** Tri-State, Synchronous User Port
- 8T33** Open Collector, Synchronous User Port
- 8T35** Open Collector, Asynchronous User Port
- 8T36** Tri-State, Asynchronous User Port

DESCRIPTION

The Interface Vector (IV) Byte is an 8-bit bidirectional data register designed to function as an I/O interface element in microprocessor systems. It contains 8 data latches accessible from either a microprocessor (IV) port or a user port. Separate I/O control is provided for each port. The 2 ports operate independently, except when both are attempting to input data into the IV Byte. In this case, the user port has priority.

A unique feature of the 8T32/33/35/36 IV Byte is the way in which it is addressed. Each IV Byte has an 8-bit, field programmable address, which is used to enable the microprocessor port. When the SC control signal is high, data at the microprocessor port is treated as an address. If the address matches the IV Byte's internally programmed address, the microprocessor port is enabled, allowing data transfer through it.

The port remains enabled until an address which does not match is presented, at which time the port is disabled (data transfer is inhibited). A Master Enable input (ME) can serve as a ninth address bit, allowing 512 IV Bytes to be individually selected on a bus, without decoding. The user port is accessible at all times, independent of whether or not the microprocessor port is selected.

A unique feature of this family is their ability to start up in a predetermined state. If the clock is maintained at a voltage less than .8V until the power supply reaches 3.5V, the user port will always be all logic 1 levels, while the IV port will be all logic 0 levels.

ORDERING

The 8T32/33/35/36 may be ordered in preaddressed form. To order a preaddressed IV Byte, use the following part number format:

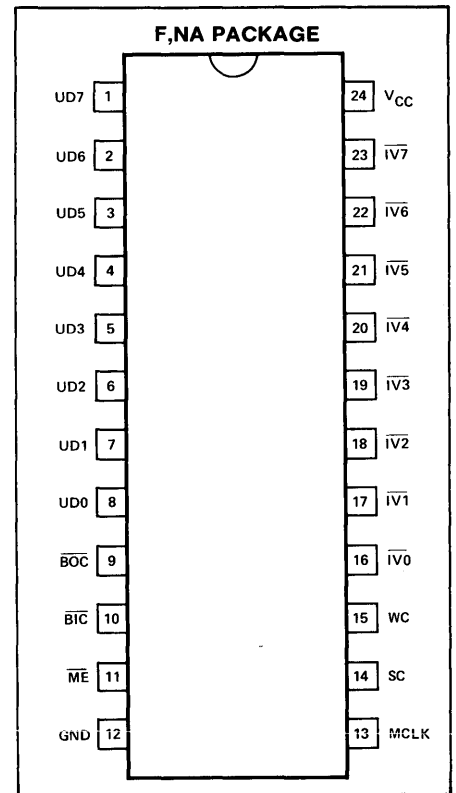
- N8TYY-XXX P
- P= F Ceramic package
NA Plastic package
 - XXX= Any address from 000 through 255 (decimal) - 256 available addresses
 - YY= IV Byte version (32, 33, 35, 36)

A stock of 8T32s and 8T36s with addresses 1 through 10 will be maintained. A small quantity of addresses 11 through 50 will also be available with a longer lead time.

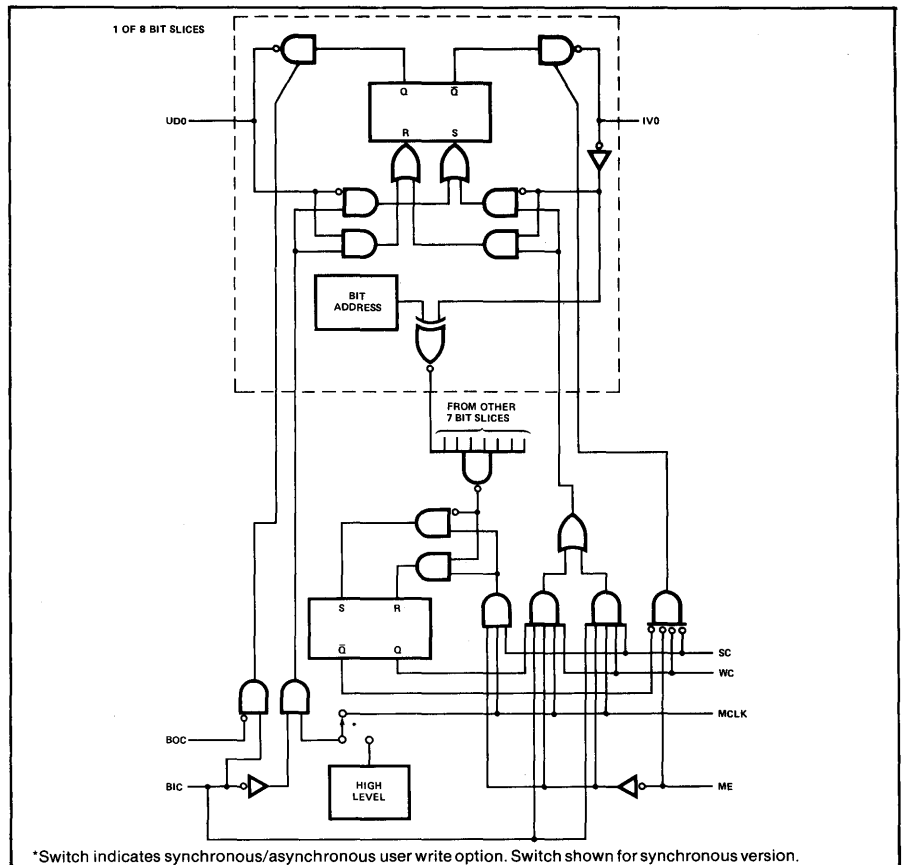
FEATURES

- A field-programmable address allows 1 of 512 IV Bytes on a bus to be selected, without decoders.
- Each byte has 2 ports, one to the user, the other to a microprocessor. IV Bytes are completely bidirectional.
- Ports are independent, with the user port having priority for data entry.
- A selected IV Byte de-selects itself when another IV Byte address is sensed.
- User data input available as synchronous (8T32, 8T33) or as asynchronous (8T35, 8T36) function.
- The User Data Bus is available with tri-state (8T32, 8T36) or open collector (8T33, 8T35) outputs.
- At power up, the IV Byte is not selected and the user port outputs are high.
- Tri-state TTL outputs for high drive capability.
- Directly compatible with the 8X300 Interpreter.
- Operates from a single 5V power supply over a temperature range of 0° C to 70° C.

PIN CONFIGURATION



BLOCK DIAGRAM



*Switch indicates synchronous/asynchronous user write option. Switch shown for synchronous version.

PIN DESCRIPTION

PIN	SYMBOL	NAME AND FUNCTION	TYPE
1-8	UD0-UD7:	User Data I/O Lines. Bidirectional data lines to communicate with user's equipment. Either tri-state or open collector outputs are available.	Active high
16-23	IV0-IV7:	Interface Vector (IV) Bus. Bidirectional data lines to communicate with controlling digital system (microprocessor).	Active low three-state
10	BIC:	Byte Input Control. User input to control writing into the IV Byte from the User Data Lines.	Active low
9	BOC:	Byte Output Control. User input to control reading from the IV Byte onto the User Data Lines.	Active low
11	ME:	Master Enable. System input to enable or disable all other system inputs and outputs. It has no effect on user inputs and outputs.	Active low
15	WC:	Write Command. When WC is high and SC is low, IV Byte, if selected, stores contents of IV0-IV7 as data.	Active high
14	SC:	Select Command. When SC is high and WC is low, data on IV0-IV7 is interpreted as an address. IV Byte selects itself if its address is identical to IV bus data; it de-selects itself otherwise.	Active high
13	MCLK:	Master Clock. Input to strobe data into the latches. See function tables for details.	Active high
24	VCC:	5V power connection.	
12	GND:	Ground.	

USER DATA BUS CONTROL

The activity of the User Data Bus is controlled by the BIC and BOC inputs as shown in Table 1.

For the 8T32 and 8T33, User Data Input is a synchronous function with MCLK. A low level on the BIC input allows data on the User Data Bus to be written into the Data Latches only if MCLK is at a high level. For the 8T35 and 8T36, User Data Input is an asynchronous function. A low level on the BIC input allows data on the User Data Bus to be latched regardless of the level of the MCLK input. Note that when 8T35 or 8T36 IV Bytes are used with the 8X300 Interpreter care must be taken to insure that the IV Bus is stable when it is being read by the 8X300 Interpreter.

To avoid conflicts at the Data Latches, input from the Microprocessor Port is inhibited when BIC is at a low level. Under all other conditions the 2 ports operate independently.

INTERFACE VECTOR BUS CONTROL

As is shown in Table 2, the activity of the microprocessor port (IV Bus) is controlled by the ME, SC, WC and BIC inputs, as well as the state of an internal status latch. BIC is included to show user port priority over the microprocessor port for data input.

Each IV Byte's status latch stores the result of the most recent IV Byte select; it is set when the IV Byte's internal address matches the IV Bus. It is cleared when an address that differs from the internal address is presented on the IV Bus.

In normal operation, the state of the status latch acts like a master enable; the microprocessor port can transfer data only when the status latch is set.

When SC and WC are both high, data on the IV Bus is accepted as data, whether or not the IV Byte was selected. The data is also interpreted as an address. The IV Byte sets its select status if its address matches the data read when SC and WC were both high; it resets its select status otherwise.

BUS OPERATION

Data written into the IV Byte from one port will appear inverted when read from the other port. Data written into the IV Byte from one port will not be inverted when read from the same port.

BIC	BOC	MCLK	USER DATA BUS FUNCTION	
			8T32, 8T33	8T35, 8T36
H	L	X	Output Data	Output Data
L	X	H	Input Data	Input Data
L	X	L	Inactive	Input Data
H	H	X	Inactive	Inactive

H = High Level L = Low Level X = Don't care

Table 1 USER PORT CONTROL FUNCTION

ME	SC	WC	MCLK	BIC	STATUS LATCH	IV BUS FUNCTION
L	L	L	X	X	SET	Output Data
L	L	H	H	H	SET	Input Data
L	H	L	H	X	X	Input Address
L	H	H	H	L	X	Input Address
L	H	H	H	H	X	Input Data and Address
L	X	H	L	X	X	Inactive
L	H	X	L	X	X	Inactive
L	L	H	H	L	X	Inactive
L	L	X	X	X	Not Set	Inactive
H	X	X	X	X	X	Inactive

Table 2 MICROPROCESSOR PORT CONTROL FUNCTION

ADDRESS PROGRAMMING

The IV Byte is manufactured such that an address of all high levels (> 2V) on the IV Data Bus inputs matches the Byte's internal address. To program a bit so a low-level input (< 0.8V) matches, the following procedure should be used:

1. Set all control inputs to their inactive state (BIC = BOC = ME = V_{CC}, SC = WC = MCLK = GND). Leave all IV Data Bus I/O pins open.
2. Raise V_{CC} to 7.75V ± .25V.
3. After V_{CC} has stabilized, apply a single programming pulse to the User Data Bus bit where a low-level match is desired. The voltage should be limited to 18V; the current should be limited to 75mA. Apply the pulse as shown in Figure 1.
4. Return V_{CC} to 0V. (Note 6).
5. Repeat this procedure for each bit where a low-level match is desired.
6. Verify that the proper address is programmed by setting the Byte's status latch (IV0-IV7 = desired address, ME = WC = L, SC = MCLK = H). If the proper address has been programmed, data presented at the IV Bus will appear inverted on the User Bus outputs. (Use normal V_{CC} and input voltage for verification.)

After the desired address has been programmed, a second procedure must be followed to isolate the address circuitry. The procedure is:

1. Set V_{CC} and all control inputs to 0V. (V_{CC} = BIC = BOC = ME = SC = WC = MCLK = 0V). Leave all IV Data Bus I/O pins open.
2. Apply a protect programming pulse to every User Data Bus pin, one at a time. The voltage should be limited to 14V; the current should be limited to 150mA. Apply the pulse as shown in Figure 2.
3. Verify that the address circuitry is isolated by applying 7V to each User Data Bus pin and measuring less than 1mA of input current. The conditions should be the same as in step 1 above. The rise time on the verification voltage must be slower than 100µs.

Absolute Maximum Ratings:

Supply voltage (Note 1) 7V
 Input Voltage (Note 1) 5.5V

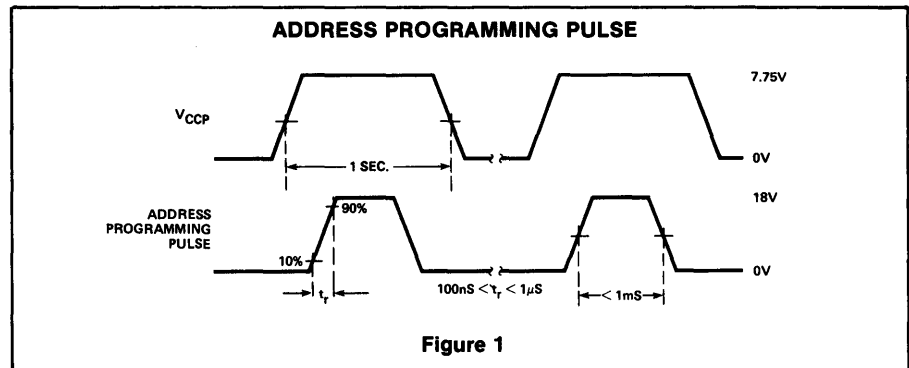


Figure 1

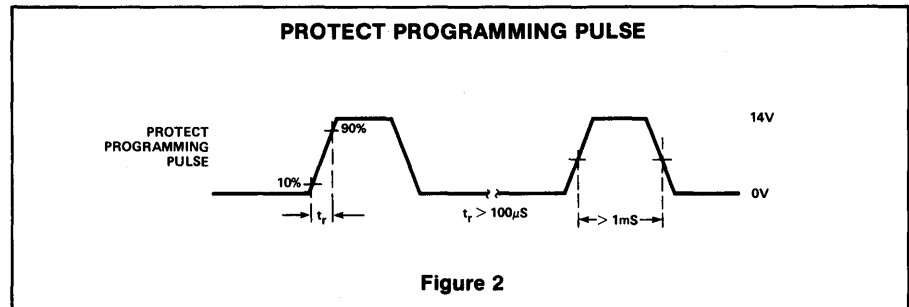


Figure 2

AC ELECTRICAL CHARACTERISTICS

PARAMETER	INPUT	TEST CONDITION	LIMITS			UNIT
			Min	Typ	Max	
t _{PD} User data delay (Note 1)	UDX MCLK* BIC†	C _L = 50pF		25 45 40	38 61 55	ns
t _{OE} User output enable	BOC	C _L = 50pF	18	26	47	ns
t _{OD} User output disable	BIC BOC	C _L = 50pF	18 16	28 23	35 33	ns
t _{PD} IV data delay (Note 1)	IVBX MCLK	C _L = 50pF		38 48	53 61	ns
t _{OE} IV output enable	ME SC WC	C _L = 50pF	14	19	25	ns
t _{OD} IV output disable	ME SC WC	C _L = 50pF	13	17	32	ns
t _W Minimum pulse width	MCLK BIC†		40 35			ns
t _{SETUP} Minimum setup time	UD□ BIC* IVX ME SC WC	(Note 2)		15 25 55 30 30 30		ns
t _{HOLD} Minimum hold time	UDX□ BIC* IVX ME SC SC	(Note 2)		25 10 10 5 5 5		ns

* Applies for 8T32 and 8T33 only.

† Applies for 8T35 and 8T36 only.

□ Times are referenced to MCLK for 8T32 and 8T33, and are referenced to BIC for 8T35 and 8T36.

NOTES:

1. Data delays referenced to the clock are valid only if the input data is stable at the arrival of the clock and the hold time requirement is met.
2. Set up and hold times given are for "normal" operation. BIC setup and hold times are for a user write operation. SC setup and hold times are for an IV Byte select operation. WC setup and hold times are for an IV Bus write operation. ME setup and hold times are for both IV write and select operations.

PROGRAMMING SPECIFICATIONS

PARAMETER	TEST CONDITIONS	LIMITS			UNITS
		Min	Typ	Max	
V_{CCP} Programming supply voltage	$V_{CCP} = 8.0V$	7.5	0	8.0	V
Address					V
Protect					V
I_{CCP} Programming supply current		250			mA
Max time $V_{CCP} > 5.25V$		1.0			s
Programming voltage					
Address		17.5		18.0	V
Protect		13.5		14.0	V
Programming current					
Address				75	mA
Protect			150	mA	
Programming pulse rise time					
Address	.1		1	μs	
Protect	100			μs	
Programming pulse width	.5		1	ms	

NOTES

3. If all programming can be done in less than 1 second, VCC may remain at 7.75V for the entire programming cycle.

PARAMETER MEASUREMENT INFORMATION

LOAD CIRCUIT FOR OPEN COLLECTOR OUTPUTS

LOAD CIRCUIT FOR TRI-STATE OUTPUTS

L - H	S1 OPEN
Z - H	S2 CLOSED
H - L	S1 CLOSED
Z - L	S2 OPEN
L - Z	S1 CLOSED
H - Z	S2 CLOSED

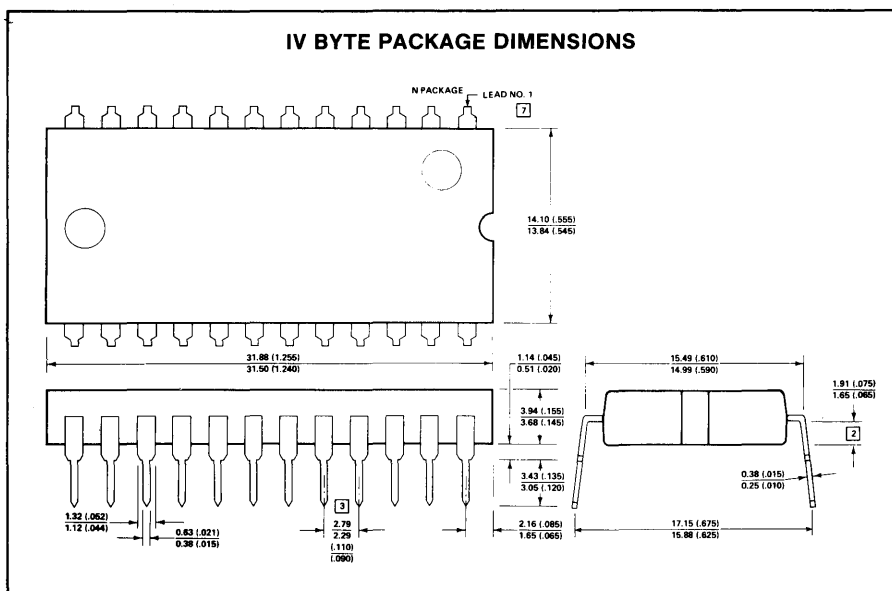
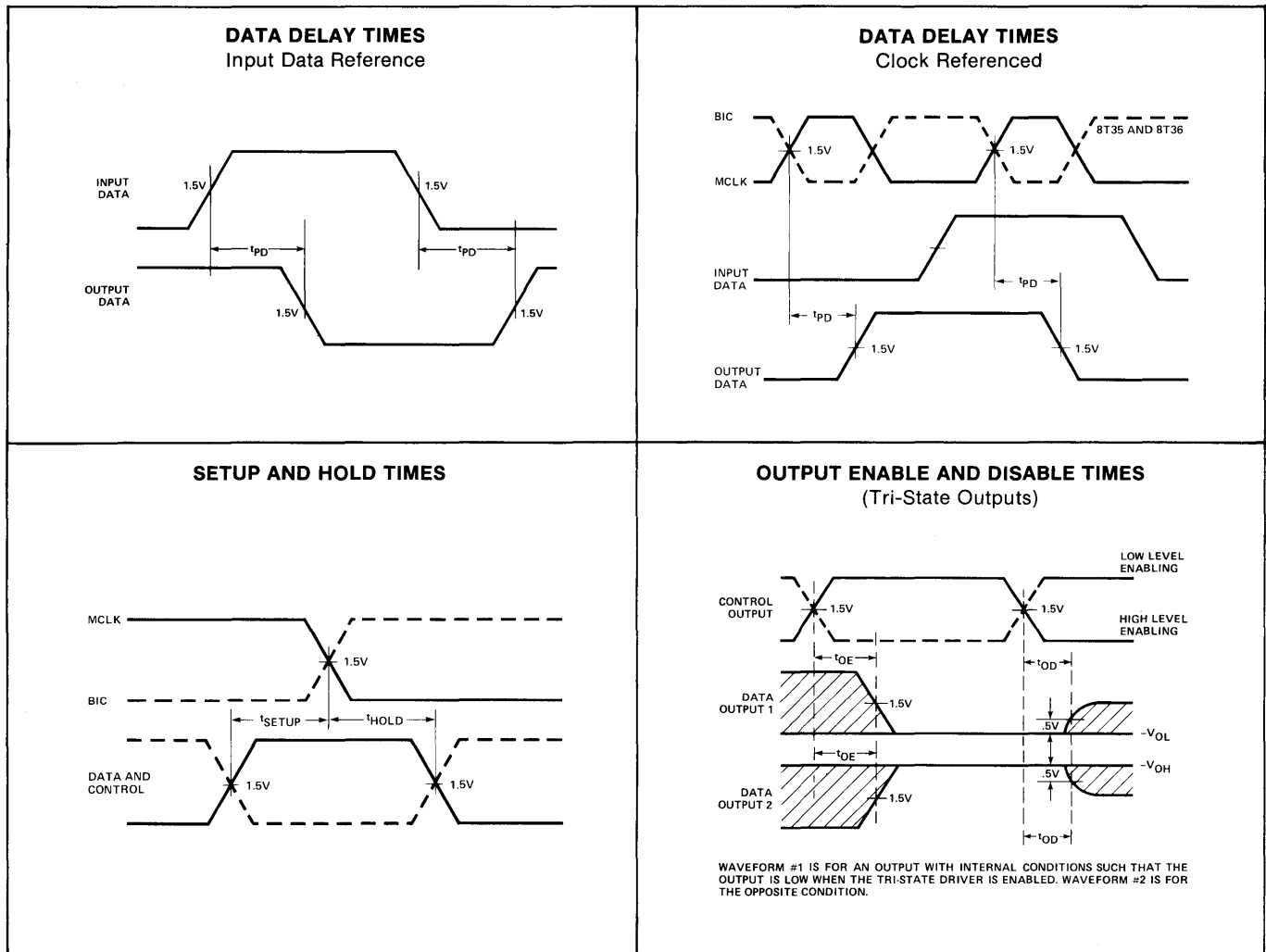
NOTE: C_L includes fixture capacitance.

INPUT WAVEFORM

$t_r \leq 5 \text{ ns}$
 $t_f \leq 5 \text{ ns}$

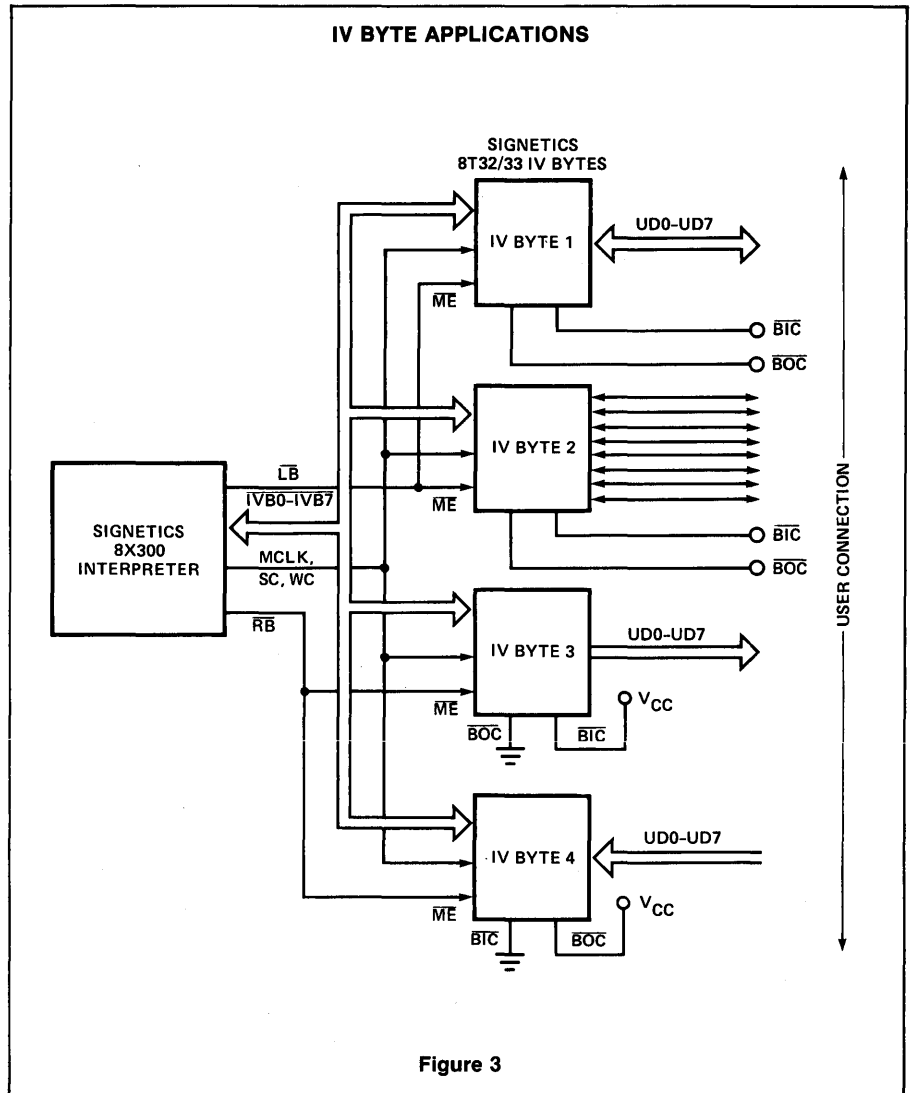
CLOCK PULSE WIDTH

PARAMETER MEASUREMENT INFORMATION (Cont'd)



APPLICATIONS

Figure 3 shows some of the various ways to use the IV Byte in a system. By controlling the BIC and BOG lines, the Bytes may be used for the input and output of data, control, and status signals. IV Byte 1 functions bidirectionally for data transfer and IV Byte 2 provides a similar function for discrete status and control lines. IV Bytes 3 and 4 serve as dedicated output and input ports, respectively.



BIPOLAR PROMS COMPONENTS

PROMS

Field Programmable Read Only Memories

Signetics offers the industry's broadest line of Bipolar High Performance PROMs. These PROMs are field programmable, which means that custom patterns are immediately available by following the provided fusing procedures. Signetics PROMs are supplied with all outputs at logical "0". Outputs are programmed to a logic "1" at any specified address by fusing a Ni-Cr link matrix.

All PROMs are fully TTL compatible, and include on-chip decoding and chip enable functions for ease of memory expansion. Tri-state and open collector output functions are available, and low input currents reduce input buffer requirements.

Most Signetics PROMs also have pin and performance compatible ROMs, offering the user the ultimate in flexibility and cost reduction.

THERMAL RATINGS

TEMPERATURE	MILITARY	COMMERCIAL
Maximum junction	175°C	150°C
Maximum ambient	125°C	75°C
Allowable thermal rise ambient to junction	50°C	75°C

MAXIMUM ALLOWABLE POWER DISSIPATION

MATERIAL	PACKAGE	# OF PINS	θ_{JA}^1 °C/W	$P_{MAX} - mW$	
				0 + 125°C	0 + 75°C
Plastic	B	16	155		480
	XA	18	130	384	577
	N	24	100	500	750
	XF	28	100	500	750
Plastic ²	BA	16	85	588	850
	XAS	18	73	685	>1000
	NA	24	75	666	1000
	XFA	28	75	666	1000
Cerdip	F	16	90	556	
		18	90	556	835
		24	60	830	>1000
Ceramic	I	16	83	600	900
		24	50	1000	>1000
		28	50	1000	>1000

NOTES

1. On a mounted surface, in still air.
2. Improved thermal characteristics due to built-in heat spreader.

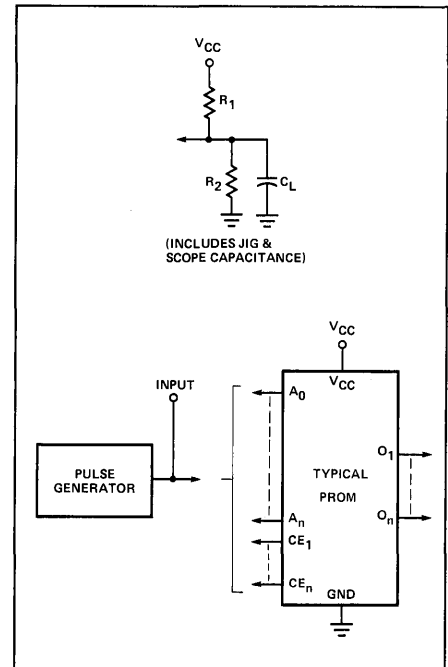
ABSOLUTE MAXIMUM GUARANTEED RATINGS

PARAMETER	LIMITS		UNIT
	Min	Max	
T_A Operating Ambient Temperature	-55	+125	°C
S82S - Military Range	0	+75	°C
N82S - Commercial Range			
T_{STG} Storage Temperature	-65	+150	°C
V_{IN} Input Voltage		+5.5	Vdc
V_{OUT} Output Voltage		+5.5	Vdc
V_{CC} Power Supply Voltage		+7	Vdc

NOTES

1. Stresses above those listed "Maximum, Guaranteed Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device of these or any other condition above those indicated in the operation section of the device specifications is not implied.
2. For operating at elevated temperatures, the device must be derated based on a +150°C maximum junction temperature and a thermal resistance of 160°C/W junction to ambient.
3. For operating at elevated temperatures, the devices must be derated based on a +160°C maximum junction temperature and a thermal resistance of 110°C/W junction to ambient.

AC TEST FIGURE



BIPOLAR PROMS COMPONENTS

ELECTRICAL CHARACTERISTICS S82S Devices — $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5$
 N82S Devices — $0^{\circ}\text{C} \leq T_A \leq +75^{\circ}\text{C}$, $4.75\text{V} + V_{CC} \leq 5.25$

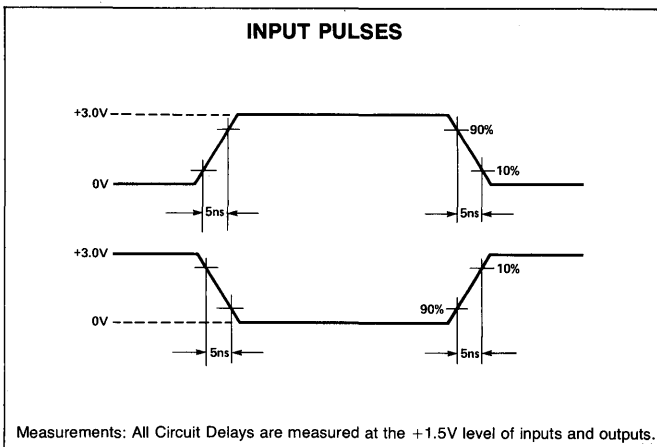
PARAMETER ⁸	INPUT VOLTAGE									OUTPUT VOLTAGE						INPUT CURRENT					
	V_{IL} (V) LOW LEVEL			V_{IH} (V) CLAMP VOLTAGE			V_{IC}^1 (V) LOW LEVEL			V_{OL}^2 (V) HIGH LEVEL			V_{OH}^6 (V) HIGH LEVEL			I_{IL} (μA) LOW LEVEL					
TEST CONDITIONS	$V_{CC} = \text{Min}$			$V_{CC} = \text{Max}$			$I_{IN} = -18\text{mA}$ $V_{CC} = \text{Min}$			$I_{OL} = 16\text{mA}$ $V_{CC} = \text{Min}$			$I_{OUT} = -2.0\text{mA}$ $CE_1 = CE_2 = "0"$ "1" STORED			$V_{IN} = 0.45\text{V}$					
DEVICE	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max			
2048-BIT																					
82S114 N			.85	2.0					-1.2				$I_{OL} = 9.6\text{mA}$			2.7	3.3				-100
82S115 N			.85	2.0					-1.2				$I_{OL} = 9.6\text{mA}$			2.7	3.3				-100

PARAMETER ⁸	INPUT CURRENT			OUTPUT CURRENT									SUPPLY CURRENT			CAPACITANCE					
	I_{IH} (μA) HIGH LEVEL			I_{OLK}^4 (μA) LEAKAGE			I_{O}^4 (OFF (μA) HI-Z STATE			I_{OS}^5 (mA) SHORT CIRCUIT			I_{CC} (mA) ³			C_{IN} (pF) INPUT			C_{OUT}^4 (pF) OUTPUT		
TEST CONDITIONS	$V_{IN} = 5.5\text{V}$			$V_{CC} = \text{Max}$ $V_{OUT} = 5.5\text{V}$ CE_1 or $CE_2 = "1"$			$V_{CC} = \text{Max}$ $V_{OUT} = 5.5\text{V}$			$V_{OUT} = 0\text{V}$ $V_{CC} = \text{Max}$			$V_{CC} = \text{Max}$			$V_{IN} = 2.0\text{V}$ $V_{CC} = 5.0\text{V}$			$V_{CC} = 5.0\text{V}$ $V_{OUT} = 2.0\text{V}$		
DEVICE	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
2048-BIT																					
82S114 N			25		N/A		$V_{OUT} = 0.5\text{V}$		-40	-20		-70		135	185			5			8
82S115 N			25		N/A																

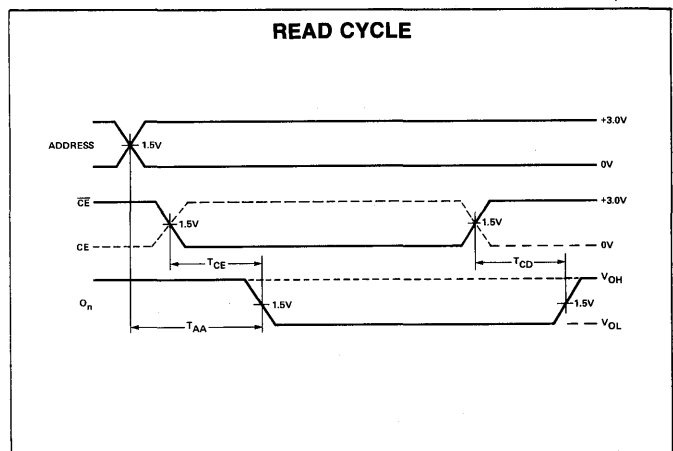
NOTES

1. Test each input one at a time.
2. Measured with the logic "0" stored. Output sink current is supplied through a resistor to V_{CC} .
3. I_{CC} is measured with the write enable and chip enable inputs grounded; all other inputs at 4.5V, and the outputs open.
4. Measured with V_{IH} applied to \overline{CE} .
5. Duration of the short circuit should not exceed one second.
6. Measured with $\overline{CE}(s) = 0\text{V}$, and output(s) at logic "1."
7. All voltage values are with respect to network ground terminal.

INPUT WAVEFORMS



(UNLESS OTHERWISE SPECIFIED)
TYPICAL AC WAVEFORMS



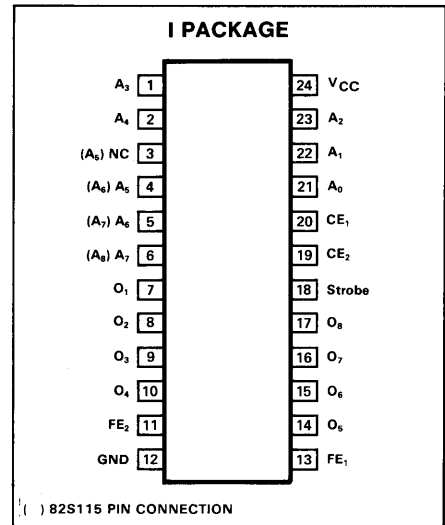
DESCRIPTION

The 82S114 and 82S115 are fully TTL compatible, and include on-chip decoding and two chip enable inputs for ease of memory expansion. They feature Tri-State outputs for optimization of word expansion in bused organizations. A D-type latch is used to enable the Tri-State output drivers. In the TRANSPARENT READ mode, stored data is addressed by applying a binary code to the address inputs while holding STROBE high. In this mode the bit drivers will be controlled solely by CE1 and CE2 lines.

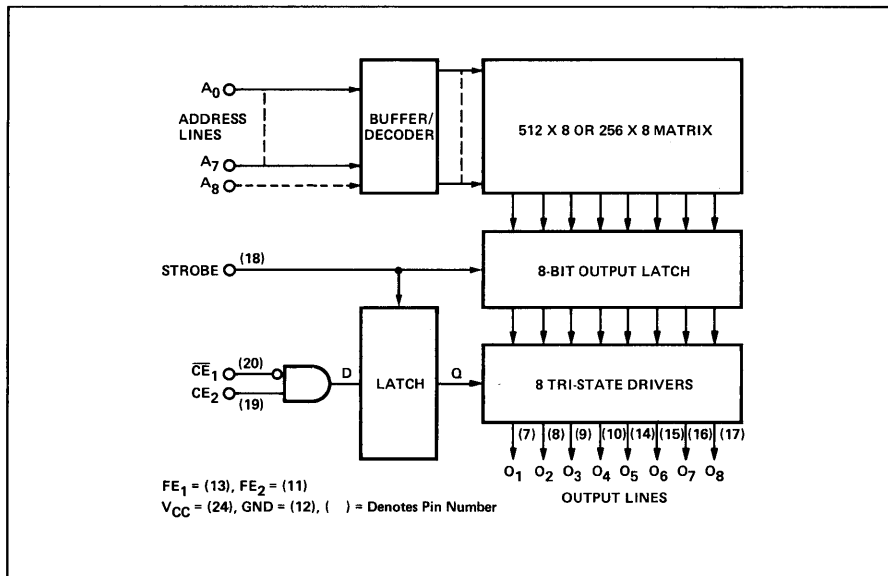
In the LATCHED READ mode, outputs are held in their previous state (1, 0, or High-Z) as long as STROBE is low, regardless of the state of address or chip enable. A positive STROBE transition causes data from the applied address to reach the outputs if the chip is enabled, and causes outputs to go to the High-Z state if the chip is disabled.

A negative STROBE transition causes outputs to be locked into their last Read Data condition if the chip was enabled, or causes outputs to be locked into the High-Z condition if the chip was disabled.

PIN CONFIGURATION



BLOCK DIAGRAM



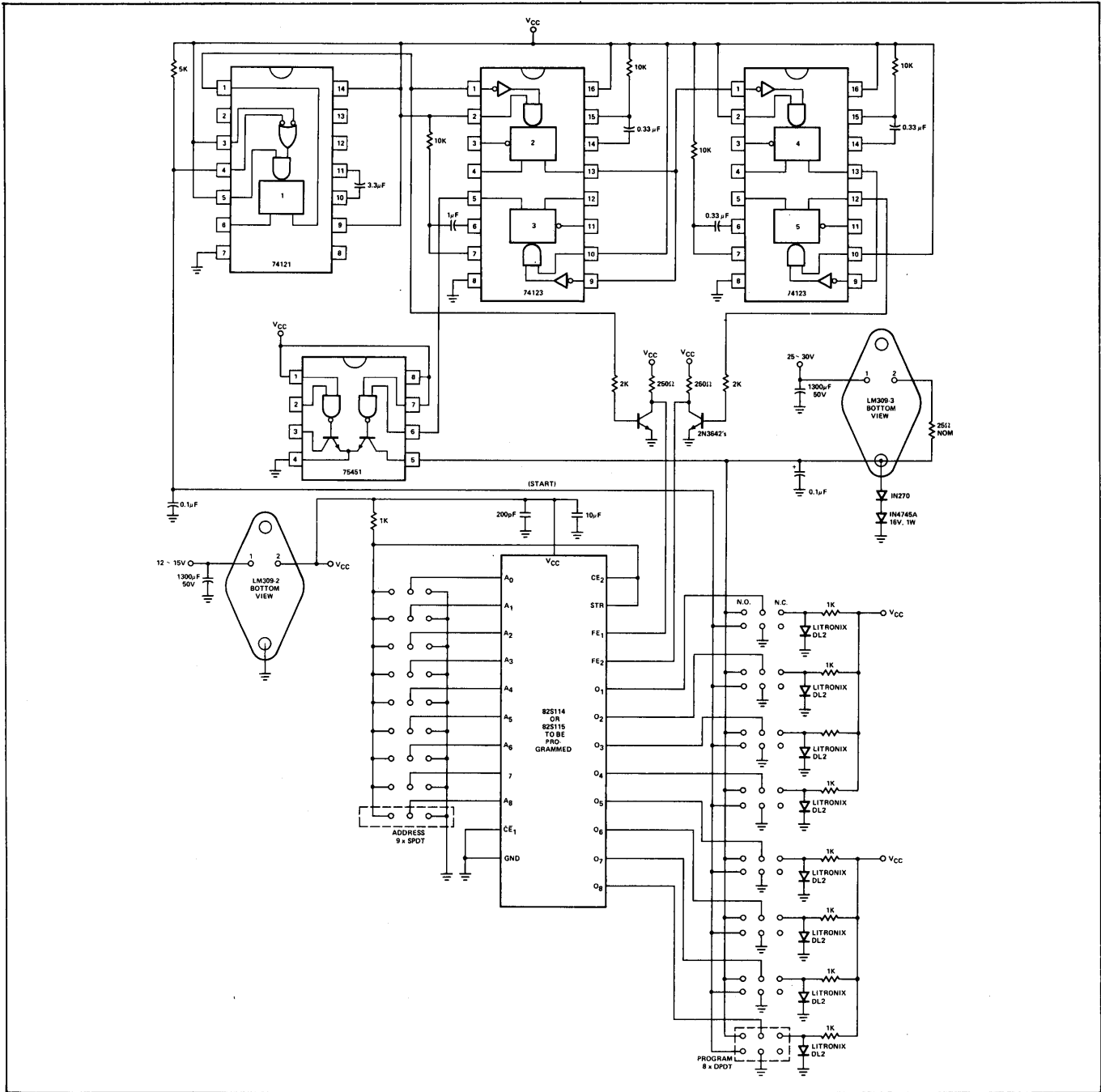
AC ELECTRICAL CHARACTERISTICS 0°C ≤ T_A ≤ +75°C, 4.75V ≤ V_{CC} ≤ 5.25V

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Typ ²	Max	
T _{AA}	Address Access Time		35	60	ns
T _{CE}	Chip Enable Access Time		20	40	ns
T _{CD}	Chip Disable Time		20	40	ns
T _{ADH}	Address Hold Time	0	-10		ns
T _{CDH}	Chip Enable Hold Time	10	0		ns
T _{SW}	Strobe Pulse Width	30	20		ns
T _{SL}	Strobe Latch Time	60	35		ns
T _{DL}	Strobe Delatch Time			30	ns
T _{CDS}	Chip Enable Set-up Time	40			ns

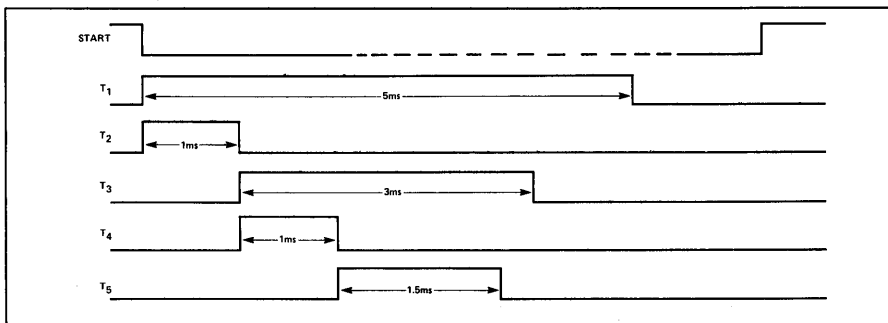
NOTES

- Positive current is defined as into the terminal referenced.
- Typical values are at V_{CC} = +5.0V and T_A = +25°C.
- No more than one output should be grounded at the same time and strobe should be disabled. Strobe is in "1" state.
- If the strobe is high, the device functions in a manner identical to conventional bipolar ROMs. The timing diagram shows valid data will appear TA nanoseconds after the address has changed and T_{CE} nanoseconds after the output circuit is enabled. T_{CD} is the time required to disable the output and switch it to an "off" or high impedance state after it has been enabled.
- In Latched Read Mode data from any selected address will be held on the output when strobe is lowered. Only when strobe is raised will new location data be transferred and chip enable conditions be stored. The new data will appear on the outputs if the chip enable conditions enable the outputs.

82S114/115 MANUAL PROGRAMMER



TIMING SEQUENCE



PROGRAMMING SPECIFICATIONS (Testing of these limits may cause programming of device.) $T_A = +25^\circ\text{C}$

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Typ	Max	
Power Supply Voltage					
V_{CCP1}	To Program	$I_{CCP} = 200 \pm 25\text{mA}$ (Transient or steady state)			V
V_{CCH}	Upper Verify Limit	5.3	5.5	5.7	V
V_{CCL}	Lower Verify Limit	4.3	4.5	4.7	V
V_{S3}	Verify Threshold	0.9	1.0	1.1	V
I_{CCP}	Programming Supply Current	$V_{CCP} = +5.0 \pm .25\text{V}$			mA
Input Voltage					
V_{IL}	Low Level Input Voltage	0	0.4	0.8	V
V_{IH}	High Level Input Voltage	2.4		5.5	V
Input Current (FE₁ & FE₂ Only)					
I_{IL}	Low Level Input Current	$V_{IL} = +0.45\text{V}$			μA
I_{IH}	High Level Input Current	$V_{IH} = +5.5\text{V}$			mA
Input Current (Except FE₁ & FE₂)					
I_{IL}	Low Level Input Current	$V_{IL} = +0.45\text{V}$			μA
I_{IH}	High Level Input Current	$V_{IH} = +5.5\text{V}$			μA
V_{OUT2}	Output Programming Voltage	$I_{OUT} = 200 \pm 20\text{mA}$ (Transient or steady state)			V
I_{OUT}	Output Programming Current	$V_{OUT} = +17 \pm 1\text{V}$			mA
T_R	Output Pulse Rise Time	10		50	μs
t_P	FE ₂ Programming Pulse Width	1		1.5	ms
t_D	Pulse Sequence Delay	10			μs
T_{PR}	Programming Time	$V_{CC} = V_{CCP}$			sec
T_{PS}	Programming Pause	$V_{CC} = 0\text{V}$			sec
T_{PR4}	Programming Duty Cycle				
$T_{PR} + T_{PS}$					60

RECOMMENDED PROGRAMMING PROCEDURE

The 82S114/115 are shipped with all bits at logical "0" (low). To write logical "1", proceed as follows:

Set-up

- Apply GND to pin 12.
- Terminate all device outputs with a 10K resistor to V_{CC} .
- Set $\overline{CE1}$ to logic "0", and $CE2$ to logic "1" (TTL levels).
- Set Strobe to logic "1" level.

Program-Verify Sequence

Step 1 Raise V_{CC} to V_{CCP} , and address the word to be programmed by applying TTL "1" and "0" logic levels to the device address inputs.

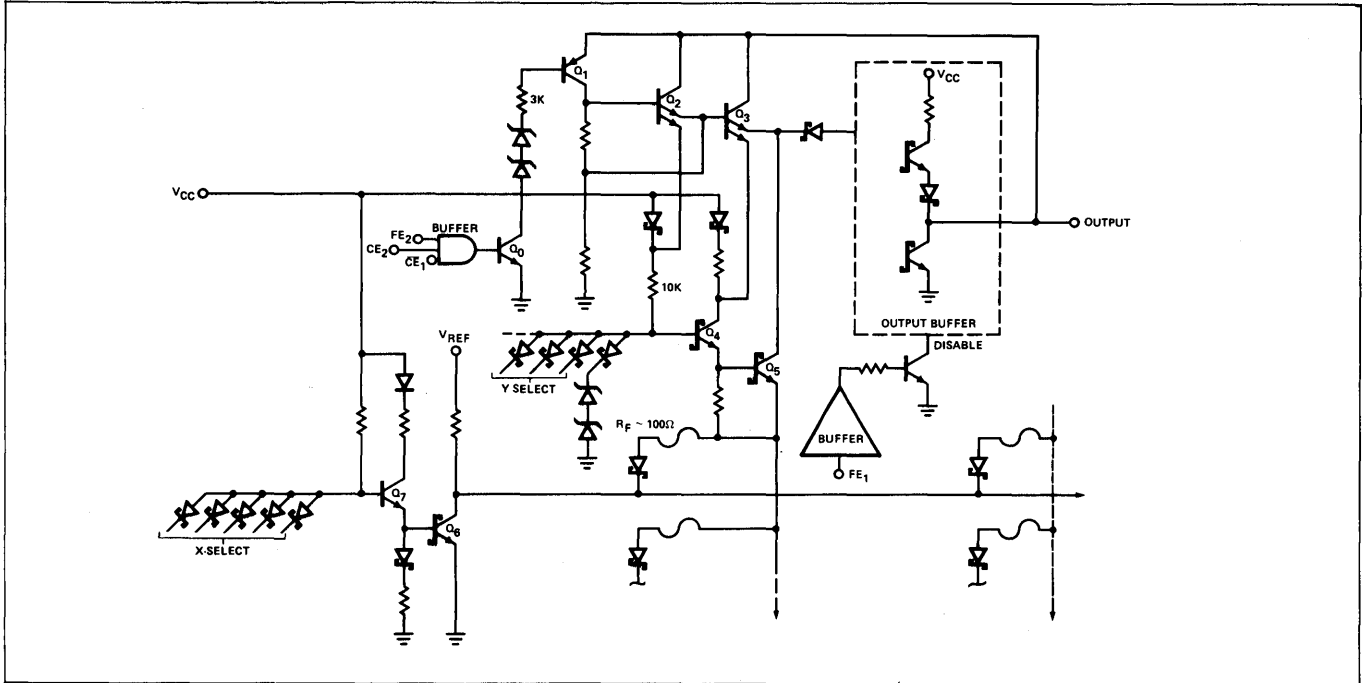
- After $10\mu\text{s}$ delay, apply to FE1 (pin 13) a voltage source of $+5.0 \pm 0.5\text{V}$, with 10mA sourcing current capability.
- After $10\mu\text{s}$ delay, apply a voltage source of $+17.0 \pm 1.0\text{V}$ to the output to be programmed. The source must have a current limit of 200mA. Program one output at the time.
- After $10\mu\text{s}$ delay, raise FE2 (pin 11) from 0V to $+5.0 \pm 0.5\text{V}$ for a period of 1ms, and then return to 0V. Pulse source must have a 10mA sourcing current capability.
- After $10\mu\text{s}$ delay, remove +17.0V supply from programmed output.
- To verify programming, after $10\mu\text{s}$ delay, return FE1 to 0V. Raise V_{CC} to $V_{CCH} = +5.5 \pm .2\text{V}$. The programmed output should remain in the "1" state. Again, lower V_{CC} to $V_{CCL} = +4.5 \pm .2\text{V}$, and verify that

- the programmed output remains in the "1" state.
- Raise V_{CC} to V_{CCP} , and repeat steps 2 through 6 to program other bits at the same address.
- Repeat steps 1 through 7 to program all other address locations.

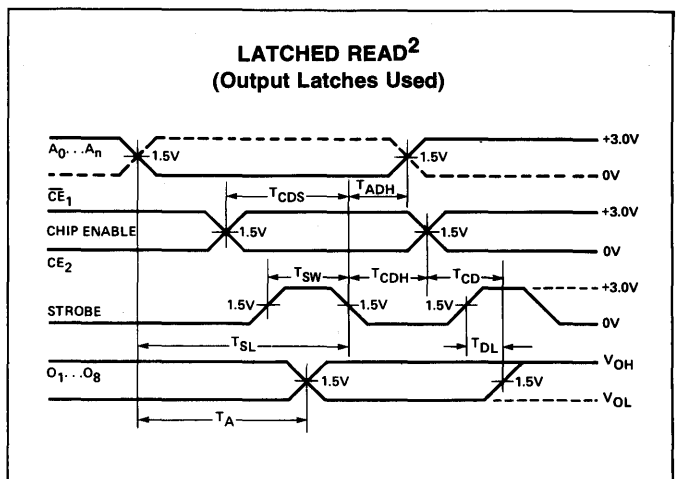
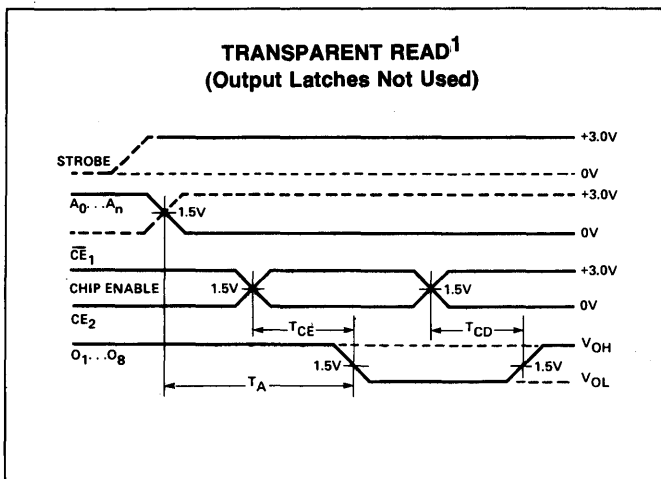
NOTES

- Bypass V_{CC} to GND with a $0.01\mu\text{F}$ capacitor to reduce voltage spikes.
- Care should be taken to insure the $17 \pm 1\text{V}$ output voltage is maintained during the entire fusing cycle. The recommended supply is a constant current source clamped at the specified voltage limit.
- V_S is the sensing threshold of the PROM output voltage for a programmed bit. It normally constitutes the reference voltage applied to a comparator circuit to verify a successful fusing attempt.
- Continuous fusing for an unlimited time is also allowed, provided that a 60% duty cycle is maintained. This may be accomplished by following each Program Verify cycle with a Rest period ($V_{CC} = 0\text{V}$) of 3mS.

TYPICAL FUSING PATH



AC WAVEFORMS



BIPOLAR RAMS COMPONENTS

Random Access Memories

A complete line of Schottky-clamped TTL, read/write memory arrays is offered. All feature open collector or tri-state output options for optimization of word expansion in bused organizations. Memory expansion is further enhanced by full on-chip address decoding, chip enable function, and PNP input transistors which reduce input loading requirements.

All devices offer high performance read access and write cycle times making these devices ideally suited in high speed memory applications such as "caches," buffers, scratch pads, writable control store, main store, etc.

ABSOLUTE MAXIMUM GUARANTEED RATINGS

PARAMETER	LIMITS		UNIT
	Min	Max	
T _A Operating Ambient Temperature	-55	+125	°C
	0	+75	°C
	-65	+150	°C
T _{STG} Storage Temperature		+150	°C
V _{IN} Input Voltage		+5.5	Vdc
V _{OUT} Output Voltage		+5.5	Vdc
V _{CC} Power Supply Voltage		+7	Vdc

NOTE

Stresses above those listed under "Maximum, Guaranteed Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device of these or any other condition above those indicated in the operation section of the device specifications is not implied.

MAXIMUM ALLOWABLE POWER DISSIPATION

MATERIAL	PACKAGE	# OF PINS	θ _{JA} ¹ °C/W	P _{MAX} -mW	
				0 + 125° C	0 + 75° C
Plastic	B	16	155	—	480
	XA	18	130	384	577
	N	24	100	500	750
	XF	28	100	500	750
Plastic ²	BA	16	85	588	850
	XAS	18	73	685	>100
	NA	24	75	666	1000
	XFA	28	75	666	1000
Cerdip	F	16	90	556	835
		18	90	556	835
		24	60	830	>1000
Ceramic	I	16	83	600	900
		24	50	1000	>1000
		28	50	1000	>1000

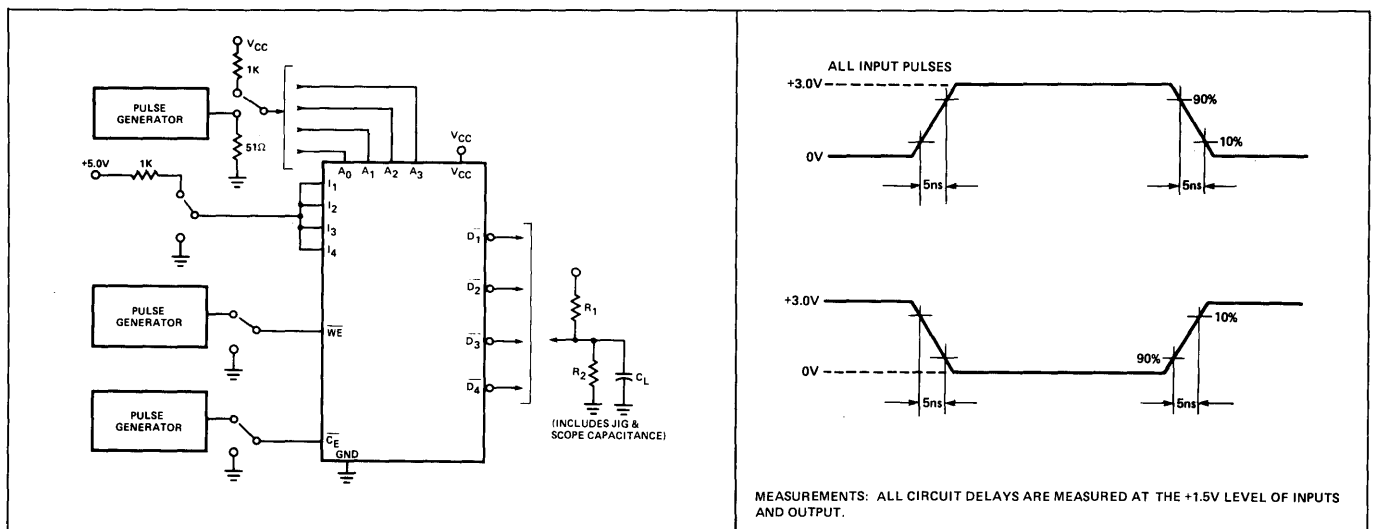
THERMAL RATINGS

TEMPERATURE	MILI-TARY	COM-MERCIAL
Maximum junction	175° C	150° C
Maximum ambient	125° C	75° C
Allow thermal rise ambient to junction	50° C	75° C

NOTES

1. On a mounted surface, in still air.
2. Improved thermal characteristics due to built-in heat spreader.

AC TEST LOAD AND WAVEFORMS



BIPOLAR RAMS COMPONENTS

ELECTRICAL CHARACTERISTICS S82S Devices — $-55^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$, $4.5\text{V} \leq V_{CC} \leq 5.5$
 N82S Devices — $0^{\circ}\text{C} \leq T_A \leq +75^{\circ}\text{C}$, $4.75\text{V} \leq V_{CC} \leq 5.25$

PARAMETER ⁸	INPUT VOLTAGE									OUTPUT VOLTAGE						INPUT CURRENT			
	V _{IL} (V) LOW LEVEL			V _{IH} (V) HIGH LEVEL			V _{IC} ¹ (V) CLAMP VOLTAGE			V _{OL} ² (V) LOW LEVEL			V _{OH} ⁶ (V) HIGH LEVEL			I _{IL} (μA) LOW LEVEL			
TEST CONDITIONS	V _{CC} = Min			V _{CC} = Max			I _{IN} = 12mA V _{CC} = Min			I _{OL} = 16mA V _{CC} = Min			I _{OUT} = -2.0mA CE ₁ = CE ₂ = "0" "1" STORED			V _{IN} = 0.45V			
DEVICE	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
256-BIT																			
82S16 S N			0.80 0.85	2.0					-1.5	0.35 0.35	0.5 0.45		2.4 2.6					-10 -10	-250 -100
82S17 S N			0.80 0.85	2.0					-1.5	0.35 0.35	0.5 0.45		N/A					-10 -10	-250 -100
82S116			0.85	2.0					-1.5	0.35	0.45		2.6					-10	-100
82S117			0.85	2.0					-1.5	0.35	0.45		N/A					-10	-100

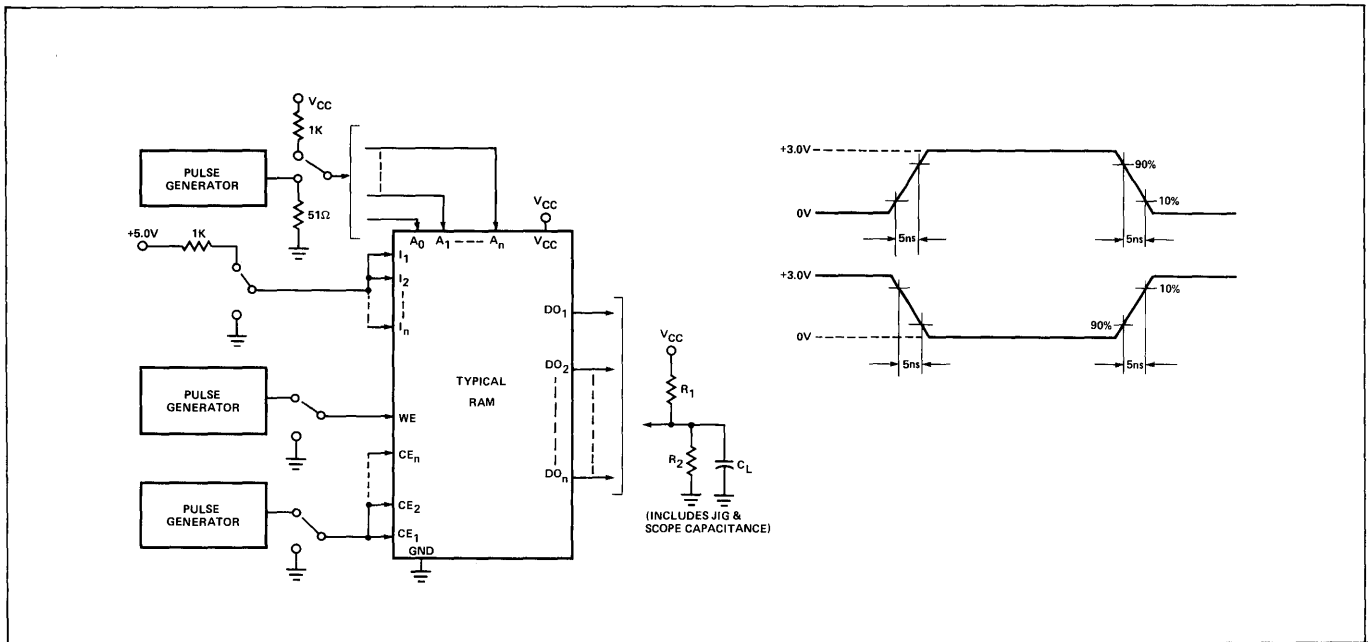
PARA-METER ⁸	INPUT CURRENT			OUTPUT CURRENT									SUPPLY CURRENT			CAPACITANCE								
	I _{IH} (μA) HIGH LEVEL			I _{OLK} ⁴ (μA) LEAKAGE			I _{O(OFF)} ⁴ (μA) HI-Z STATE			I _{OS} (mA) ⁵ SHORT CIRCUIT			I _{CC} ³ (mA)			INPUT (pF)			OUTPUT ⁴ (pF)					
TEST CONDITIONS	V _{IN} = 5.5V			V _{CC} = Max V _{OUT} = 5.5V CE ₁ or CE ₂ = "1"			V _{CC} = Max V _{OUT} = 5.5V			V _{OUT} = 0V V _{CC} = Max			V _{CC} = Max			V _{IN} = 2.0V V _{CC} = 5.0V			V _{CC} = 5.0V V _{OUT} = 2.0V					
DEVICE	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
256-BIT																								
82S16 S N		1	25		N/A			1 1	50 40						80 80									
									V _{OUT} = 0.45V -1 -1			-20			-70						5			8
82S17 S N		1	25		1	40			N/A						80 80						5			8
82S116 N		1	25		N/A			1 -1	40 -40						80 115						5			8
82S117 N		1	25		1	40			N/A						80 115						5			8

NOTES

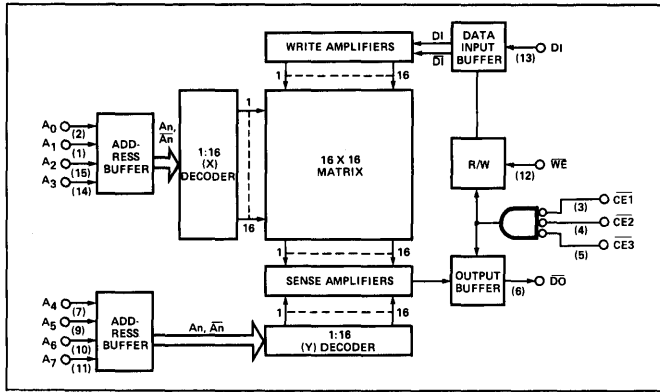
- Test each input one at the time.
- Measured with the logic "0" stored. Output sink current is supplied through a resistor to V_{CC}.
- I_{CC} is measured with the write enable and chip enable inputs grounded; all other inputs at 4.5V, and the outputs open.
- Measured with V_{IH} applied to CE.
- Duration of the short circuit should not exceed one second.
- Measured with CE(s) = 0V, and output(s) at logic "1."
- 10°C ≤ T_A ≤ 75°C
- All voltage values are with respect to ground terminal.
- The Operating Ambient Temperature Ranges are guaranteed with transverse air flow exceeding 400 linear feet per minute and a two minute warm-up. Typical thermal resistance values of the package at maximum temperature are:
 ϕ_{JA} Junction to Ambient at 400fpm air flow—50°C/Watt
 ϕ_{JA} Junction to Ambient—still air—90°C/Watt
 ϕ_{JA} Junction to Case—20°C/Watt

BIPOLAR RAMS COMPONENTS

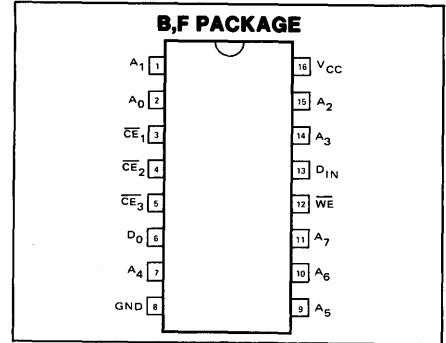
AC TEST FIGURE



BLOCK DIAGRAM



PIN CONFIGURATION



TRUTH TABLE

MODE	CE*	WE	D _{IN}	D _{OUT}	
				82S16/116	82S17/117
READ	0	1	X	STORED DATA	STORED DATA
WRITE "0"	0	0	0	1	1
WRITE "1"	0	0	1	0	0
DISABLED	1	X	X	High-Z	1

*"0" = All CE inputs low; "1" = one or more CE inputs high.
 X = Don't care.

AC ELECTRICAL CHARACTERISTICS 0°C ≤ T_A ≤ +75°C, 4.75V ≤ V_{CC} ≤ 5.25V

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Typ ¹	Max	
Propagation Delays					
T _{AA}	Address Access Time				
T _{CE}	Chip Enable Access Time		30	40	ns
T _{CD}	Chip Enable Output Disable Time		15	25	ns
T _{WD}	Write Enable to Output Disable Time		15	25	ns
			30	40	ns
Write Set-up Times					
T _{WSA}	Address to Write Enable	0	-5		ns
T _{WSD}	Data In to Write Enable	25	15		ns
T _{WSC}	CE to Write Enable	0	-5		ns
Write Hold Times					
T _{WHA}	Address to Write Enable	0	-5		ns
T _{WHD}	Data In to Write Enable	0	-5		ns
T _{WHC}	CE to Write Enable	0	-5		ns
T _{WP}	Write Enable Pulse Width	25	15		ns

54/74 ELECTRICAL CHARACTERISTICS (See Notes—Page 46)

PARAMETER	INPUT VOLTAGE									OUTPUT VOLTAGE						INPUT CURRENT			
	V _{IL} (V) LOW LEVEL			V _{IH} (V) HIGH LEVEL			V _{IC} (V) CLAMP VOLTAGE			V _{OL} (V) LOW LEVEL			V _{OH} (V) HIGH LEVEL			I _{IL} (mA) LOW LEVEL			
	V _{CC} = Min			V _{CC} = Min			V _{CC} = Min I _I = -12mA			V _{CC} = Min V _{IN} = * I _{OL} = 16mA V _{OL} = 0.4V			V _{CC} = Min V _{IN} = * I _{OH} = -400μA			V _{CC} = Max V _{IN} = 0.4V			
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
54/7400	54 74			0.8	2					-1.5		0.22	0.4	2.4	3.3				-1.6
54/7427	54 74			0.8	2					-1.5		0.22	0.4	2.4	3.3				-1.6
54/7474	54 74			0.8	2					-1.5		0.22	0.4	2.4	3.5				-1.6
																			-1.6
																			-3.2
54/74147	54 74			0.8	2					-1.5			0.4	2.4					-1.6

PARAMETER	INPUT CURRENT						POWER SUPPLY CURRENT												
	I _{IH} (μA) HIGH LEVEL			I _I (mA) INPUT CURRENT			I _{OS} (mA) SHORT CIRCUIT			I _{CCL} (mA) LOW LEVEL			I _{CCH} (mA) HIGH LEVEL			I _{OH} (μA) REVERSE			
	V _{CC} = Max V _{IN} = 2.4V			V _{CC} = Max V _{IN} = 5.5V			V _{CC} = Max			V _{CC} = Max V _{IN} = 5V			V _{CC} = Max V _{IN} = 0V			V _{CC} = Min V _{IN} = * V _{OH} = 5.5V			
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
54/7400	54 74			40			1	-20		-55		12	33			4		8	
54/7427	54 74			40			1	-20		-55		16	26			10		16	
54/7474	54 74		D	40			1	-20		-57		17	30						
			Present or clock	80			1	-18		-57									
			Clear	120			1												
54/74157	54 74			40			1	-20		-55		30	48						

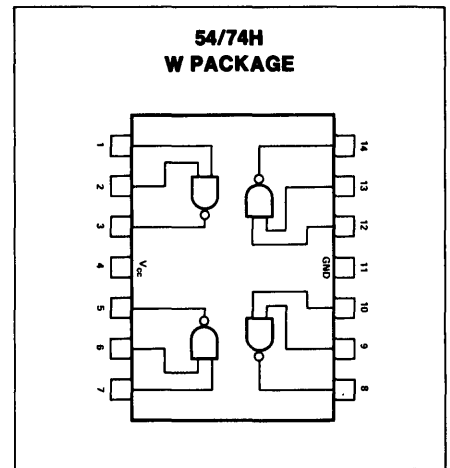
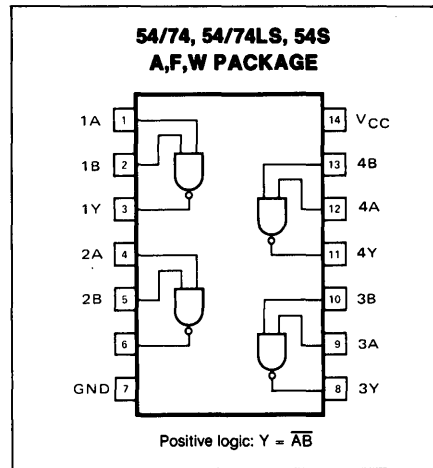
54/74 ELECTRICAL CHARACTERISTICS NOTES

1. All inputs grounded, outputs open.
2. With all outputs open, ICC is measured with Q and Q outputs high in turn. At the time of measurement, the clock input is grounded.
3. ICC is measured with outputs open, A = B grounded, and all other inputs at 4.5V.
4. ICC is measured with all outputs open. Both RO inputs grounded following momentary connection to 4.5V and all other inputs grounded.
5. ICC is measured with all outputs and serial inputs open; A, B, C, and D inputs grounded, mode control at 4.5V and a momentary 3V then ground, applied to both clock inputs.
6. ICC is measured with clear input grounded and all other inputs and outputs open.
7. ICC is measured with outputs open and 4.5V applied to all data and clear inputs; the measurement is made after a momentary ground, then 4.5V is applied to the clock.
8. ICC is measured with inputs at 4.5V, outputs open.
9. ICCL is measured with clock input high, then again with the clock input low with all other inputs low and all outputs open.
10. ICCH is measured with the load input high, then again with the load input low, with all other inputs high and all outputs open.
11. ICC is measured with outputs open, serial inputs grounded, the clock input at 2.4V, and a momentary ground, then 4.5V applied to clear.
12. ICC is measured under the following worst case conditions. 4.5V are applied to all data inputs and both enable inputs, all address inputs are grounded, and all outputs are open.
13. With outputs open, ICC is measured for the following conditions:
 - Condition A—S0 through S3, M and A inputs are at 4.5V, all other inputs grounded.
 - Condition B—S0 through S3 and M are at 4.5V, all other inputs are grounded.
14. ICC is measured with outputs open, clear and load inputs grounded, and all other inputs at 4.5V.
15. With all outputs open, inputs A through D grounded, and 4.5V applied to S0, S1, clear and the serial inputs, ICC is tested with a momentary ground then 4.5V applied to the clock.
16. With all outputs open, shift/load grounded and 4.5V applied to the J, K and data inputs, ICC is measured by applying a momentary ground, followed by 4.5V to clear, then applying a momentary ground followed by a 4.5V to clock.
17. ICC is measured with the outputs open and all data and select inputs at 4.5V under the following conditions:
 - Condition A—Strobe grounded
 - Condition B—Strobe grounded
18. ICC is measured with the outputs open and all data and select inputs at 4.5V under the following conditions:
 - Condition A—All inputs grounded.
 - Condition B—Output control at 4.5V, all inputs grounded.
19. ICC is measured with all outputs open and all possible inputs grounded while achieving the stated output conditions.
20. ICC is measured with one input of each gate at 4.5V, the other inputs grounded, and the outputs open.
21. ICC is measured with outputs open under the following conditions:
 - Condition A—All inputs grounded.
 - Condition B—All B inputs low, other at 4.5V
 - Condition C—All inputs at 4.5V
22. ICC is measured with the outputs open, the serial input and mode control at 4.5V, and the data inputs grounded under the following conditions:
 - Condition A—Output control at 4.5V and a momentary 3V then ground applied to clock input.
 - Condition B—Output control and clock input grounded.
23. ICCL
24. 54/74S ICC limits are per gate.

SPEED/PACKAGE AVAILABILITY

54 F,W	74 A,F
54H F,W	74H A,F
54LS F,W	74LS A,F
54S F,W	74S A,F

PIN CONFIGURATION



SWITCHING CHARACTERISTICS $V_{CC} = 5V, T_A = 25^\circ C$

TEST CONDITIONS	54/74			54/74H			54/74LS			54/74S			UNIT
	$C_L = 15pF$ $R_L = 400\Omega$			$C_L = 25pF$ $R_L = 280\Omega$			$C_L = 15pF$ $R_L = 2k\Omega$			$C_L = 15pF$ $R_L = 280\Omega$			
PARAMETER	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Propagation delay time t_{PLH} Low-to-high		11	22		5.9	10		9	15	2	3 $C_L = 50pF$ 4.5	4.5	ns
t_{PHL} High-to-low		7	15		6.2	10		10	15	2	3 $C_L = 50pF$ 5	5	ns

Load circuit and typical waveforms are shown at the front of section.

SPEED/PACKAGE AVAILABILITY

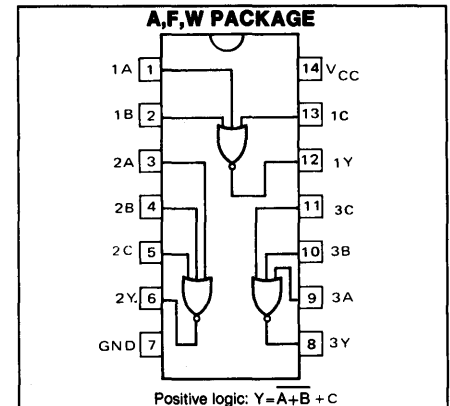
54 F,W	74 A,F
54LS F,W	74LS A,F

SWITCHING CHARACTERISTICS $V_{CC} = 5V, T_A = 25^\circ C$

TEST CONDITIONS	54/74			54/74LS			UNIT
	$C_L = 15pF$ $R_L = 400\Omega$			$C_L = 15pF$ $R_L = 2k\Omega$			
PARAMETER	Min	Typ	Max	Min	Typ	Max	
Propagation delay time t_{PLH} Low-to-high		7	11		5	15	ns
t_{PHL} High-to-low		10	15		9	15	ns

Load circuit and typical waveforms are shown at the front of section.

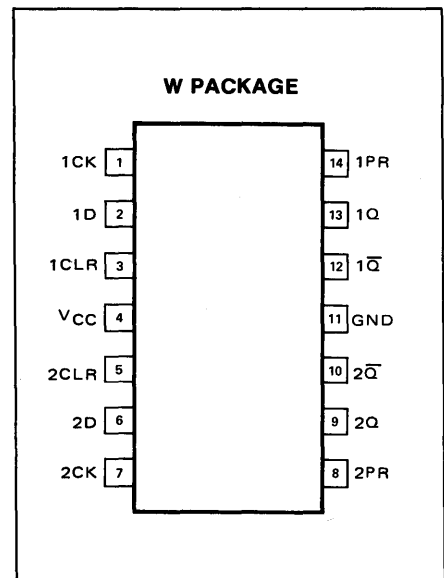
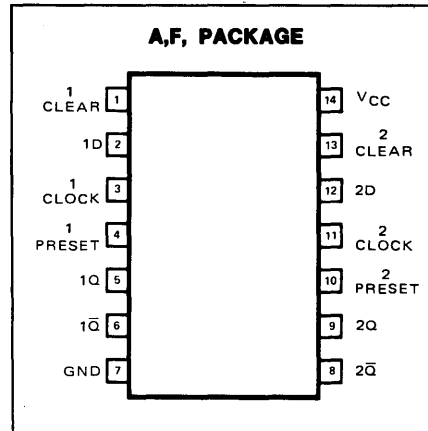
PIN CONFIGURATION



SPEED/PACKAGE AVAILABILITY

54	F,W	74	A,F
54H	F,W	74H	A,F
54LS	F,W	74LS	A,F
54S	F,W	74S	A,F

PIN CONFIGURATION

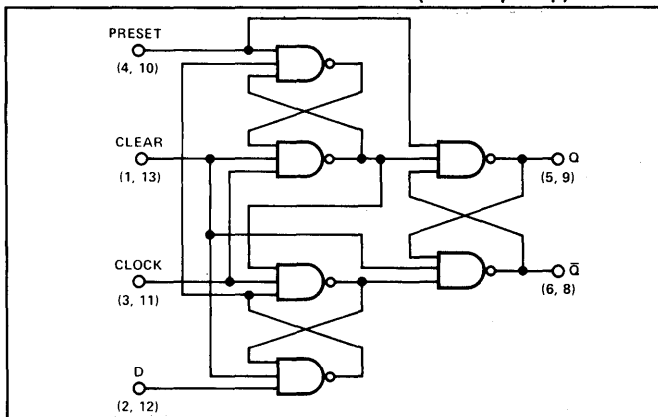


DESCRIPTION

These monolithic dual edge-triggered D-type flip-flops feature individual D, clock, preset, and clear inputs.

Preset and clear inputs are active-low and operate independently of the clock input. When preset and clear are inactive (high), information at the D input is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the high or low level, the D-input signal has no effect at the output.

FUNCTIONAL BLOCK DIAGRAM (Each Flip-Flop)



TRUTH TABLE (Each Flip-Flop)

Preset	INPUTS			OUTPUTS	
	Clear	Clock	D	Q	Q̄
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q ₀	Q̄ ₀

H = high level (steady state) L = low level (steady state)
 *This condition is nonstable. It will not remain after clear and preset return to their inactive (high) state.

SWITCHING CHARACTERISTICS $V_{CC} = 5V, T_A = 25^\circ C$

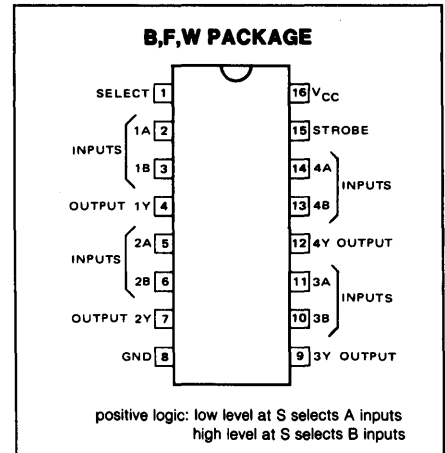
TEST CONDITIONS		FROM INPUT	TO OUTPUT	54/74			54/74H			54/74LS			54/74S			UNIT
				Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f_{Clock}	Clock frequency			15	25		35	43		25	33		75	90		MHz
$t_w(Clock)$	Width of clock input pulse									25	25					
	Clock high			30			15						6			ns
	Clock low			37			13.5						7.3			ns
$t_w(Clear)$	Width of clear input pulse			30			25			25			7			ns
$t_w(Preset)$	Width of preset input pulse			30			25			25			7			ns
t_{Setup}	Input setup time			20↓	15								3↓			ns
	High level						10↓			25						
	Low level						15↓			20						
t_{Hold}	Input hold time			5↓	2		5↓			5			2↓			ns
	Propagation delay time															
t_{PLH}	Low-to-high	Clear, Preset				25			20		8	25		5	6	ns
														8	CLK=1	
														13.5	CLK=0	
t_{PHL}	High-to-low					40			30		16	40		5	8	
t_{PHL}	Low-to-high	Clock				25	4	8.5	15		8	25		7	9	ns
t_{PHL}	High-to-low					40		13	20		16	40		7	9	

Load circuit and typical waveforms are shown at the front of section.

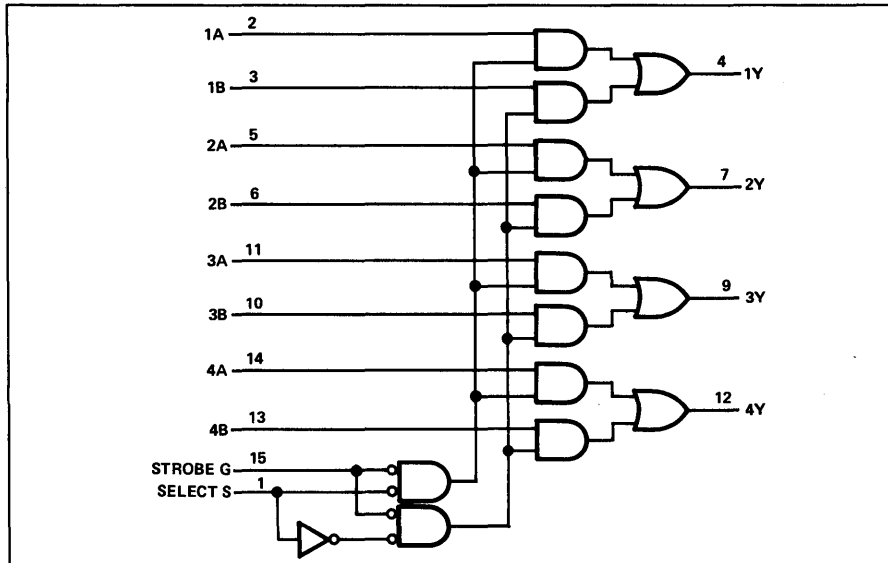
SPEED/PACKAGE AVAILABILITY

54	F,W	74	B,F
54LS	F,W	74LS	B,F
54S	F,W	74S	B,F

PIN CONFIGURATION



BLOCK DIAGRAM



TRUTH TABLE

INPUTS		OUTPUT Y		
STROBE	SELECT	A	B	
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

H = high level, L = low level, X = irrelevant

SWITCHING CHARACTERISTICS $V_{CC} = 5V, T_A = 25^\circ C$

TEST CONDITIONS			54/74			54/74LS			54/74S			UNIT
			$C_L = 15pF$ $R_L = 400\Omega$			$C_L = 15pF$ $R_L = 2k\Omega$			$C_L = 15pF$ $R_L = 280\Omega$			
PARAMETER	FROM INPUT	TO OUTPUT	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Propagation delay time												
t_{PLH} Low-to-high	Data	Any		9	14		9	14		5	7.5	ns
t_{PHL} High-to-low				9	14		9	14		4.5	6.5	
t_{PLH} Low-to-high	Enable	Any		13	20							
t_{PHL} High-to-low				14	21							
t_{PLH} Low-to-high	Select	Any		15	23		15	23		9.5	15	
t_{PHL} High-to-low				18	27		8	27		9.5	15	
t_{PLH} Low-to-high	Strobe	Any					13	20		8.5	12.5	
t_{PHL} High-to-low							14	21		7.5	12	

Load circuit and typical waveforms are shown at the front of section.

APPLICATIONS MEMOS

Signetics acknowledges the cooperation of Scientific Micro Systems, Inc. in the preparation of this chapter of the manual.

DESCRIPTION

Typical interfaces to the 8X300 employ the 8T32/33 or 8T35/36 bidirectional I/O ports. These devices provide a single connection between the 8X300 and the user status control and data lines. Each interface is denoted as an Interface Vector and is field programmed to a specific address.

ADDRESSING DATA ON THE INTERFACE VECTOR

The Interface Vector is comprised of general purpose 8-bit I/O registers called Interface Vector (IV) Bytes. The IV registers serve to select IV bytes. In order for an instruction to access (read or write) an IV byte, the address of that byte must be output to the IVL or IVR registers.

Thus, two instructions are required to operate on an Interface Vector byte:

XMIT ADDRESS, IVL MACHINE INSTRUCTION

Each of the two IV registers (IVL and IVR) may be set to select an IV byte, therefore two I/O ports may be active at one time—one on the Right Bank (IVR) and one on the Left Bank (IVL). Data may be input and output in one instruction following the selection of IV bytes:

XMIT	ADDRESS1,IVL
XMIT	ADDRESS2,IVR
ADD	LB, RB

Once the IV byte is selected (addressed) it will remain selected until another address is output to the same IV register. Since an IV register (IVL, IVR) can be used only as a destination field of an instruction, any instruction sending data to IVL or IVR can be used to select an IV byte.

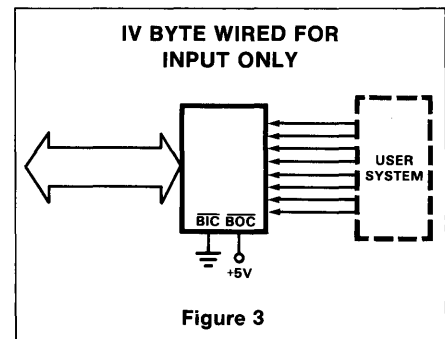
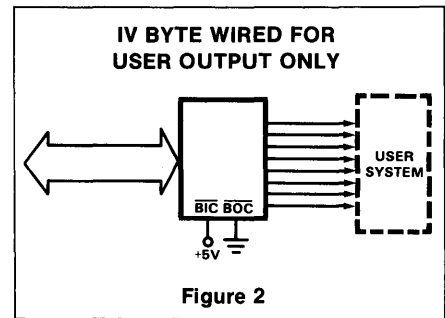
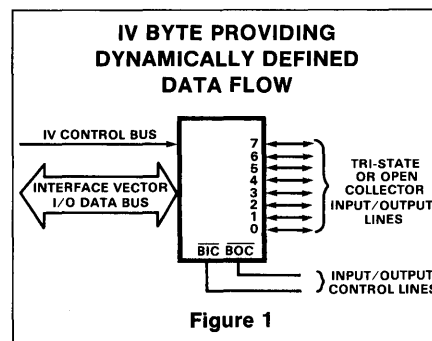
From the user's standpoint, however, all IV byte outputs can be read by an external

device regardless of whether they are selected or not.

The address range of IVL and IVR is 0-255₁₀.

ELECTRICAL CHARACTERISTICS OF THE INTERFACE VECTOR

Each IV byte consists of 8 storage latches which hold data transferred between the Interpreter and the User System, 8 tri-state input/output lines and 2 input/output control lines, called Byte Input Control (BIC) and Byte Output Control (BOC) as shown in Figure 1. The control lines functions are summarized in Table 1. Table 2 contains a summary of the electrical characteristics of the IV byte.



Working storage consisting of RAM may be connected to either or both left and right I/O banks. An example of such an arrangement is shown in Figure 4. Paging may be added to the memory to extend the addressability.

CONTROL LINES		FUNCTION
BOC (low true)	BIC (low true)	
H	H	8 I/O lines in high impedance state—disable
L	H	8 I/O lines in output mode—8 bit storage latch data available in the output lines.
X	L	8 I/O lines in input mode—data can be read by Interpreter

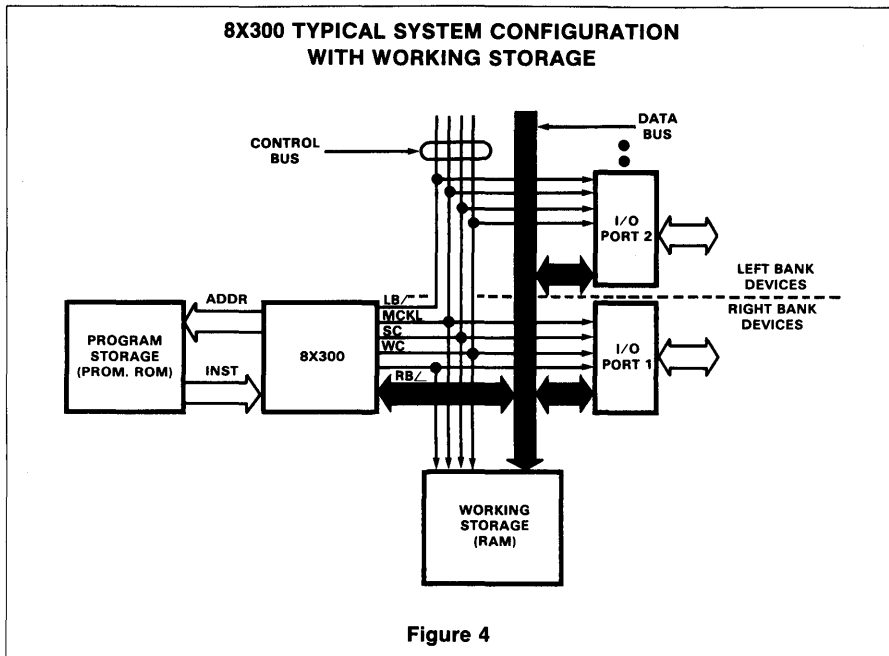
Table 1 FUNCTIONS OF THE BIC AND BOC LINES

PARAMETER	TEST CONDITIONS	LIMITS			UNIT
		Min	Typ	Max	
V _{IH}	High level input voltage	2		5.5	V
V _{IL}	Low level input voltage	-1		0.8	V
V _{IC}	Input clamp voltage			-1	V
V _{OH}	High level output voltage	2.4			V
V _{OL}	Low level output voltage			0.5	V
I _{IH}	High level input current ¹			100	uA
I _{IL}	Low level input current ¹			-800	uA
I _{OS}	Output short circuit current	-20		-200	mA
C _{IN}	Data input capacitance			12	pF

NOTE

1. Input current is always present regardless of the state of BIC and BOC.

Table 2 IV BYTE TERMINAL ELECTRICAL CHARACTERISTICS



FLOPPY DISC INTERFACE

DESCRIPTION

The 8X300 controls a floppy disc drive with a minimal amount of additional circuitry. In this example, byte assembly and disassembly are performed by the program ("bit banging") to reduce interface circuitry. Addition of such circuitry would increase hardware costs and decrease significantly peak processor utilization.

Data is transferred to and from the floppy disc via I/O driver routines. These I/O driver routines provide a standard software interface to a floppy disc and require 180 words of program storage. When not transferring data to and from the disc, the 8X300 is available to service other devices such as keyboards, displays or data communication lines. Figure 1 illustrates the system.

DESIGN APPROACH

Data bytes are assembled or disassembled by sensing a clock, inputting data bit or generating a clock, and outputting a data bit. Preamble patterns, track address, and other disc format requirements are implemented by programming. Disc drive head must be stepped to the desired track before data transfer is initiated. Disc drive status is monitored to determine any error conditions. A four step procedure is followed to implement the design:

1. Analyze interface. Analyze floppy disc relative to: Number of control/data lines, timing and data rates associated with each control/data line, and electrical characteristics of each control/data line. Determine any supplemental circuits needed for electrical compatibility (see Table 1).
2. Perform functional analysis. The functions to be programmed and any which require supplemental logic are determined. In this case, supplemental logic is utilized to process the 250ns pulses associated with DATA, CLOCK and WR DATA. Optional logic for byte assembly and disassembly also are shown. The programmed functions are:
 - a. Byte assembly/disassembly
 - b. Generate preamble, track address, timing and sector synchronization
 - c. Sense clock and disc status
 - d. Step head to desired track
3. Define the program to process input and to generate output (see Figure 2).
4. Determine 8X300 configuration (see Table 2).

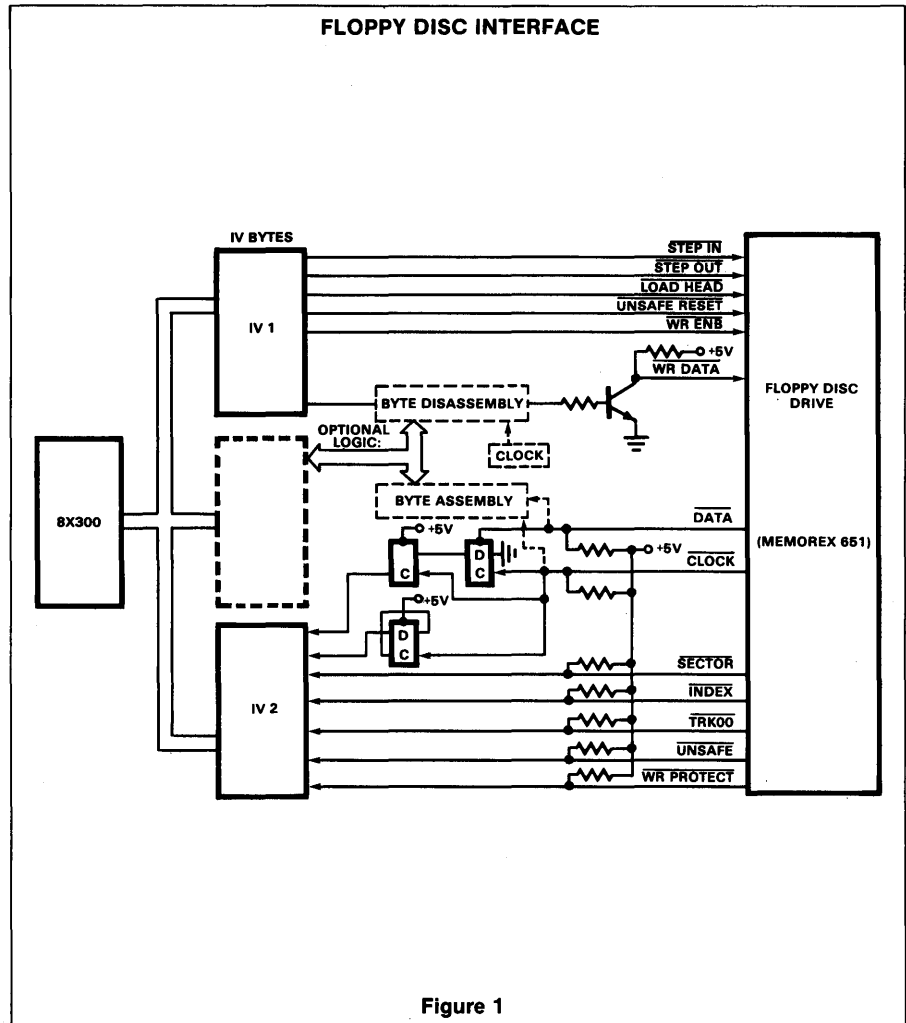
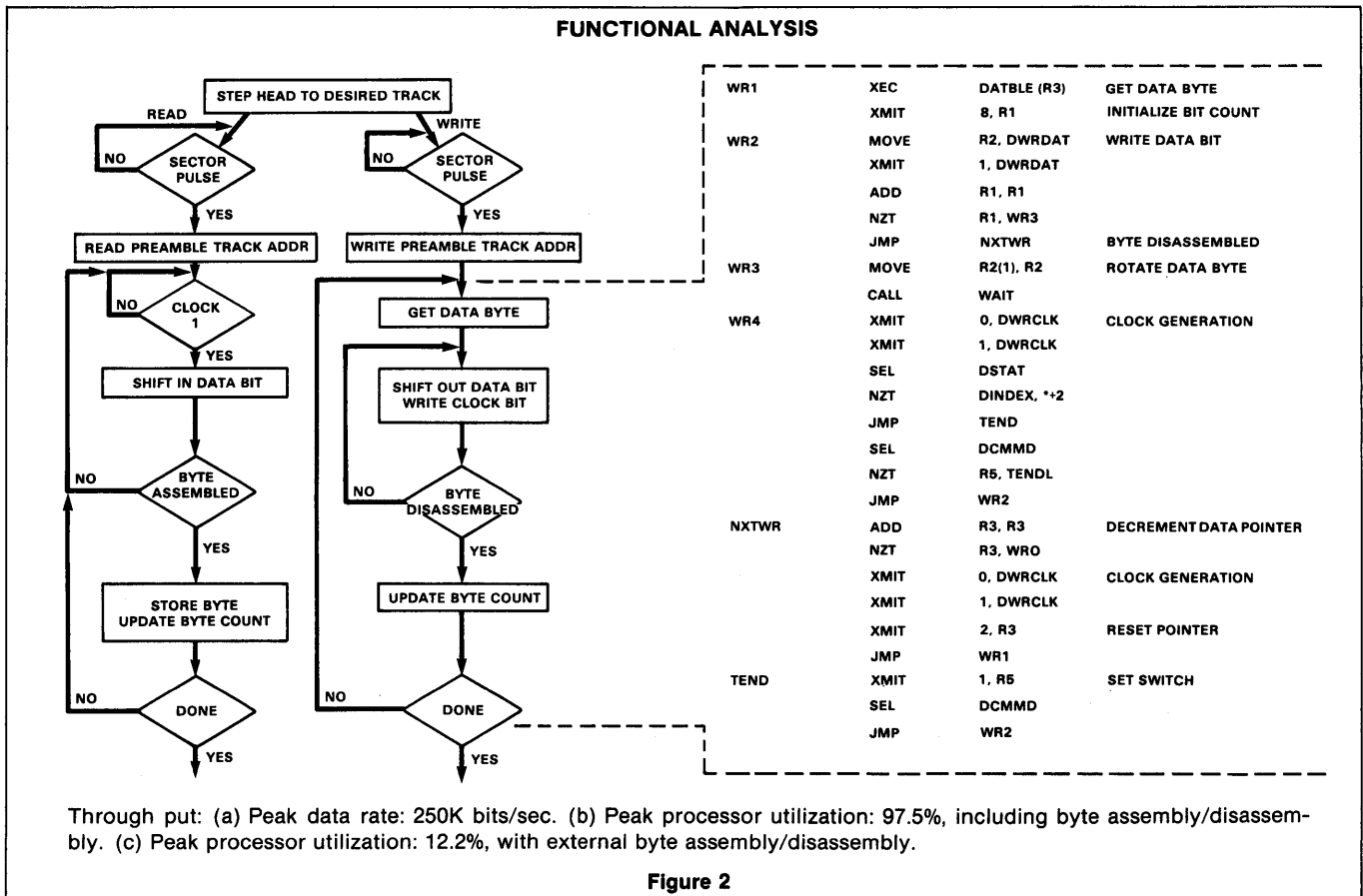


Figure 1

SIGNAL NAME	DATA RATE ⁻¹	SIGNAL DURATION	ELECTRICAL CHARACTERISTICS	# IV BITS	INTERFACE REQUIRED	FUNCTION
STEP IN	20ms	.01-10ms	TTL with pullup	1		Step head 1 track in
STEP OUT	20ms	.01-10ms	TTL with pullup	1		Step head 1 track out
LOAD HEAD	Level		TTL with pullup	1		Load head
UNSAFE RESET	Level		TTL with pullup	1		Clears unsafe condition
WR ENB	Level		TTL with pullup	1		Enables write operation
WR DATA	2μs	.25μs	50mA current	1	2R,T	Data/clock to disc
SECTOR	5ms	1ms	OC output	1	R	Sector indicated
INDEX	160ms	1ms	OC output	1	R	Begin of track indicator
TRK00	Level		OC output	1	R	Head on track 00
UNSAFE	Level		OC output	1	R	Unsafe condition indicator
WR PROTECT	Level		OC output	1	R	Write protected disc
DATA	4μs	.25μs	OC output	1	R,2FF	Data from disc
CLOCK	4μs	.25μs	OC output	1	R,FF	Clock from disc

R = Resistor
 T = Transistor
 FF = Flip-Flop

Table 1 INTERFACE ANALYSIS



ROM/PROM FOR PROGRAM STORAGE	WORKING STORAGE FOR DATA BUFFERS	IV BYTES FOR INPUT/OUTPUT INTERFACE
Input Driver 76 words	256 Bytes	6 IV bits for output 7 IV bits for input Total: 2 IV bytes
Output Driver 74 words		
Head Step Driver 30 words		
Total 180 words		

Table 2 8X300 CONFIGURATION

TELETYPE MULTIPLEXER

DESCRIPTION

The 8X300 is easily interfaced to a teletype or similar asynchronous device. Processor utilization is less than .1%, even when used in a character assembly mode.

A single 8X300 can be used as a multiplexer for many low speed asynchronous devices. For example, the 8X300 can be used as a front end multiplexer for a large computer system. Figure 3 illustrates the system.

DESIGN APPROACH

A basic teletype I/O driver routine receives, transmits and echoes a character. Character assembly/disassembly is implemented by sensing start bit, sampling data bit and generating output bit timing. A four-step procedure is followed to implement the design:

1. Analyze interface. Analyze teletype relative to: Number of control/data lines, timing and data rates associated with each control/data line, electrical characteristics of each control/data line, and determine any supplemental circuits needed for electrical compatibility (see Table 3).
2. Perform functional analysis. The functions to be programmed and any which require supplemental logic are determined. In this case, no supplemental logic is required. The programmed functions are:
 - a. Character assembly/disassembly
 - b. Sense start bit
 - c. Generate bit timing and simultaneous character echo
3. Define the program to process input and to generate output (see Figure 4).
4. Determine 8X300 configuration (see Table 4).

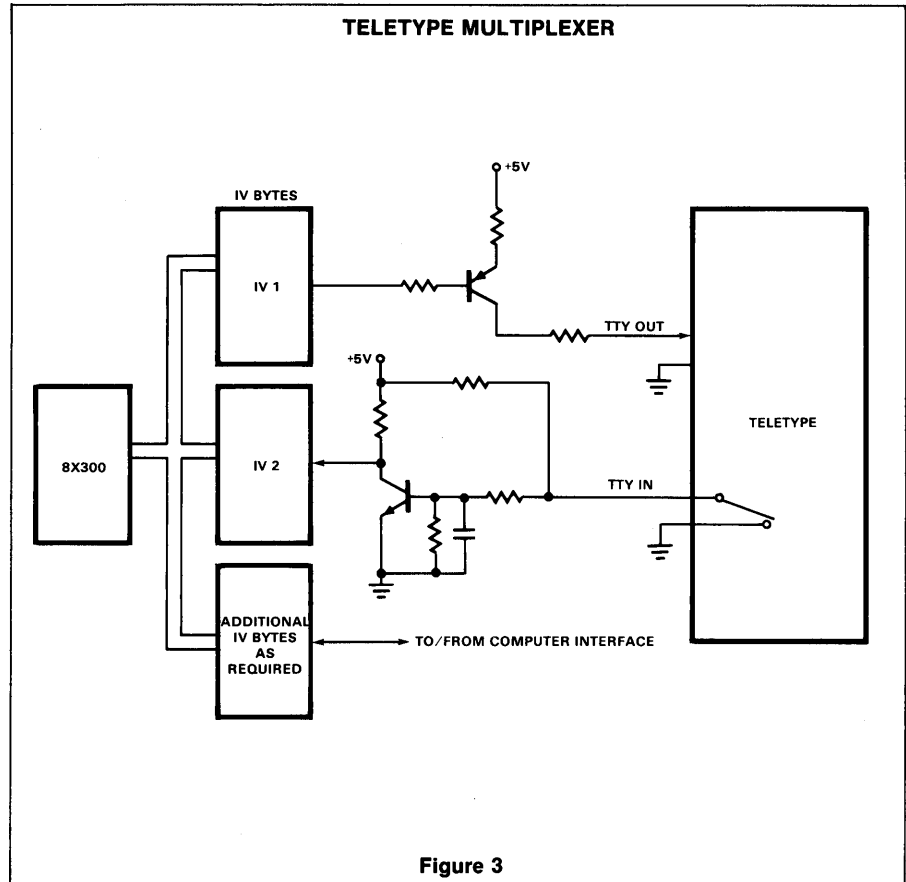
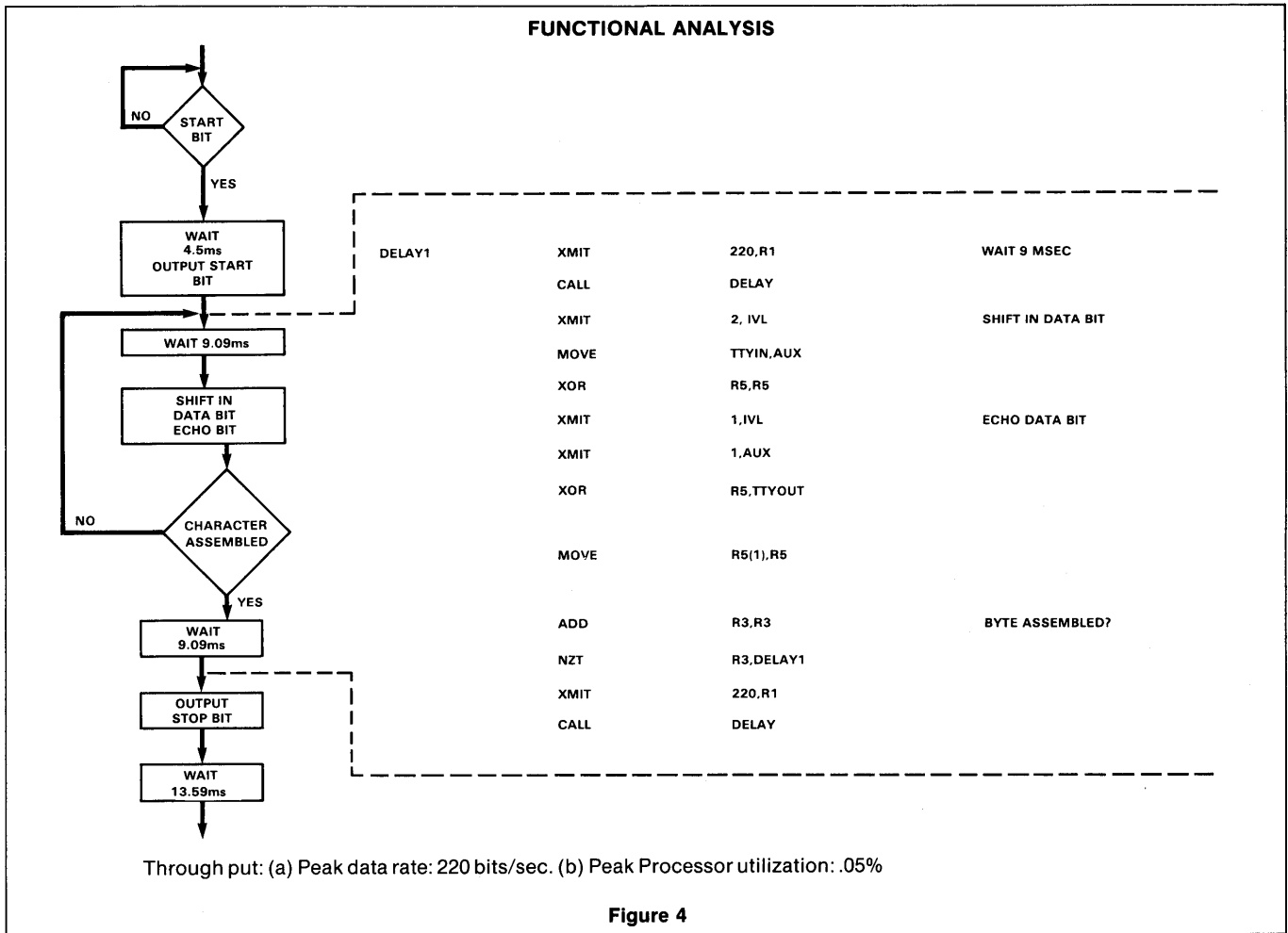


Figure 3

SIG- NAL NAME	DATA RATE ¹	SIGNAL DURA- TION	ELECTRICAL CHARACTER- ISTICS	# IV BITS	INTER- FACE RE- QUIRED	FUNCTION
TTY OUT	9.09ms	9.09ms	20mA current	1	3R,T	Data to TTY printer
TTY IN	9.09ms	9.09ms	20mA current	1	4R,T,C	Data from TTY keyboard

R = Resistor
T = Transistor
C = Capacitor

Table 3 INTERFACE ANALYSIS



ROM/PROM FOR PROGRAM STORAGE	WORKING STORAGE FOR DATA BUFFERS	IV BYTES FOR INPUT/OUTPUT INTERFACE
Teletype driver 49 words	2 bytes per Teletype	1 IV bit, for output, per Teletype 1 IV bit for input, per Teletype Total: 2 IV bytes per 8 Teletypes
Delay routine 10 words		
Total 59 words		

Table 4 8X300 CONFIGURATION

DATA CONCENTRATOR

DESCRIPTION

The 8X300 multiplexes multiple low speed terminals. It buffers the data in its working storage for efficient transmission over common carrier or other data link facilities. Single inquiry/response terminals are interfaced to a single half-duplex synchronous line via a Universal Asynchronous Receive-/Transmit (UART) interface. This eliminates cabling to each terminal. The 8X300 transfers inquiry and response messages between terminals and a remote computer data base via a data communications line. Various communication data rates are accommodated by simple program modification. Figure 5 illustrates the system.

DESIGN APPROACH

The 8X300 polls each terminal requesting an input character or signaling an output character. Each character is transferred over a high speed (9600 baud) synchronous line whose data rate determines the scan time of the 8X300 multiplexing program. The multiplexer program formats polling messages, maintains status, generates and checks the Longitudinal Redundancy Character, performs character recognition, and buffers characters. Additional driver programs are required to communicate with the full duplex data communications line to/from a remote computer data base. A four step procedure is followed to implement the design:

1. Analyze interface. Analyze UART relative to: Number of control/data lines, timing and data rates associated with each control/data line, electrical characteristics of each control/data line and determine any supplemental circuits needed for electrical compatibility (see Table 5).
2. Perform functional analysis. The functions to be programmed and any which require supplemental logic are determined. In this case, no supplemental logic is required. The programmed functions are:
 - a. Maintain current line status
 - b. Generate synchronization pattern, poll command, sense character synch
 - c. Resynchronize with clock and monitor modem and UART status
3. Define the program to process input and to generate output (see Figure 6).
4. Determine the 8X300 configuration (see Table 6).

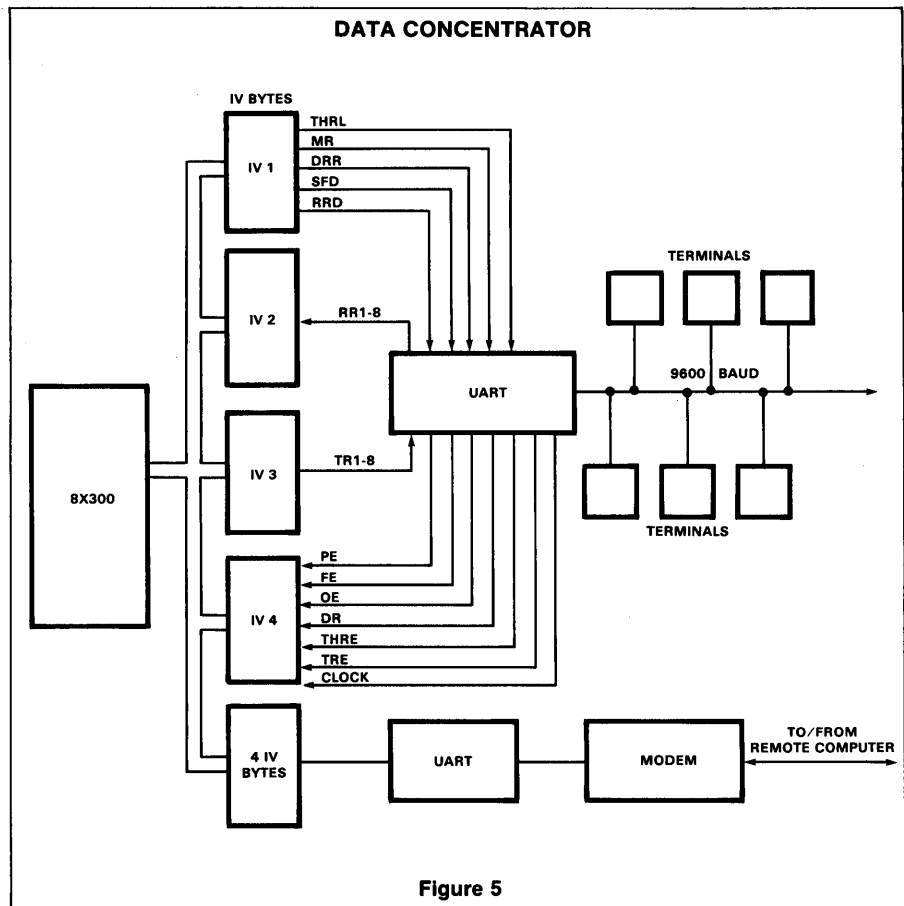


Figure 5

SIGNAL NAME	DATA RATE ¹	SIGNAL DURATION	ELECTRICAL CHARACTERISTICS	# IV BITS	INTERFACE REQUIRED	FUNCTION
TR1-8	1.041ms	1.2-10 μ s	TTL	8	-	Output data
THRL	1.041ms	1.2-10 μ s	TTL	1	-	Load output data
MR	level		TTL	1	-	Master reset
DRR	level		TTL	1	-	Data received reset
SFD	level		TTL	1	-	Status flag disable
RRD	level		TTL	1	-	Receiver Register disable
RR1-8	1.041ms	1.041ms	TTL	8	-	Received data
PE	level		TTL	1	-	Parity error
FE	level		TTL	1	-	Frame error
OE	level		TTL	1	-	Over run error
DR	level		TTL	1	-	Data received flag
THRE	level		TTL	1	-	XMTR holding reg. empty
TRE	level		TTL	1	-	Transmitter register empty
CLOCK	1.041ms	1.041ms	TTL	1	-	Data rate clock

Table 5 INTERFACE ANALYSIS

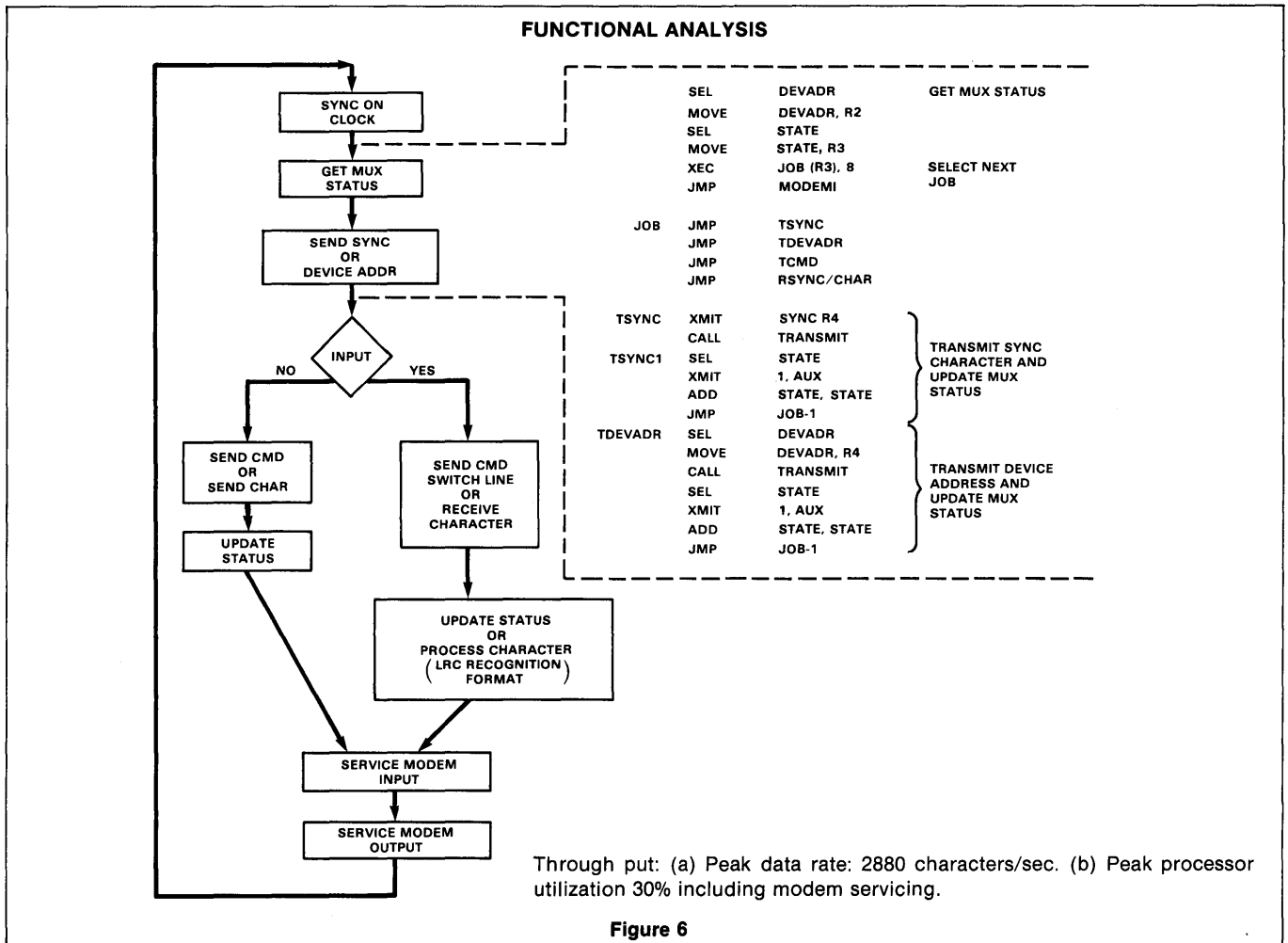


Figure 6

ROM/PROM FOR PROGRAM STORAGE		WORKING STORAGE FOR DATA BUFFERS	IV BYTES FOR INPUT/OUTPUT INTERFACE
Multiplexer driver	156 words	32 bytes	13 IV bits for output per UART
Character processing	100 words		15 IV bits for input per UART
Total	256 words		Total: 4 IV bytes per UART

Table 6 8X300 CONFIGURATION

REMOTE ALPHANUMERIC TERMINAL CONTROLLER

DESCRIPTION

The 8X300 interfaces to simple keyboard/display devices with a minimal amount of interface circuitry. The display may be buffered or the 8X300 system can supply buffering and refresh. In this example, the personality of the keyboard/display terminal is programmed into program storage to implement various editing and format functions. A single 8X300 can be used to control a local cluster of alphanumeric terminals since the processor utilization for a single terminal is very low. Messages to and from each terminal are transferred to a remote computer (interface not shown). Figure 7 illustrates the system.

DESIGN APPROACH

A terminal driver routing inputs and buffers messages in working storage. The driver also performs character and line deletion functions and implements a flicker free display of the message. A special set of control characters are used to terminate a message and forward the message. A four step procedure is followed to implement the design:

1. Analyze interface. Analyze keyboard and display relative to: Number of control and data lines, timing and data rates associated with each control/data line, electrical characteristics of each control/data line and determine supplemental circuits needed for electrical compatibility. Here the interfaces are completely compatible electrically (see Table 7).
2. Perform function analysis. The functions to be programmed and any which require supplemental logic are determined. In this case, no supplemental logic is required. The programmed functions are:
 - a. Store a message input from keyboard
 - b. Update display to produce flicker free output
 - c. Implement character delete, line delete editing functions
 - d. Recognize end of message control character.
3. Define the program to process input and to generate output (see Figure 8).
4. Determine the 8X300 configuration (see Table 8).

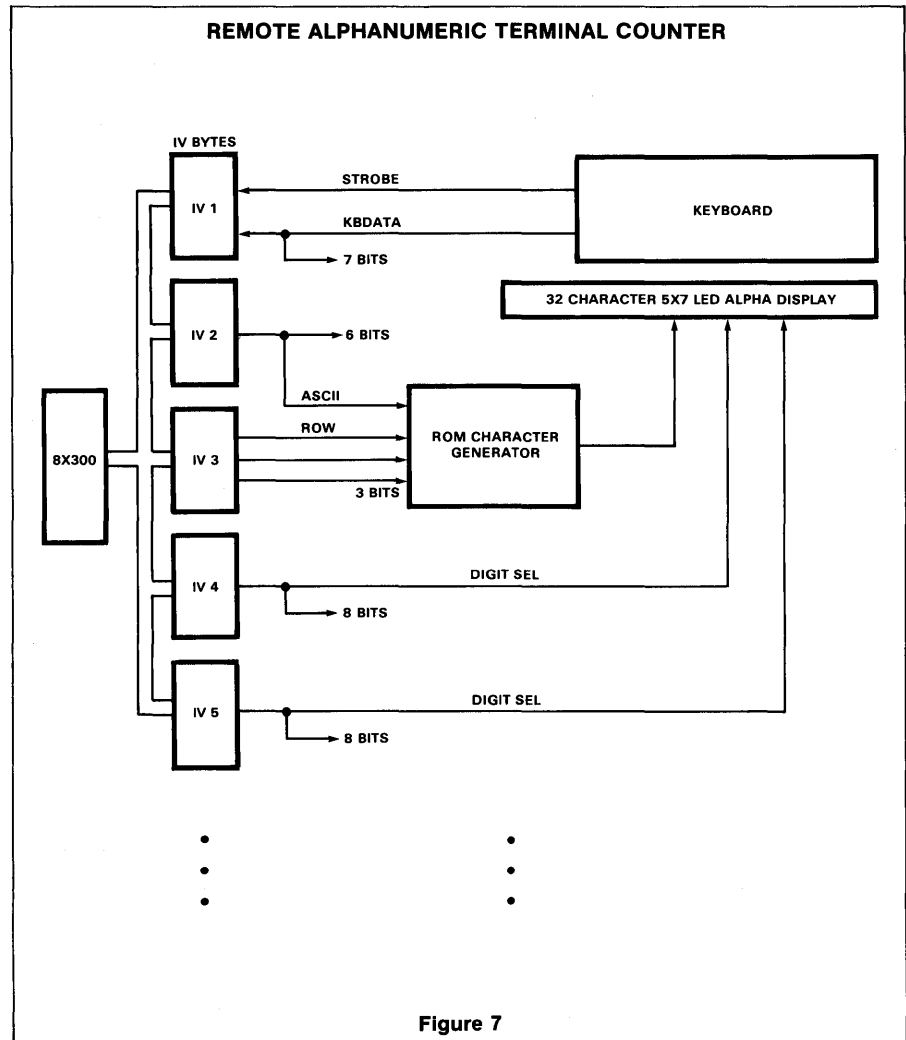
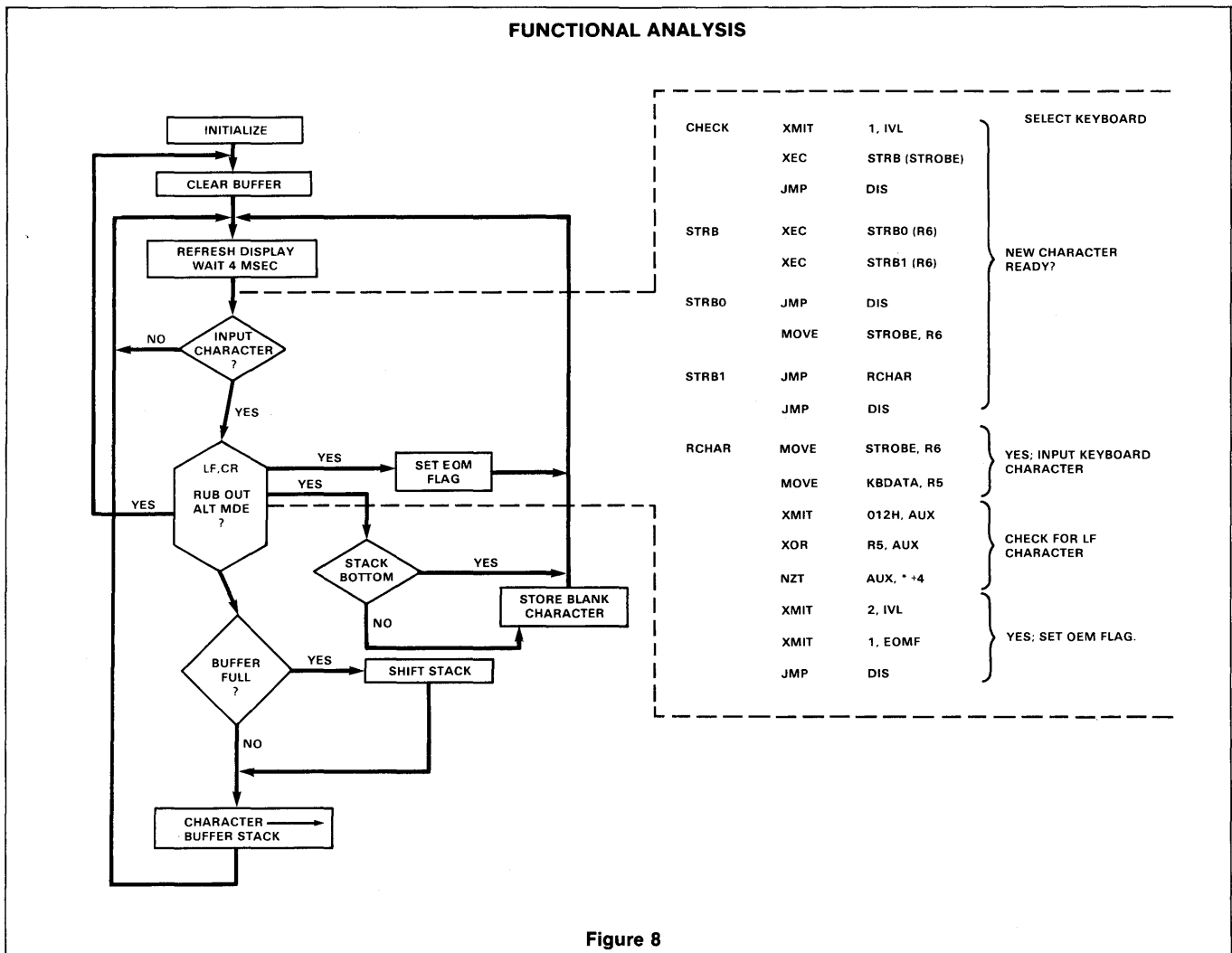


Figure 7

SIGNAL NAME	DATA RATE ¹	SIGNAL DURATION	ELECTRICAL CHARACTERISTICS	# IV BITS	INTERFACE REQUIRED	FUNCTION
STROBE	level	4msec (min)	TTL	1	-	Input Character ready
KBDATA	level	4msec (min)	TTL	7	-	Keyboard input character
ASCII	level	16.6msec (max) 200ns (min)	TTL	6	-	Select character
ROW	level		TTL	3	-	Select row of digit
DIGIT SEL	level		TTL	32	-	Select digit for display

Table 7 INTERFACE ANALYSIS



ROM/PROM FOR PROGRAM STORAGE	WORKING STORAGE FOR DATA BUFFERS	IV BYTES FOR INPUT/OUTPUT INTERFACE
Keyboard/driver 140 words	32 bytes per display	41 IV bits for output per display 8 IV bits for input per display Total: 7 IV bytes per display

Table 8 8X300 CONFIGURATION

COMPUTER I/O BUS EMULATOR

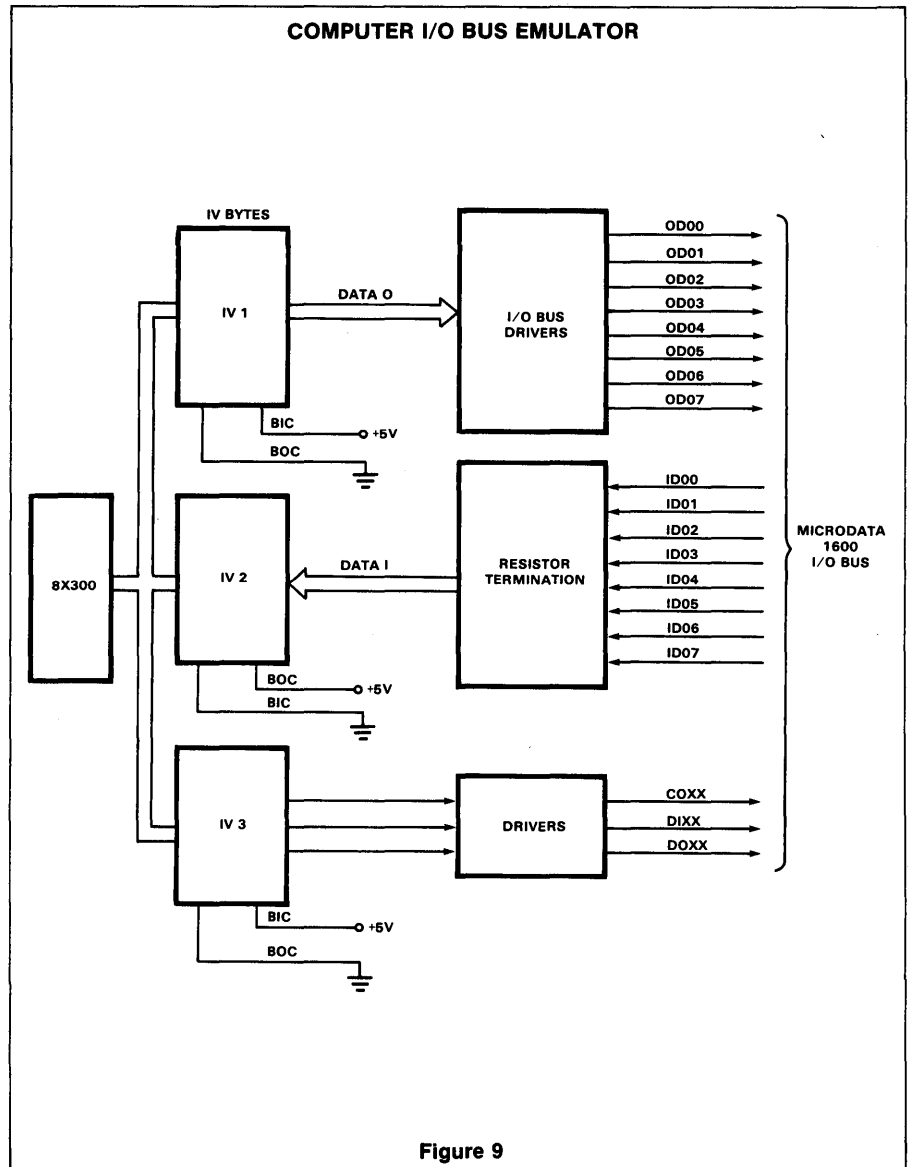
DESCRIPTION

The 8X300 system emulates a Microdata 1600 I/O bus. Microdata I/O bus compatible peripherals may then be easily connected to and controlled by a standard 8X300 system. A Microdata I/O bus driver program provides a standard software interface to peripheral devices and requires only 27 words of program storage. Figure 9 illustrates the system.

DESIGN APPROACH

Data bytes are transferred to and from the I/O bus in accordance with Microdata I/O bus specifications. Control signal timing and data transfer sequences are generated by programming. A four step procedure is followed to implement the design:

1. Analyze interface. Analyze Microdata I/O bus relative to: Number of control/data lines, timing and data rates associated with each control/data line, electrical characteristics of each control/data line and determine supplemental circuits needed for electrical compatibility (see Table 9).
2. Perform functional analysis. The functions to be programmed and any which require supplemental logic are determined. In this case, no supplemental logic is required. The programmed functions are:
 - a. Transfer bytes in and out
 - b. Generate control signal timing and data transfer sequences
3. Define the program to process input and to generate output (see Figure 10).
4. Determine the 8X300 configuration (see Table 10).



SIGNAL NAME	DATA RATE ¹	SIGNAL DURATION	ELECTRICAL CHARACTERISTICS	# IV BITS	INTERFACE REQUIRED	FUNCTION
OD00-07	level		open collector	8	8D	Data/address from computer
ID00-07	level		TTL	8	8R	Data to computer
COXX	4μs	1.25μs	open collector	1	D	Control output timing
DIXX	4μs	1.25μs	open collector	1	D	Data input timing
DOXX	4μs	.75-1.25μs	open collector	1	D	Data output timing

D = Open collector driver
R = Resistors

Table 9 INTERFACE ANALYSIS

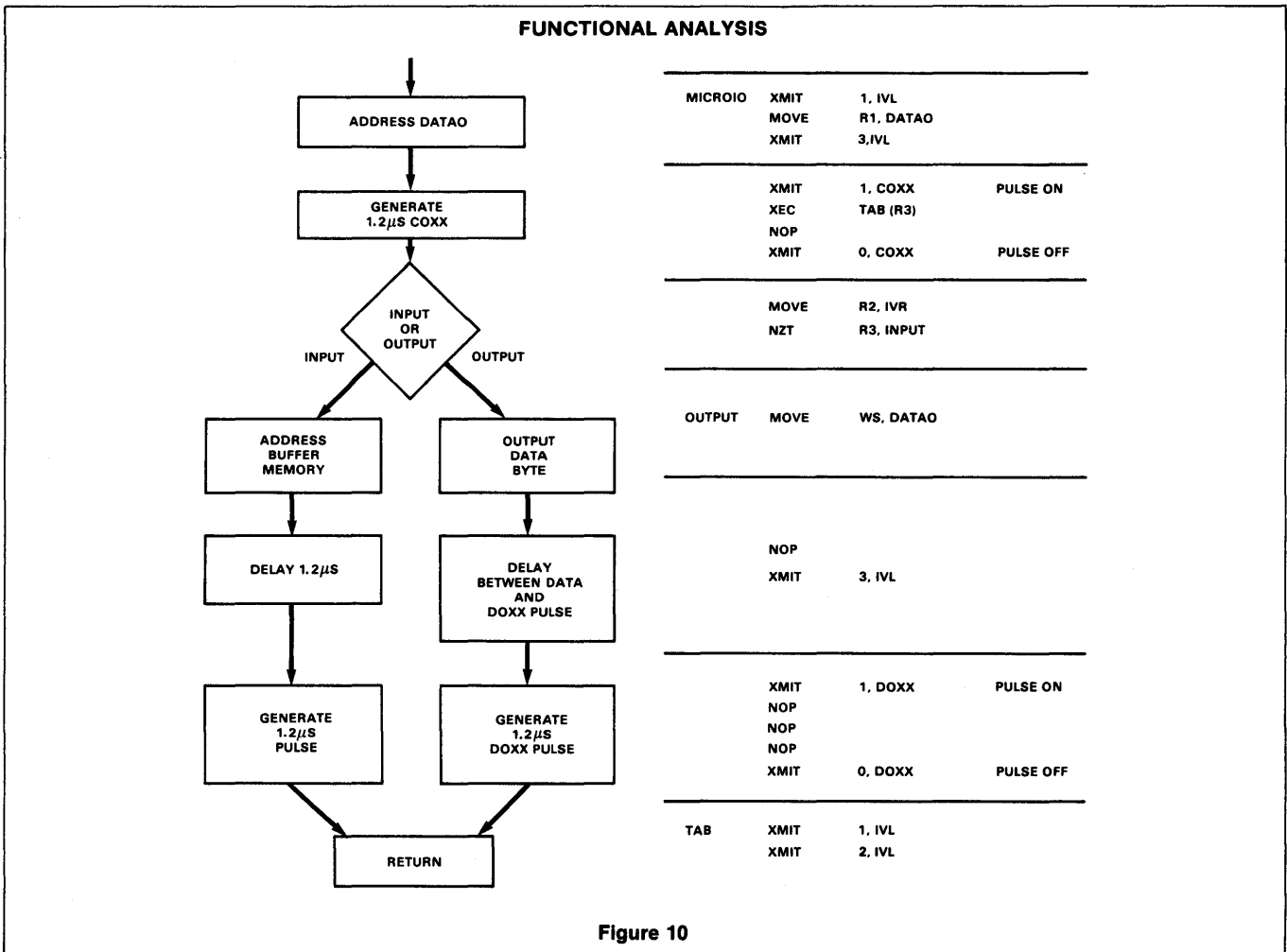


Figure 10

ROM/PROM FOR PROGRAM STORAGE	WORKING STORAGE FOR DATA BUFFERS	IV BYES FOR INPUT/OUTPUT INTERFACE
I/O Driver 27 words	Depends on peripheral	11 IV bits for output 8 IV bits for input Total: 3 IV bytes per peripheral

Table 10 8X300 CONFIGURATION

INTERFACE TO EXTERNAL READ/WRITE MEMORY

DESCRIPTION

The 8X300 controls the storage, retrieval and processing of large blocks of data. Data is stored in a large capacity (up to 64K bytes) read/write RAM external to the 8X300 system. The memory is assembled from widely available n-channel (n-MOS) static or dynamic RAM circuits. Minimal interface circuitry is required to connect the 8X300 Interface Vector bytes to the address, data and control lines of the external memory. Figure 11 illustrates the system.

DESIGN APPROACH

Data bytes are read from or written into memory through a single IV type. Two additional IV bytes are used as a 16-bit address register to the external memory. 16 bits provide an address range of 65K bytes. The read/write control signals to the memory require two IV bits. Instruction sequences are used for memory read and memory write operations to implement 1 to 2 microsecond memory access times. A four step procedure is followed to implement the design:

1. Analyze interface. Analyze n-MOS RAM circuits relative to: Number of control/data lines, timing and data rates associated with each control/data line, electrical characteristics of each control/data line and determine any supplemental circuits needed for electrical compatibility (see Table 11).
2. Perform functional analysis. The functions to be programmed and any which require supplemental logic are determined. The programmed functions are:
 - a. Store memory address in IV bytes ADRHI, ADRLO.
 - b. Set appropriate read/write control bits
 - c. Wait for memory operation complete
3. Define the program to process input and to generate output.
 - a. GET instruction sequence to read memory location addressed by contents of IV bytes ADRHI, ADRLO (see Figure 12).
 - b. PUT instruction sequence to write data into the memory location addressed by the contents of IV bytes ADRHI, ADRLO (see Figure 13).
4. Determine the 8X300 configuration (see Table 12).

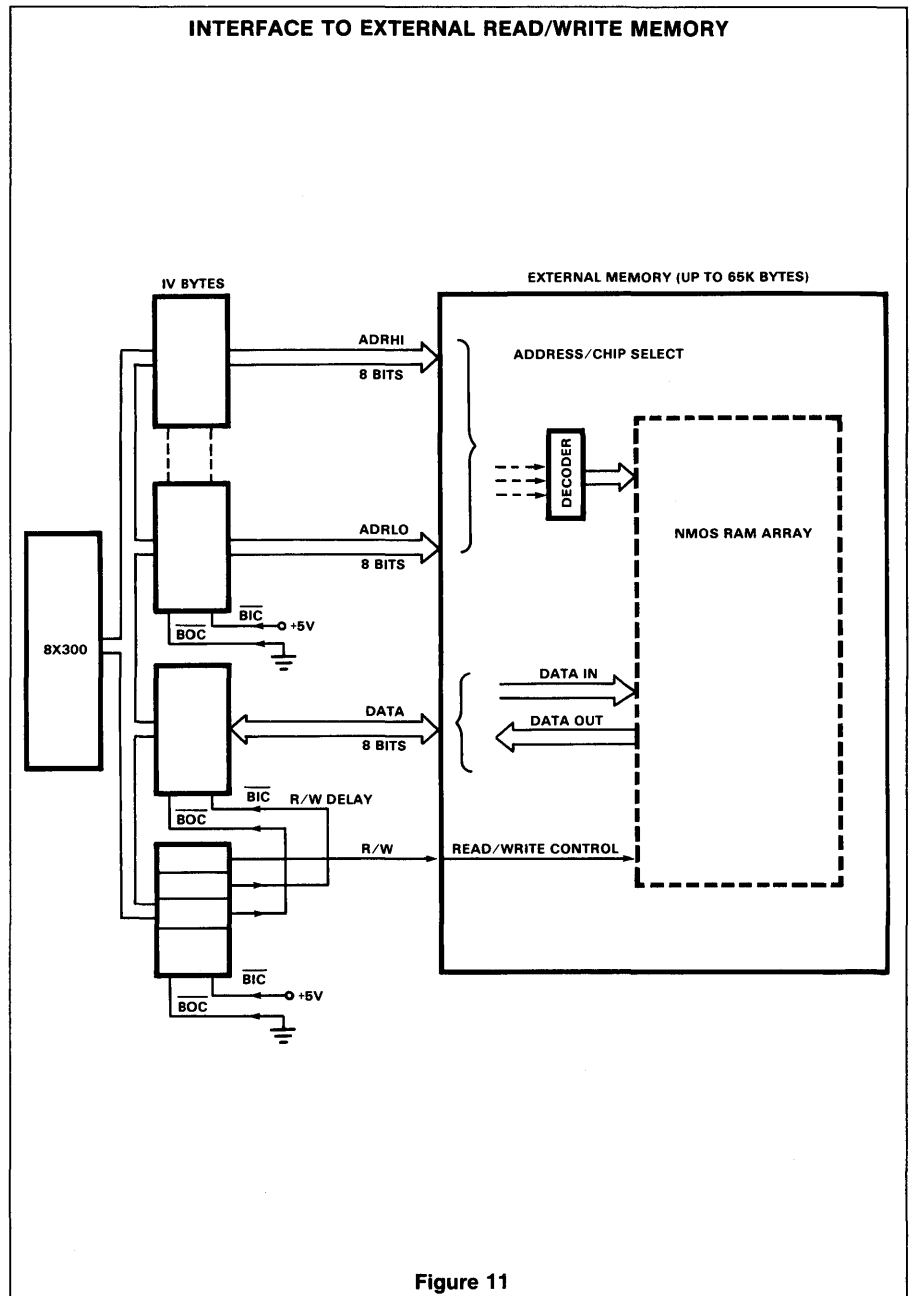


Figure 11

SIGNAL NAME	DATA RATE ⁻¹	SIGNAL DURATION	ELECTRICAL CHARACTERISTICS	# IV BITS	INTERFACE REQUIRED	FUNCTION
ADRHI	Level		TTL	8	* none	Most significant byte. Memory address, and chip select input
ADRLO	Level		TTL	8	* none	Least significant byte memory address
DATA	Level		TTL	8	* none	Memory data
R/W	500ns (min)	>250ns	TTL	1	* none	Memory read/write control
R/W DELAY	500ns (min)	>500ns	TTL	1	* none	Data enable delay during memory write

Table 11 INTERFACE ANALYSIS

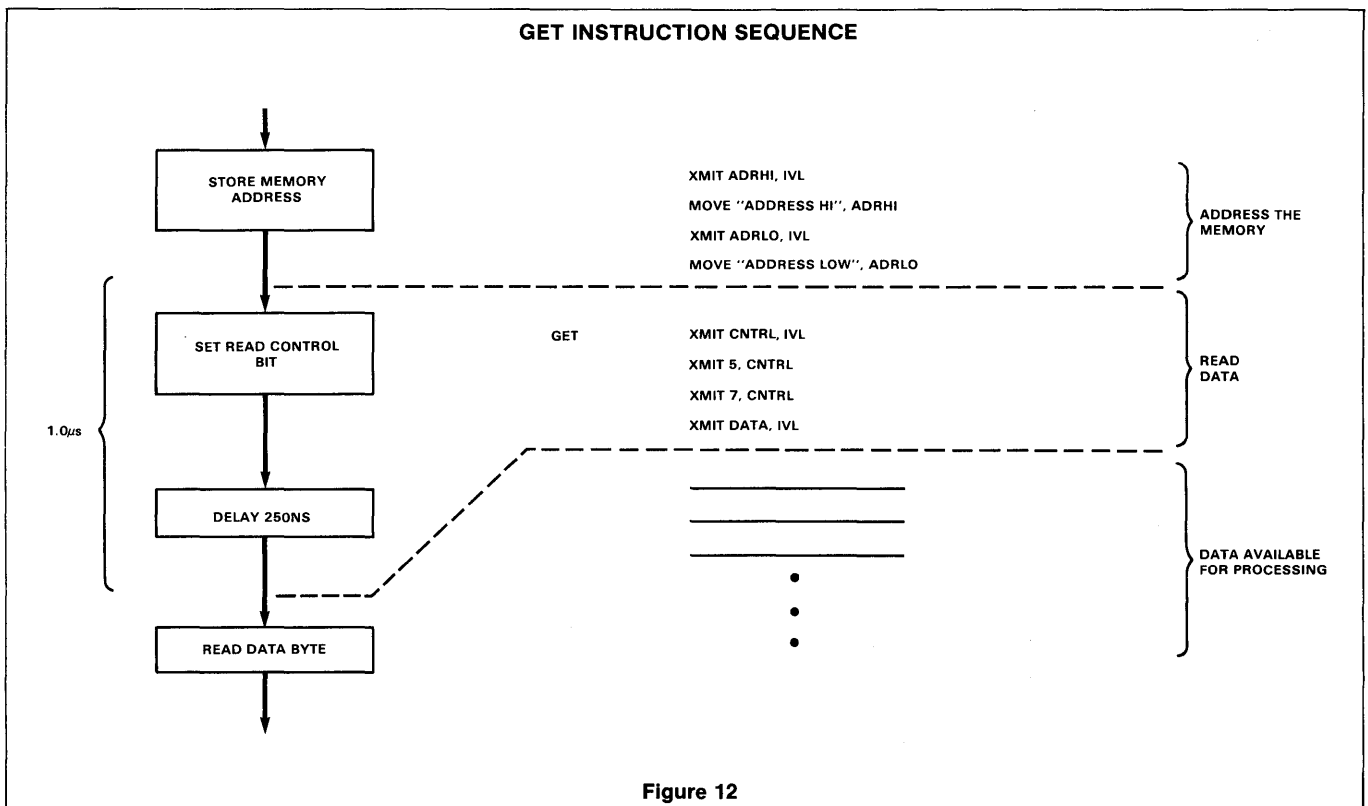


Figure 12

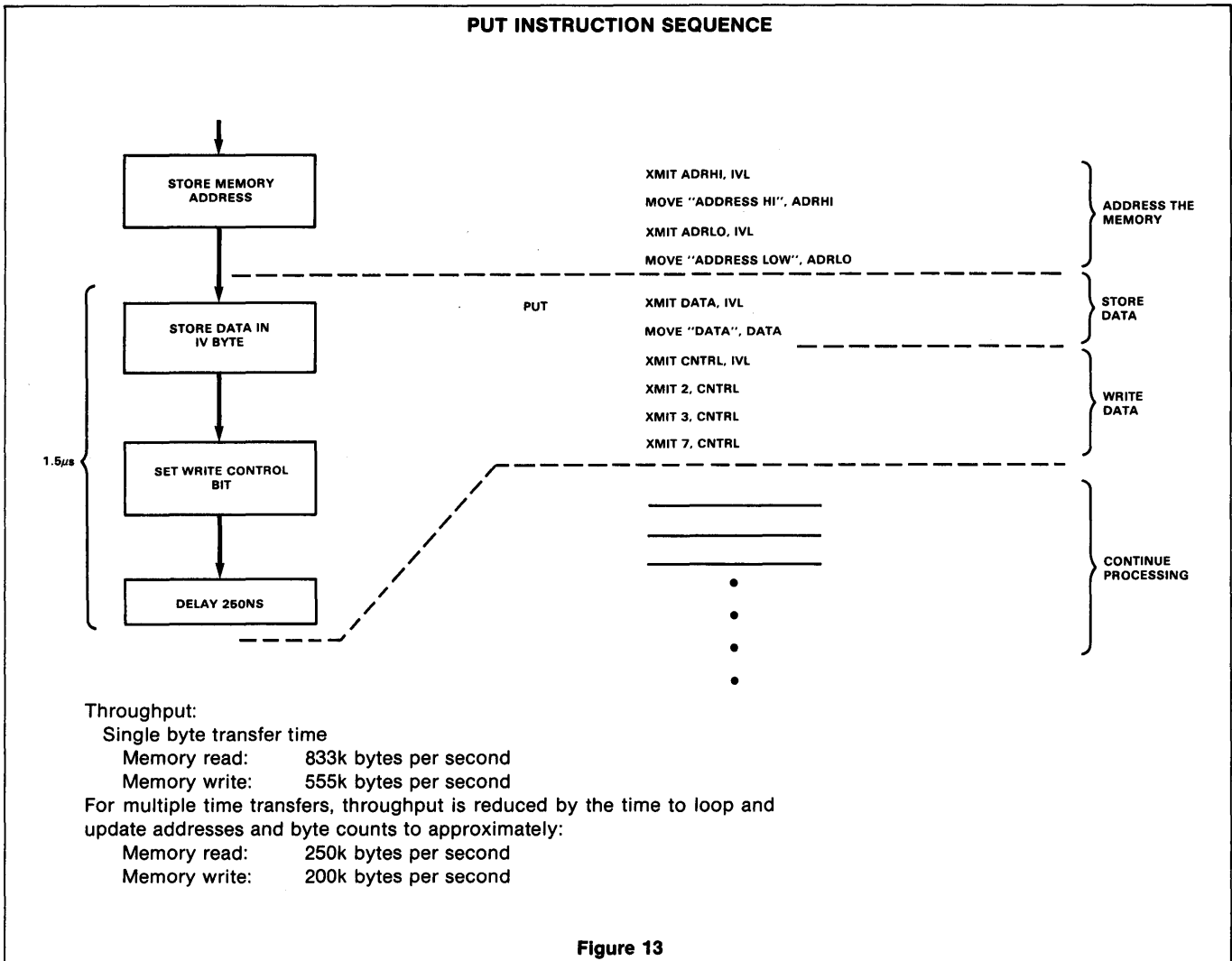


Figure 13

ROM/PROM FOR PROGRAM STORAGE		WORKING STORAGE	IV BYTES FOR INPUT/OUTPUT INTERFACE
GET sequence	4 words	None	18 IV bits for output 8 IV bits for input and output Total: 4 IV bytes
PUT sequence	6 words		

Table 12 8X300 CONFIGURATION

256 WAY BRANCH

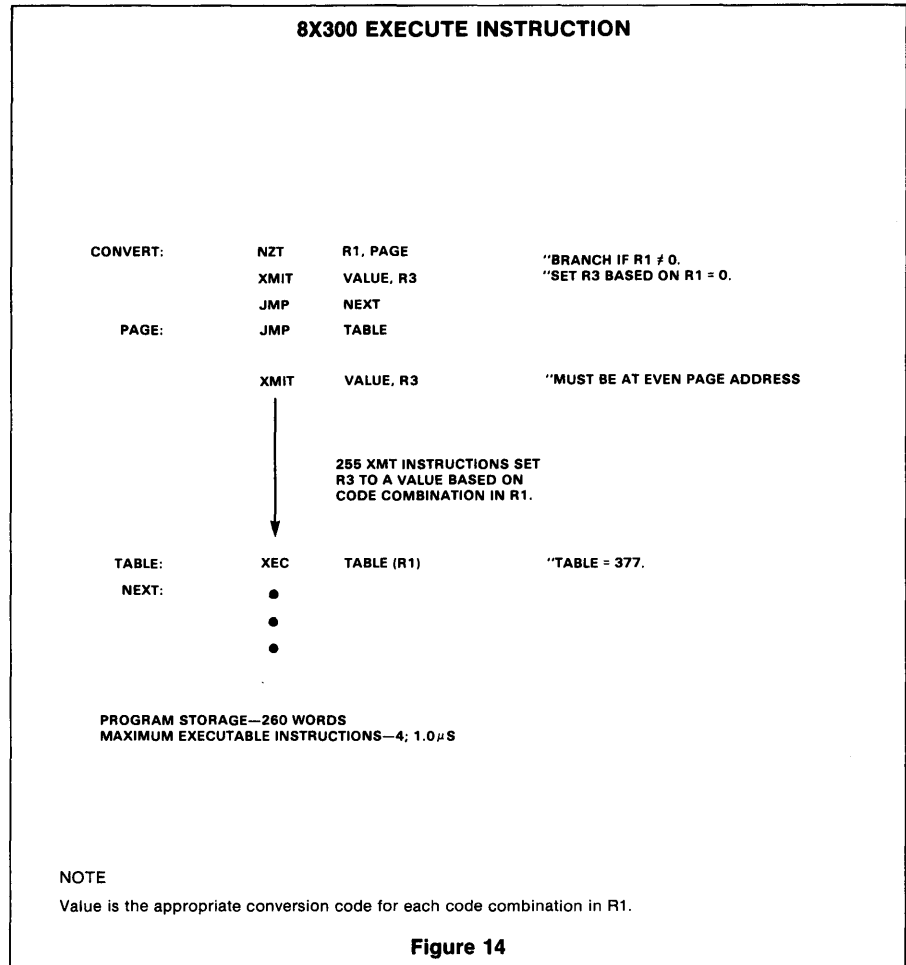
DESCRIPTION

Many data communication applications require conversion of one code structure to another. The 8X300's Execute instruction provides a fast and efficient method of performing this conversion.

A single Execute instruction can provide up to a 255 way branch based on a byte stored in a register.

This assumes one of the 256 values does not occur during operation of the Execute table. This is easily prevented by testing for one of the values before entering the table, thereby completing the 256 way branch. The example in Figure 14 details how the test for R1 equal to zero is performed first (NZT). If zero, the appropriate conversion value is loaded into R3 (XMIT). If not zero, then the Execute table determines which of the other 255 combinations is in R1 and loads the appropriate conversion value in R3.

The 256 way branch requires 260 words of program storage and 1.0 microseconds maximum to execute. The Execute table and the Execute instruction must all be located with one 256 byte page where the first instruction address contains zeros in the 8 least significant bits. The other four instructions may be placed anywhere within the 8X300's address space.



FAST IV SELECT

DESCRIPTION

The fast IV select is implemented by adding bits to the instruction word, in increments of 4 or 8 bits. This technique allows IV bytes and working storage to be selected within the same instruction where it is used. This can save important processor time by saving one instruction cycle for each select instruction. It eliminates the need for the IV select instruction. It trades fewer instruction cycle times for hardware. It also trades 16-bit select instructions for 4 to 8-bit select fields, thus saving 8 to 12 bits of program storage for every select instruction saved. To some extent, this reduces the cost impact of a larger instruction word. The technique can be used on both IV and buffer storage (including working storage). When used on IV, a decoder is used following an address hold latch to select one IV per address combination. Buffer storage does not require the decoder, instead it utilizes the address directly.

The fast select IV can be used on the same system with normal select IV since all the fast select IV contains the same address. The Master Enable (ME) input of each fast select IV is enabled by the AND of Bank Select (LB, RB) and the single line decode.

Due to memory access delays, the clock used to latch the fast select address is delayed with a couple of inverter delays to assure address validity. On large systems, there are extra delays which may require the address to be programmed in the instruction prior to its usage. Then a double set of address hold latches are used so the address will appear sufficiently early.

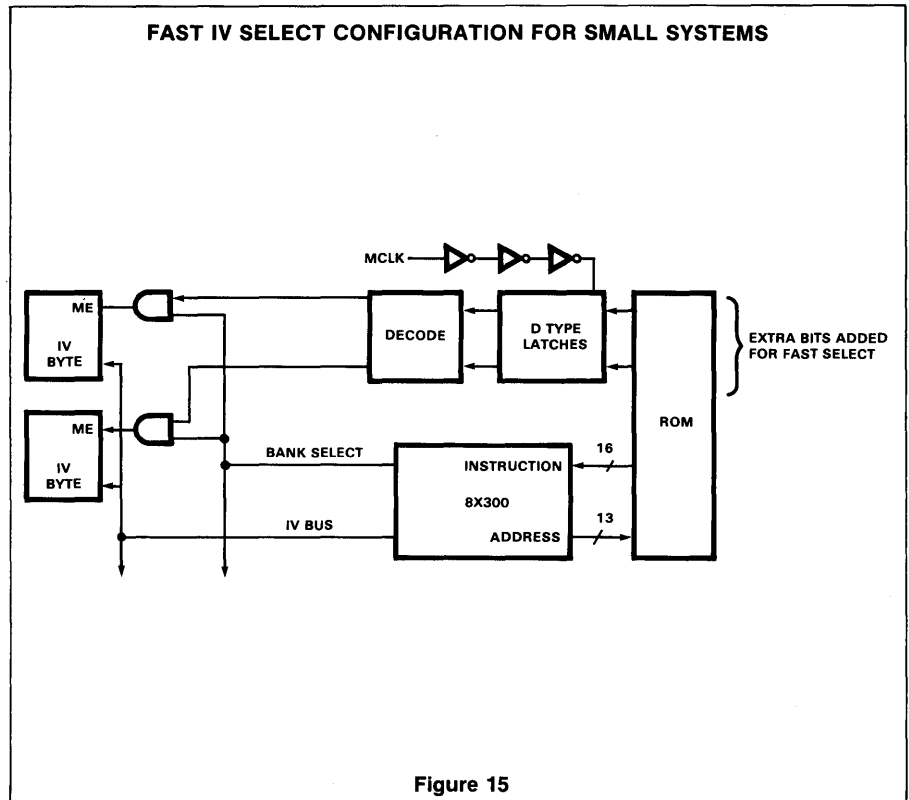


Figure 15

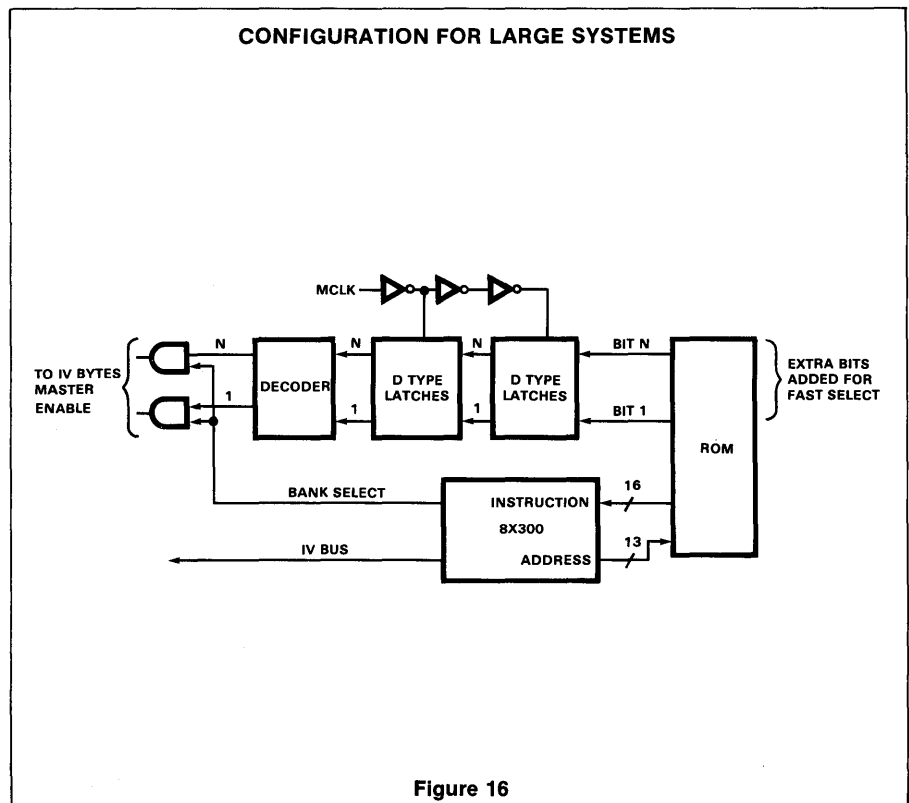
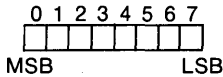


Figure 16

DESCRIPTION

The 8X300 has a repertoire of 8 instruction classes which allow the user to test input status lines, set or reset output control lines, and perform high speed input/output data transfers. All instructions are 16 bits in length. Each instruction is fetched, decoded and executed completely in 250ns.

Data is represented as an 8-bit byte; bit positions are numbered from left to right, with the least significant bit in position 7.



Within the Interpreter, all operations are performed on 8-bit bytes. The Interpreter performs 8-bit, unsigned 2's complement arithmetic.

INSTRUCTION FORMATS

The general 8X300 instruction format is:

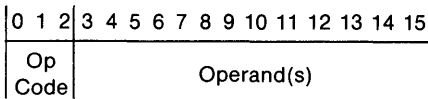


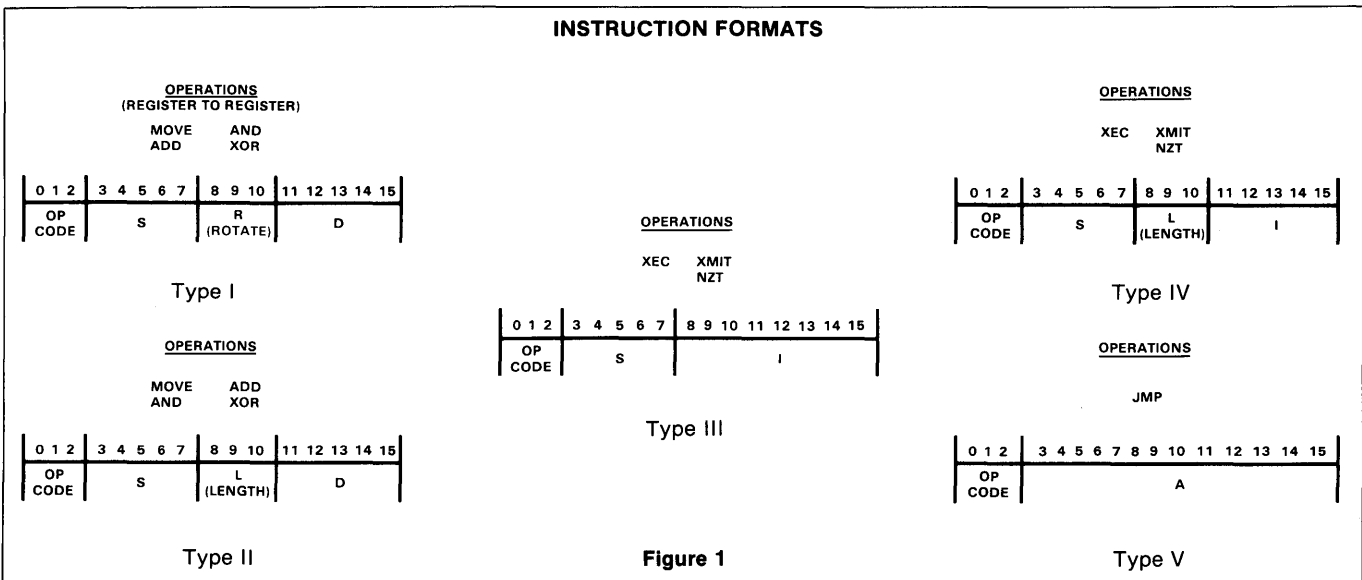
Table 1 contains a summary of the 8X300 instruction set and description of the operand fields.

All instructions are specified by a 3-bit Operation (Op) Code field. The operand may consist of the following fields: Source (S) field, Destination (D) field, Rotate/Length (R/L) field, Immediate (I) Operand field, and (Program Storage) Address (A) field.

The instructions are divided into 5 format types based on the Op Code and the form of the Operand(s) as shown in Figure 1.

OPERATION	FORMAT	RESULT	NOTES
MOVE		Content of data field addressed by S, R/L replaces data in field specified by D, R/L.	If S and D both are register addresses then R L specifies a right rotate of R/L places applied to the register specified by S.
ADD		Sum of AUX and data specified by S, R/L replaces data in field specified by D, R/L.	
AND		Logical AND of AUX and data specified by S, R/L replaces data in field specified by D, R/L.	
XOR		Logical exclusive OR of AUX and data specified by S, R/L replaces data in field specified by D, R/L.	
XMIT		The literal value I replaces the data in the field specified by S, L.	
NZT		If the data in the field specified by S, L equals zero, perform the next instruction in sequence. If the data specified by S,L is not equal to zero, execute the instruction at address determined by using the literal I as an offset to the Program Counter.	If S specifies an IV or WS address then I is limited to the range 00-377. I is limited to the range 000-377 otherwise.
XEC		Perform the instruction at address determined by applying the sum of the literal I and the data specified by S,L as an offset to the Program Counter. If that instruction does not transfer control, the program sequence will continue from the XEC instruction location.	The offset operation is performed by reducing the value of PC to the nearest multiple of 32 (if I : 00-37) or 256 (if I : 000-377) and adding the offset.
JMP		The literal value I replaces contents of the Program Counter.	I limited to the range 00000-07777.

Table 1 8X300 INSTRUCTION SUMMARY



INSTRUCTION FIELDS

Op Code Field (3-Bit Field)

The Op Code field is used to specify 1 of 8 8X300 instructions.

S,D Fields (5-Bit Fields)

The S and D fields specify the source and destination of data for the operation defined by the Op Code field. The Auxiliary Register is the implied source for the instructions ADD, AND and XOR which require two source fields. That is, instructions of the form:

ADD X, Y

imply a third operand, say Z, located in the Auxiliary Register so that the operation which takes place is actually X + Z, with the result stored in Y. This powerful capability means that 3 operands are referenced in 250ns.

The S and/or D fields may specify a register, or a 1 to 8-bit I/O field, or a 1 to 8-bit Working Storage field. S and D field value assignments in octal are shown in Table 2.

R/L Field (3-Bit Field)

The R/L field performs one of two functions, specifying either a field length (L) or a right rotation (R). The function it specifies for a given instruction depends upon the contents of the S and D fields:

- A. When both S and D specify registers, the R/L field is used to specify a right rotation of the data specified by the S field. (Rotation occurs on the bus and not in the source register.) The register source data is right rotated within one instruction cycle time independent of the number (0 to 7) of bit positions specified in the R/L field.
- B. When either or both the S and D fields specify an IV or Working Storage data field, the R/L field is used to specify the length of the data field (within the byte) accessed, as shown in Figure 2.

I Field (5/8-Bit Field)

The I field is used to load a literal value (a binary value contained in the instruction into a register, IV or Working Storage data field or to modify the low order bits of the Program Counter.

The length of the I field is based on the S field in XEC, NZT, and XMIT instructions.

- A. When S specifies a register, the literal I is an 8-bit field (Type III format).
- B. When S specifies an IV or Working Storage data field, the literal I is a 5-bit field (Type IV format).

A Field (13-Bit Field)

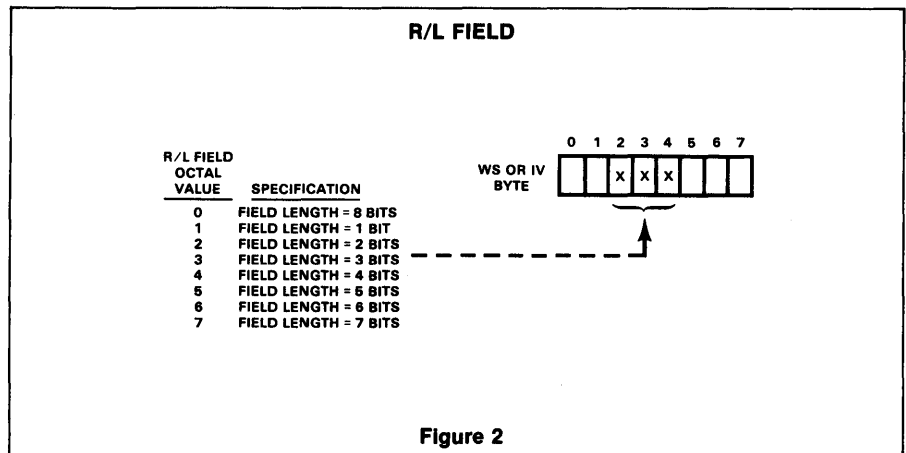
The A field is a 13-bit Program Storage address field. This allows the 8X300 to directly address 8192 instructions.

REGISTER OPERATIONS

When a register is specified as the source, and an IV or Working Storage field as the destination, the least significant bits of the operations (MOVE, ADD, AND, XOR) are merged with the original destination data. The least significant bits of the result are stored in the IV or Working Storage data field specified by the D and R/L fields in the instruction.

OP CODE OCTAL VALUE	INSTRUCTION	RESULT
0	MOVE S,R/L,D	(S) → D
1	ADD S,R/L,D	(S) plus (AUX) → D
2	AND S,R/L,D	(S) ^ (AUX) → D
3	XOR S,R/L,D	(S) ⊕ (AUX) → D
4	XEC I,R/L,S or I,S	Execute instruction at current PC offset by I + (S)
5	NZT I,R/L,S or I,S	Jump to current PC offset by I if (S) ≠ 0
6	XMIT I,R/L,D or I,D	Transmit literal I → D
7	JMP A	Jump to program location A

Table 2 S AND D FIELD VALUE ASSIGNMENTS IN OCTAL



When an IV or Working Storage field of 1 to 8 bits is specified as the source, and a register as the destination, the 8-bit result of the operations (MOVE, ADD, AND, XOR) is stored in the register. The operations ADD, AND, XOR actually use the IV or Working Storage data field (1 to 8 bits) with leading zeros to obtain 8-bit source data for use with the 8-bit AUX data during the operation.

Because IVL and IVR are write-only pseudo registers, they can be specified as destination fields only (see Table 3). Operations involving IVL and IVR as sources are not possible. For example, it is not possible to increment IVR or IVL in a single instruction, and the contents of IVL or IVR cannot be transferred to a working register, IV byte, or Working Storage location.

The OVF (Overflow) Register can only be used as a source field; it is set or reset only by the ADD instruction.

INSTRUCTION DESCRIPTIONS

The following instruction descriptions employ MCCAP (the 8X300 Cross Assembly Program) programming notation. This notation varies somewhat from the instruction descriptions provided in Tables 1 and 3. Thus, for example, explicit L field definition, as shown in Table 1 and Table 3, is not required by MCCAP instructions; MCCAP creates appropriate variable field addresses from the information contained in the Data Declaration statements provided by the programmer at the beginning of his program.

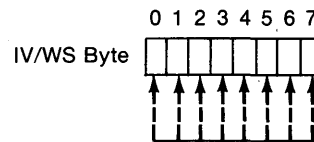
The 8X300 instruction set is described below with examples shown in Figures 3 through 10.

0₈-17₈ is used to specify 1 of 7 working registers (R1-R6, R11), Auxiliary Register, Overflow Register, IVL and IVR write-only registers.

OCTAL VALUE		OCTAL VALUE	
00	Auxiliary Register (AUX)	10	OVF-Overflow register-Used as an S (source) field only.
01	R1	11	R11
02	R2	12	Unassigned
03	R3	13	Unassigned
04	R4	14	Unassigned
05	R5	15	Unassigned
06	R6	16	Unassigned
07	IVL Register-IV Byte address write-only register-Specified only in D field in all instructions	17	IVR Register-Working Storage address write-only register-Specified only in D field in all instructions

a. Register Specification

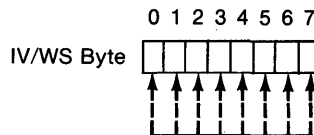
20₈-27₈ is used to specify the least significant bit of a variable length field within the IV/WS Byte previously selected by the IVL register. The length of the field is determined by R/L.



OCTAL VALUE	
20	Field within previously selected IV/WS Byte; position of LSB = 0
21	Field within previously selected IV/WS Byte; position of LSB = 1
22	Field within previously selected IV/WS Byte; position of LSB = 2
23	Field within previously selected IV/WS Byte; position of LSB = 3
24	Field within previously selected IV/WS Byte; position of LSB = 4
25	Field within previously selected IV/WS Byte; position of LSB = 5
26	Field within previously selected IV/WS Byte; position of LSB = 6
27	Field within previously selected IV/WS Byte; position of LSB = 7

b. Left Bank Field Specification

30₈-37₈ is used to specify the least significant bit of a variable length field within the IV/WS Byte previously selected by the IVR Register. The length of the field is determined by R/L.



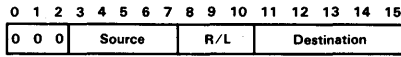
OCTAL VALUE	
30	Field within previously selected IV/WS Byte; position of LSB = 0
31	Field within previously selected IV/WS Byte; position of LSB = 1
32	Field within previously selected IV/WS Byte; position of LSB = 2
33	Field within previously selected IV/WS Byte; position of LSB = 3
34	Field within previously selected IV/WS Byte; position of LSB = 4
35	Field within previously selected IV/WS Byte; position of LSB = 5
36	Field within previously selected IV/WS Byte; position of LSB = 6
37	Field within previously selected IV/WS Byte; position of LSB = 7

c. Right Bank Field Specification

Table 3 S AND D FIELD SPECIFICATIONS

**MOVE S,D or
MOVE S(R),D**

Format: Type I, Type II



**Operation: (S)→(D)
Description**

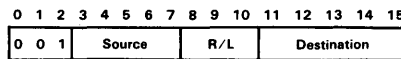
Move data. The contents of S are transferred to D; the contents of S are unaffected. If both S and D are registers, R/L specifies a right rotate of the source data before the move. Otherwise, R/L is implicit and specifies the length of the source and/or destination IV/WS field. If the MOVE is between an IV byte and a Working Storage byte, an 8-bit field is always moved.

Example

Store the least significant 3 bits of register 5 (R5) in bits 4, 5 and 6 of the IV byte previously addressed by the IVL register.

**ADD S,D or
ADD S(R),D**

Format: Type I, Type II



Operation

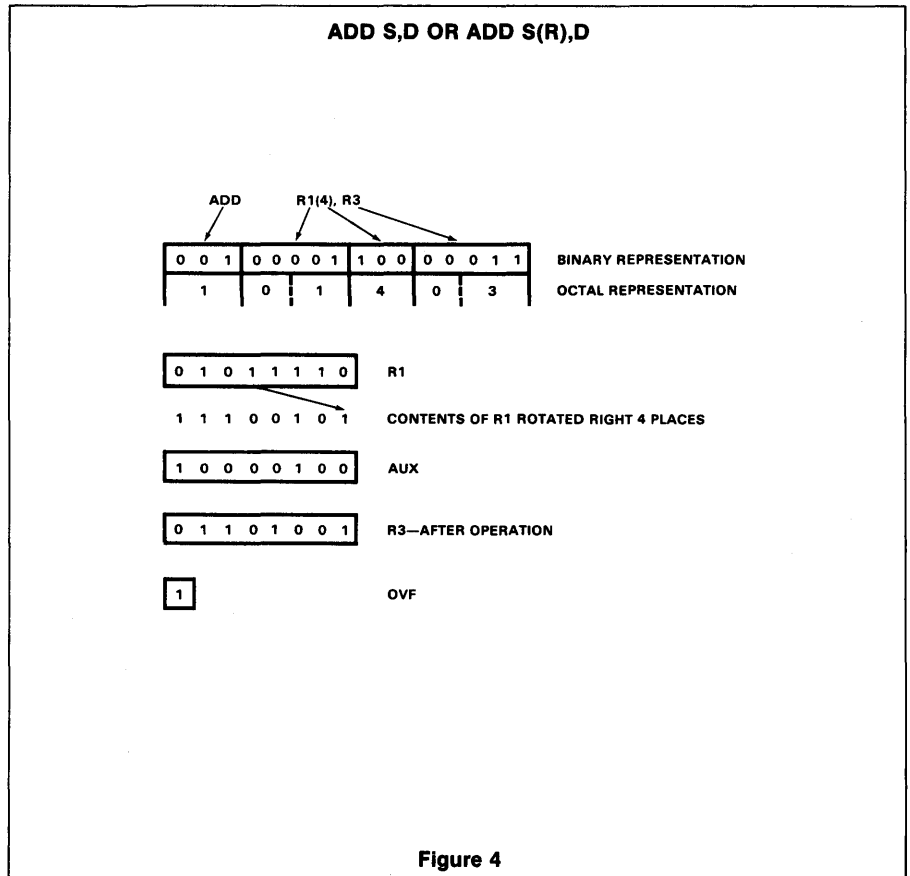
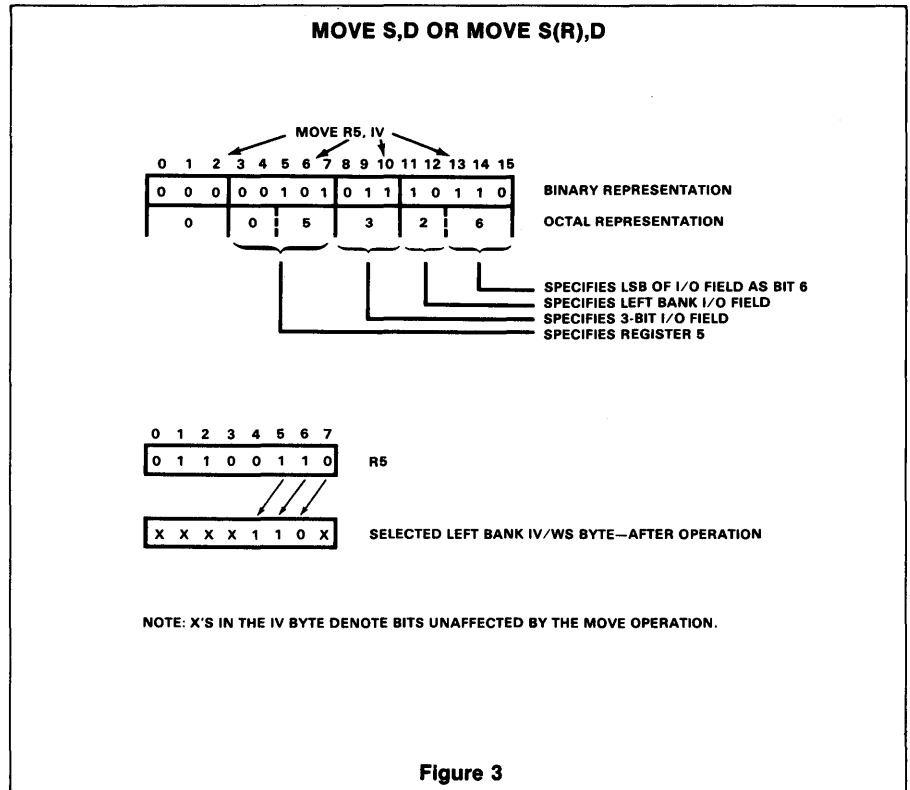
(S) plus (AUX) D; set OVF if carry from most significant bit occurs.

Description

Unsigned 2's complement addition. The contents of S are added to the contents of the Auxiliary Register (which is the implied source). The result is stored in D; OVF is updated. If both S and D are registers, R/L specifies a right rotate of the source (S) data before the operation. Otherwise, R/L is implicit and specifies the length of the source and/or destination IV/WS fields. S and AUX are unaffected unless specified as the destination.

Example

Add the contents of R1 (rotated 4 places) to AUX and store the result in R3.



**AND S,D OR
AND S(R),D**

Format: Type I, Type II

Operation: (S) \wedge (AUX) \rightarrow D

Description

Logical AND. The AND of the source field and the Auxiliary Register is stored into the destination. If both S and D are registers, R/L specifies a right rotate of the source (S) data before the AND operation. Otherwise R/L is implicit and specifies the length of the source and/or destination IV/WS fields. S and AUX are unaffected unless specified as a destination.

Example

Store the AND of the selected right bank byte and AUX in R4. The right bank data field is called WSBCD and is 4 bits long and located in bits 2, 3, 4 and 5.

**XOR S,D OR
XOR S(R),D**

Format: Type I, Type II

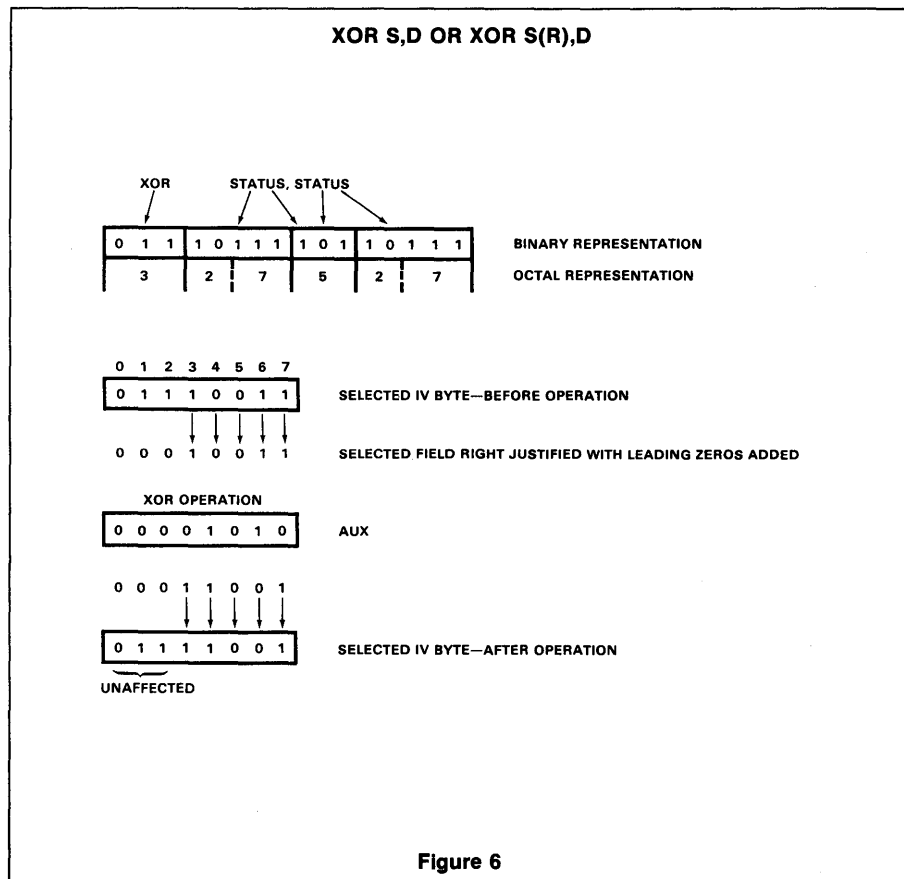
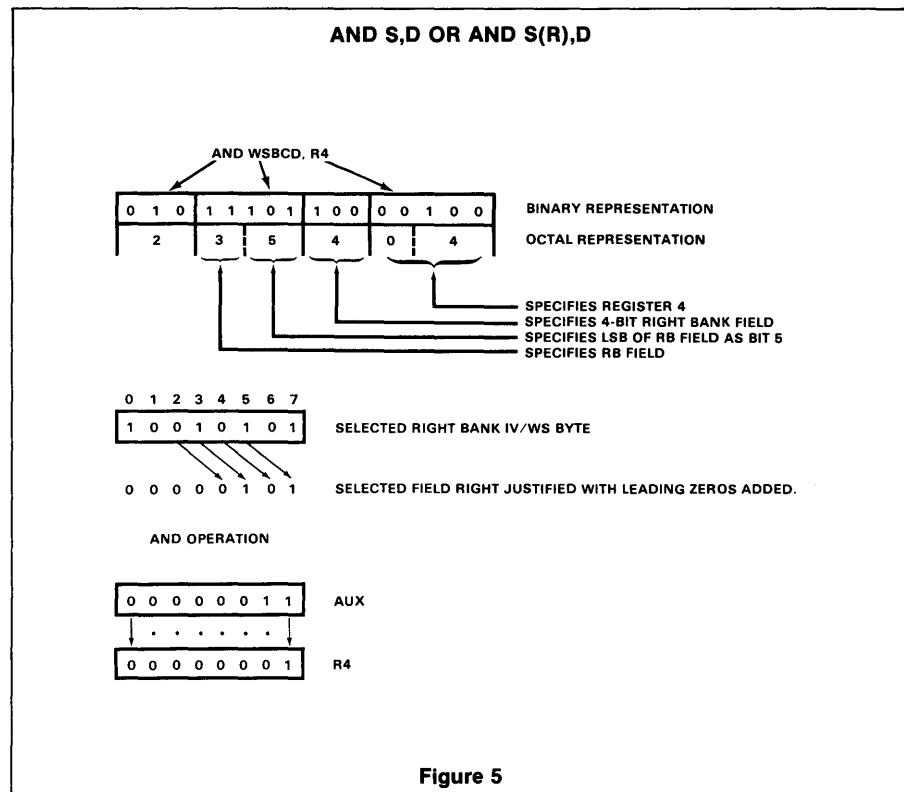
Operation: (S) \oplus (AUX) \rightarrow D

Description

Exclusive OR. The exclusive OR of the source field and the Auxiliary Register is stored in the destination. If both S and D are registers, R/L specifies a right rotate of the source (S) data before the XOR operation. Otherwise R/L is implicit and specifies the length of the source and/or destination IV/WS fields. S and AUX are unaffected unless specified as a destination.

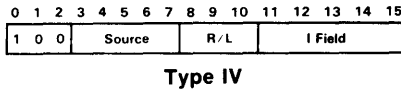
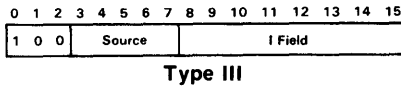
Example

Replace the selected IV byte field with the XOR of the field and AUX. The IV byte field is called STATUS and is 5 bits in length and located in bits 3, 4, 5, 6 and 7 of LB.



XEC I(S)

Format



Operation

Execute instruction at the address specified by the Address Register with lower 5/8 bits replaced by (S) + I.

Description

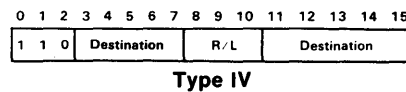
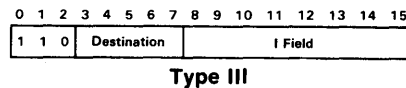
Execute the instruction at the address determined by replacing the low order bits of the Address Register (AR) with the low order bits of the sum of the literal I and the contents of the source field. If S is a register, the low order 8 bits of AR are replaced; if S is an IV or Working Storage field, the low order 5 bits of AR are replaced, resulting in an execute range of 256 and 32 respectively. The Program Counter is not affected unless the instruction executed is a JMP or NZT (whose branch is taken).

Example

Execute one of n JMPs in a table of JMP instructions determined by the value of the selected IV byte field. The table follows immediately after the XEC instruction and the IV field is called INTERPT and is a 3-bit field located in bits 4, 5 and 6.

XMIT I,D

Format



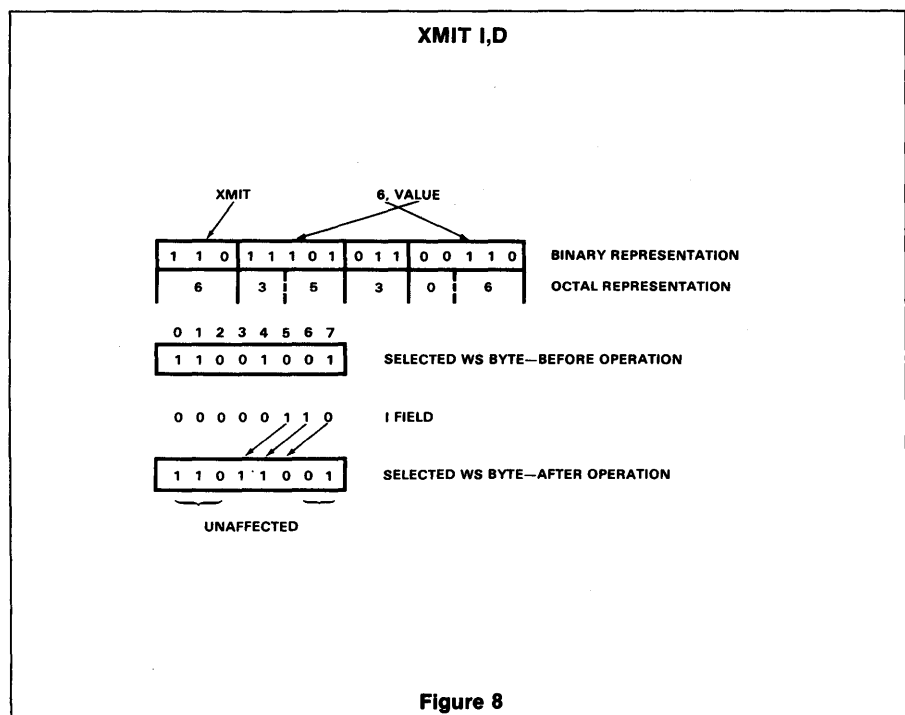
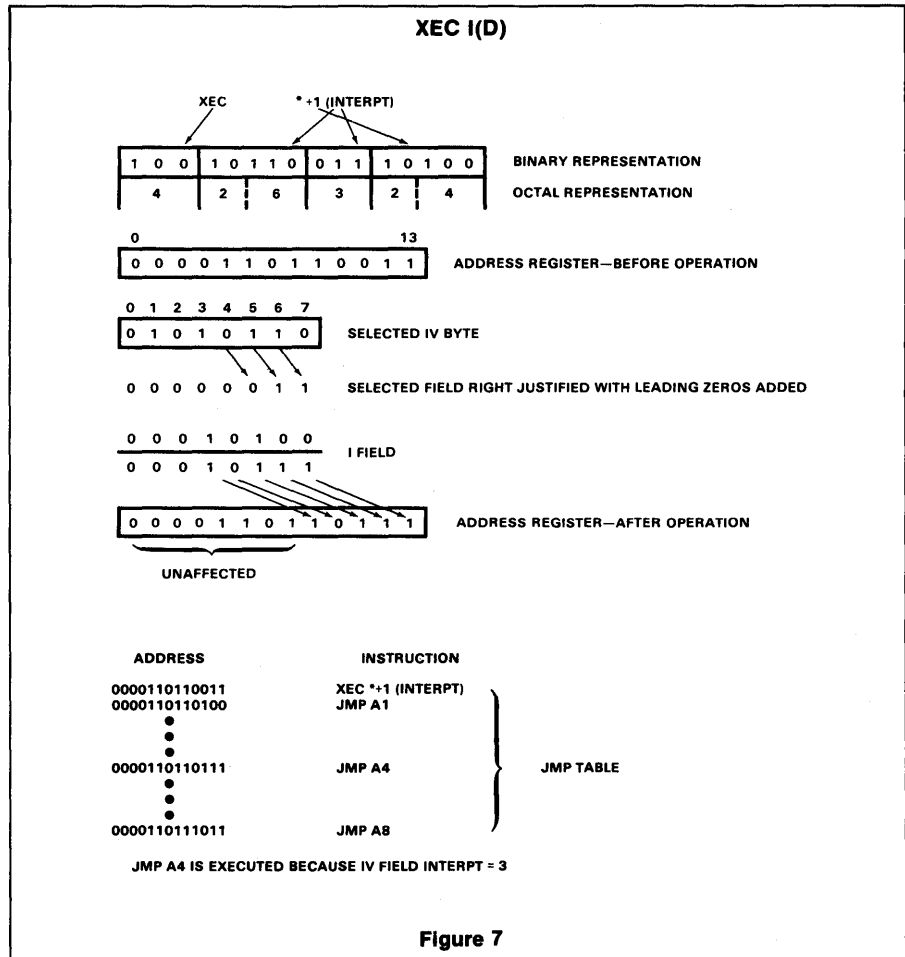
Operation: I → (D)

Description

Transmit literal. The literal field I is stored in D. If D is a register, an 8-bit field is transferred; if D is an IV or Working Storage field, up to a 5-bit field is transferred.

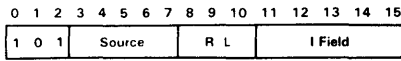
Example

Store the bit pattern 110 in the selected Working Storage field. The field name is VALUE and is located in bits 3, 4 and 5.

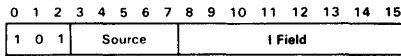


NZT S,I

Format



Type III



Type IV

Operation

Non-Zero Transfer. If (S) ≠ 0, PC offset by I-PC; otherwise PC + 1 → PC.

Description

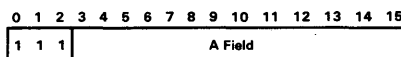
If the data specified by the S field is non-zero, replace the low order bits of the Program Counter with I. Otherwise, processing continues with the next instruction in sequence. If S is a register, the low order 8 bits of the PC are replaced; if S is an IV or Working Storage field, the low order 5 bits of the PC are replaced, resulting in an NZT range of 256 and 32 respectively.

Example

Jump to Program Address ALPHA if the selected IV byte field is non-zero. The field name is OVERFLO and it is a 1-bit field located in bit 3.

JMP A

Format: Type V



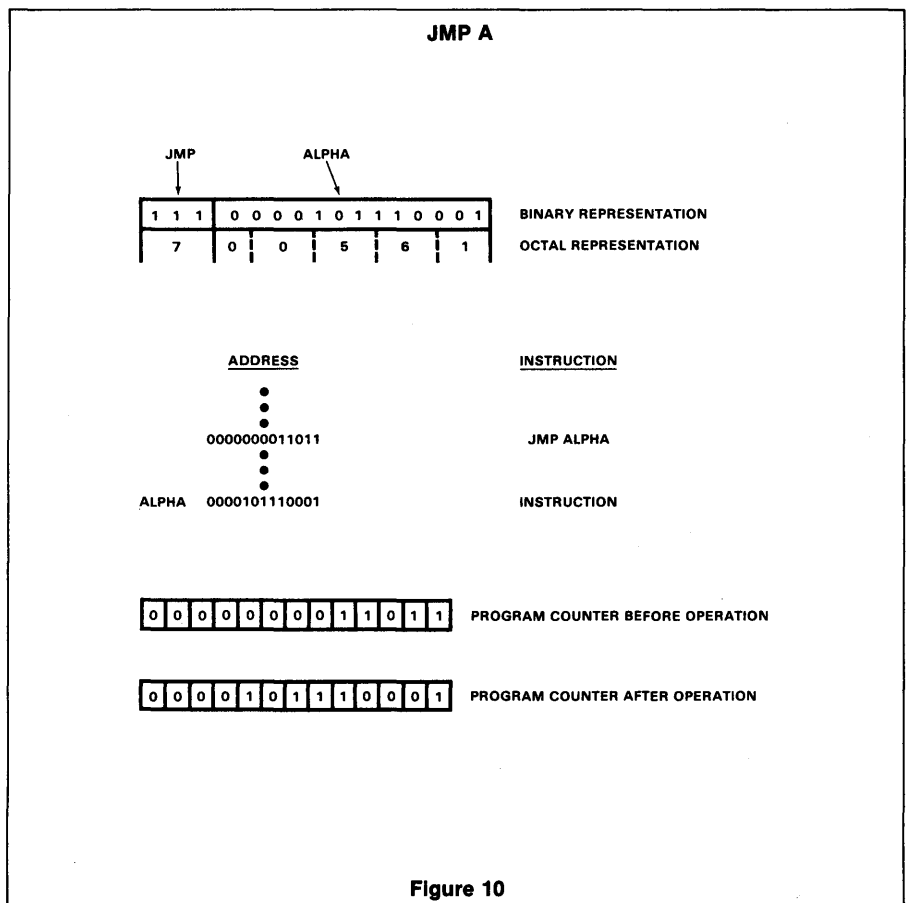
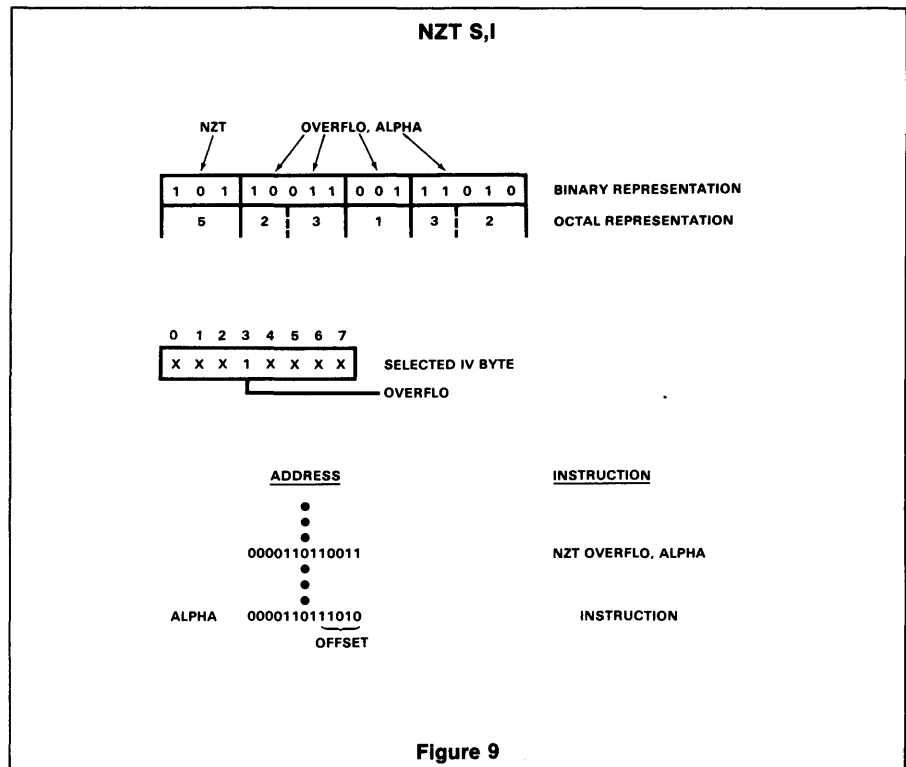
Operation: A → PC

Description

The literal value A is placed in the Program Counter and processing continues at location A. A has a range of 0-17777₈ in current systems (0-8191).

Example

Jump to location ALPHA (0000101110001)



DESCRIPTION

The 8X300 Cross Assembly Program, MCCAP, provides a programming language which allows the user to write programs for the 8X300 in symbolic terms. MCCAP translates the user's symbolic instructions into machine-oriented binary instructions. For example, the jump instruction, JMP, to a user defined position, say ALPHA, in program storage is coded as:

JMP ALPHA

and is translated by MCCAP into the following 16-bit word (see Figure 1).

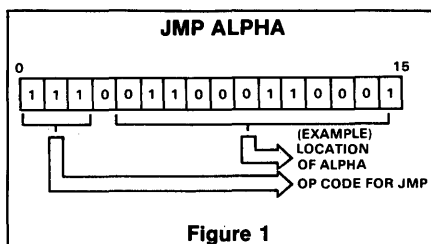


Figure 1

MCCAP allocates the 8X300 program storage and assigns Interface Vector and Working Storage address to symbols as declared in the user's program.

The ability to define data of the Interface Vector as symbolic variables is a powerful feature of MCCAP. Interface Vector variables may be operated on directly using the same instructions as those for variables in Working Storage and for the working registers.

The Assembler Declaration statements of MCCAP allow the programmer to define symbolic variable names for data elements tailored to his application. Individual bits and sequences of bits in Working Storage and on the Interface Vector may be named and operated upon directly by 8X300 instructions.

In addition to simplifying the language and bookkeeping of the program, MCCAP provides program segmentation and communication between segments; i.e., the main program and any subprograms. If a sequence of code appears more than once in a program, it can be written as a separate program segment, a subprogram, and called into execution whenever that subprogram's function is required. Program segmentation also permits the construction of a program in logically discrete units. These segments need not be written sequentially or even by the same person. The various program segments provide a function description, or block diagram, of the application. Communication between segments means that control and data can be transferred in both directions. MCCAP automati-

MCCAP SOURCE PROGRAM			
MICROCONTROLLER SYMBOLIC ASSEMBLER VER 1.0			
1680			
1681			
1682	01544	PROC	RDCMMD
1683			
1684	01544	6 07003	SEL IVRESP FDC RESPONSE BYTE
1685	01545	6 20101	XMIT UR, BCTRL ESTABLISH USER READ ONLY
1686	01546	6 07002	SEL IVDATA HOLDS COMMAND BYTE
1687	01547	0 27305	MOVE FUNC, R5 FUNCTION CODE
1688	01550	0 24306	MOVE DADDR, R6 DISK ADDRESS
1689	01551	0 21202	MOVE BUFF, R2 BUFFER FUNCTION CODE
1690	01552	6 07003	SEL IVRESP
1691	01553	6 25100	XMIT 0, DONE SHOW COMMAND IN PROGRESS
1692	01554	6 20100	XMIT UW, BCTRL RESTORE USER WRITE
1693	01555	6 27101	XMIT 1, XFR SIGNAL USER FDC ACCEPTED BYTE
1694	01556	6 07001	SEL IVCTRL USER CONTROL BYTE
1695	01557	5 26117	NZT CMMD,* WAIT FOR CMMD TO GO LOW
1696	01560	6 07003	SEL IVRESP FDC RESPONSE BYTE
1697	01561	6 27100	SEL IVCTRL LOWER XFR SIGNAL
1698	01562	6 07001	SEL IVCTRL USER CTRL BYTE
1699	01563	4 26123	XEC *(CMMD), 2 WAIT FOR NEXT COMMAND SIGNAL
1700	01564	6 07003	SEL IVRESP SECOND COMMAND BYTE AVAILABLE
1701	01565	6 20101	XMIT UR, BCTRL SET IVDATA TO USER READ ONLY
1702	01566	6 07002	SEL IVDATA 2ND COMMAND BYTE
1703	01567	0 27704	MOVE TRACK, R4 TRACK ADDRESS
1704	01570	0 27503	MOVE SECT, R3 SECTOR ADDRESS
1705	01571	6 07003	SEL IVRESP FDC RESPONSE BYTE
1706	01572	6 27101	XMIT 1, XFR SIGNAL USER
1707	01573	6 20100	XMIT UW, BCTRL RESTORE USER WRITE
1708	01574	6 07001	SEL IVCTRL
1709	01575	5 26136	NZT CMMD,* WAIT FOR CMMD TO GO LOW
1710	01576	6 07003	SEL IVRESP FDC RESPONSE BYTE
1711	01577	6 27100	XMIT 0, XFR LOWER XFR SIGNAL
1712			
1713	01600	7 01652	RTN RETURN
1714			
1715			END RDCMMD

Figure 2

cally generates the code for subprogram entry and exit mechanisms when the appropriate CALL and RTN statements are invoked.

MCCAP OUTPUT

The output from a MCCAP compilation includes an assembler listing and an object module. During pass two of the assembly process, a program listing is produced. The listing displays all information pertaining to the assembled program. This includes the assembled octal instructions, the user's original source code and error messages. The listing may be used as a documentation tool through the inclusion of comments and remarks which describe the function of a particular program segment. The main purpose of the listing, however, is to convey all pertinent information about the assembled program, i.e., the memory addresses and their contents.

The object module is also produced during pass two. This is a machine-readable computer output produced on paper tape. The output module contains the specifications necessary for loading the memory of the Microcontroller Simulator (MCSIM), for loading the memory of the SMS ROM Simulator, or for producing ROMs or PROMs. The object module can be produced in MCSIM, ROM Simulator or BNPF format.

An example of a MCCAP source program is shown in Figure 2.

PROGRAM STRUCTURE

Program Segments

A MCCAP program consists of one or more program segments. Program segments are the logically discrete units, such as the main program and subprograms, which comprise a user's complete program. Program segments consist of sequences of program statements. The first program segment must be the main program. The main program names the overall program and is where execution begins. All other segments are subprograms; each subprogram must be named. Control and data can be passed in both directions between segments. No segment may call itself, or one of its callers, or the main program. Program segments take the form as shown in Figure 3.

The Assembler Declaration statements define variables and constants. They must precede the use of the declared variables and constants in the Executable Statements in a program. The Executable Statements are those which result in the generation of one or more executable machine instructions.

Subprograms

Subprograms are program segments which perform a specific function. A major reason for using subprograms is that they reduce programming and debugging labor when a specific function is required to be executed at more than one point in a program. By

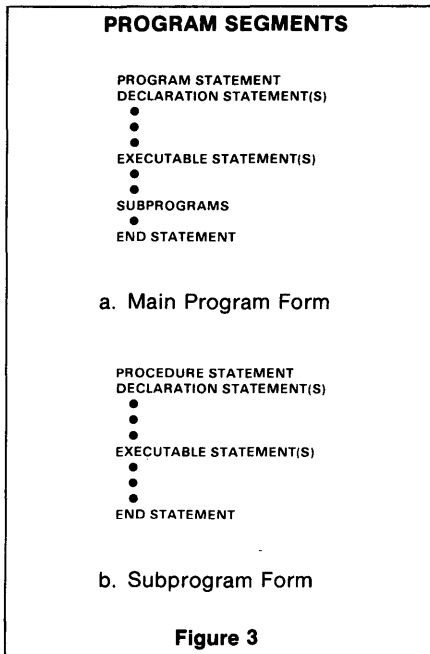


Figure 3

creating the required function as a subprogram, the statements associated with that function may be coded once and executed at many different points in a program. Figure 3 illustrates an example.

The program structure in Figure 3 causes the code associated with PROC WAIT to be executed three times within PROG MANY-WAIT. This is accomplished even though the statements associated with PROC WAIT are coded only once, rather than three times.

Subprogram Calls and Returns

For user-provided procedures, a jump to the associated procedure and a return link are created for each procedure reference. The instructions to accomplish this result in subprogram entry time. The instructions to accomplish subprogram exit result in exit time. The user may utilize the MCCAP procedure mechanism for linking calling programs with called programs or he may create his own instructions to do so. The following describes the linkage mechanism and timing for MCCAP user procedures.

Linkage between called and calling programs is achieved through the generation of an indexed "return jump" table, the length of which corresponds to the number of different times in the program that the subprograms are called. This table is generated automatically by MCCAP when procedure CALL and RTN statements are invoked. For each procedure reference, MCCAP creates two statements in the calling program. Thus, the time required for the subprogram entry is 0.5 microseconds. The subprogram

return mechanism requires the execution of three instructions or 0.75 microseconds. These times do not include saving and restoring of the working registers. The total time to save all working registers is 3.5 microseconds, the same time to restore all registers. Saving of all working registers is normally not necessary, but worst case calculations for entry and exit time below do include this time. Thus, subprogram exit and entry times are:

$$0.5\mu s \leq \text{Entry Time} \leq 4.0\mu s$$

$$0.75\mu s \leq \text{Exit Time} \leq 4.25\mu s$$

Details of the code required for procedure CALL and RTN are provided in the Programming Examples section.

Macros

A macro is a sequence of instructions that can be inserted in the assembly source text by encoding a single instruction. The macro is defined only once and may then be invoked any number of times in the program. This facility simplifies the coding of programs, reduces the chance of errors, and makes programs easier to change.

A macro definition consists of a heading, a body and a terminator. This definition must precede any call on the macro. In MCCAP, the heading consists of the MACRO statement which marks the beginning of the macro and names it. The body of the macro is made up of those MCCAP statements which will be inserted into the source code in place of the macro call. The terminator consists of an ENDM statement which marks the physical end of the macro definition.

MCCAP Statements

The MCCAP language consists of thirty statements categorized as follows:

- Assembler Directive Statements
- Assembler Declaration Statements
- Communication Statements
- Macro Statements
- Machine Statements

The following lists the statements in each category, describes their use, and provides examples. Detailed use of the instructions including rules of syntax and parameter restrictions are described in the MCCAP Reference Manual.

Assembler Directive Statements

Assembler Directive statements define program structure and control the assembler outputs. They do not result in the generation of 8X300 executable code. There are twelve Assembler Directive statements:

- PROG Statement
- PROC Statement

- ENTRY Statement
- END Statement
- ORG Statement
- OBJ Statement
- IF Statement
- ENDIF Statement
- LIST Statement
- NLIST Statement
- EJCT Statement
- SPAC Statement

PROG Statement

Use
Defines the names and marks the beginning of a main program.

Example: PROG PROCESS

PROC Statement

Use
Defines the names and marks the beginning of a subprogram.

Example: PROC WAIT

ENTRY Statement

Use
Defines the name and marks the location of a secondary entry point to a subprogram.

Example: ENTRY POINT 2

END Statement

Use
Terminates a program segment or a complete program.

Examples: END SUB1
END MAIN

ORG Statement

Use
Sets the program counter to the value specified in the operand field.

Example: ORG 200

OBJ Statement

Use
To specify the format of the object module.

Examples: OBJ R
OBJ M
OBJ N

NOTE

"R" indicates the ROM Simulator format. "M" indicates the Microcontroller Simulator format. "N" indicates BNPF format.

IF Statement

Use
To mark the beginning of a sequence of code, which may or may not be assembled depending on the value of an expression.

Examples: IF VAL
IF X + Y

ENDIF Statement*Use*

To mark the end of sequence of code, which is to be conditionally assembled. In the case of nested IF statements, an ENDF is paired with the most recent IF.

Example: ENDF

LIST Statement*Use*

To select and control output of a MCCAP assembly.

Example: LIST S,O,M,I

NLIST Statement*Use*

To suppress elements of the output from a MCCAP assembly.

Example: NLIST O,M,I

EJCT Statement*Use*

To cause the output listing to be advanced to the next page.

Example: EJCT

SPAC Statement*Use*

To insert blank lines into the output listing. The number of lines inserted is indicated in the operand field.

Example: SPAC 3

Assembler Declaration Statement

Assembler Declaration statements define and describe the data, constants and variables, in a program or subprogram. There are four Assembler Declaration statements:

EQU Statement
SET Statement
LIV Statement
RIV Statement

EQU Statement*Use*

To define a fixed constant.

Examples: FIVE EQU 5
ON EQU 1

SET Statement*Use*

To define and assign a value to a constant, which may later be assigned a new value by another SET statement.

Example: OFF SET 0

LIV Statement*Use*

To define and assign symbolic names to variables, usually IV bytes, located on the left bank of the Interface Vector.

Example: LITE LIV 23,2,1

NOTE

The effect of the above example is to define a variable whose name is LITE. It is located in a byte whose address is 23. The right-most bit of LITE is bit 2 and its length is 1 bit.

RIV Statement*Use*

To define and assign symbolic names to variables, usually in Working Storage, located on the right bank of the Interface Vector.

Example: DATA RIV 200,6,3

NOTE

The effect of the above example is to define a variable whose name is DATA. It is located in a byte whose address is 200. The right-most bit DATA is bit 6 of the byte and its length is 3 bits.

Communication Statements

Communication statements are executable statements which provide the mechanism for main program to subprogram linkage. They provide the means by which subprograms are called and returned from. There are two kinds of Communication statements:

CALL Statement
RTN Statement

CALL Statement*Use*

To transfer control from a calling program to the called subprogram. The CALL statement causes the generation of two 8X300 instructions.

Examples: CALL WAIT
CALL SINE

NOTE

The above are valid statements to be coded into the program if WAIT and SINE have been defined in PROC statements. The effect of invoking these statements is to transfer execution control to the procedures WAIT and SINE respectively.

RTN Statement*Use*

To transfer control from a called subprogram to a calling program.

Example: RTN

Macro Statements

Macro statements provide the mechanism for defining macros and for inserting them into the source code. There are three Macro statements:

MACRO Statement
ENDM Statement
MACRO CALL Statement

MACRO Statement*Use*

To mark the beginning of a macro definition. The MACRO statement forms the heading of the macro definition.

Examples: MAC1 MACRO
MAC2 MACRO A,B,C

NOTE

The second example would mark the beginning of a macro called MAC2. The "A,B,C" represents a formal parameter list. These parameters, used in writing the macro body, will be replaced by the actual parameters listed in the MACRO CALL statement.

ENDM Statement*Use*

To mark the end of a macro definition. The ENDM statement forms the terminator of the macro definition.

Example: ENDM

MACRO CALL Statement*Use*

To indicate where a macro is to be inserted into the source code and to specify any actual parameters needed by the macro.

Example: MAC2 DATA, INPUT, RESULT

NOTE

There is no single macro call statement. Any macro name which has been defined as such may be coded as if it were a valid MCCAP statement. The macro name is coded in the operation field and the actual parameters are placed in the operand field.

Machine Statement

Machine statements are the MCCAP symbolic representations of the 8X300 executable statements. Machine statements have a one to one correspondence to 8X300 instructions. Each Machine statement results in the generation of a single 8X300 instruction. There are eight Machine statements:

MOVE Statement
ADD Statement
AND Statement
XOR Statement
XMIT Statement
XEC Statement
NXT Statement
JMP Statement

MOVE Statement*Use*

To copy the contents of a specified register, WS variable or IV variable into a specified register, WS or IV. Defined in Instruction Descriptions.

Examples: MOVE R1(6),R6
MOVE X,Y

NOTE

The first example illustrates a six place right rotate of R1's data before it is moved to R6. The contents of R1 are not affected. The second example may be a Working Storage or Interface Vector variable move, depending on the way X and Y are defined in Declaration Statements.

ADD Statement*Use*

To add the contents of a specified register, WS variable, or IV variable to the contents of the AUX register and place the result in a specified register, WS variable or IV variable.

Examples: ADD R1(3),R2
ADD DATA,OUTPUT

NOTE

The first example illustrates a three place right rotate of R1's data before the addition is carried out. Under certain conditions a rotate may be used to multiply the specified operand by a power of 2 before the addition is done. The contents of R1 are not affected. The second example suggests that the contents of WS variable have been added to the contents of the AUX register and the result placed in an IV variable, making the result immediately available to the user's system.

AND Statement*Use*

To compute the logical AND of the contents of a specified register, WS variable or IV variable and the contents of the AUX register. The logical result is placed in a specified register, WS variable or IV variable. In actual practice, the AND statement is often used to mask out undesired bits of a register.

Examples: AND R2,R2
AND R3(1),R5
AND X,Y

NOTE

The first example illustrates the use of an AND statement in what might be a masking operation. If the AUX register contains 00001111 then this statement sets the 4 high order bits of R2 to 0 no matter what they were originally. The 4 low order bits of R2 would be unaffected.

The second example illustrates a one place rotate to the right of R3's data before the AND is carried out. The contents of R3 are not affected. In the third example, X and Y may be parts of the same WS or IV byte, or one may be a WS byte and the other an IV byte.

XOR Statement*Use*

To compute the logical exclusive OR of the contents of a specified register, WS variable

or IV variable and the contents of the AUX register, and place the result in a specified register, WS variable or IV variable. In practice, the XOR statement is often used to complement a value and to perform comparisons.

Examples: XOR R6,R11
XOR R1(7),R4
XOR X,Y

NOTE

The first example illustrates the use of an XOR statement in what might be a complementing operation. If the AUX register contains all 1's then the execution of this statement results in the complement of the contents of R6 replacing the contents of R11. The second and third examples are of the same form as the second and third examples of the AND statement.

XMIT Statement*Use*

To transmit or load literal values into registers, WS variables or IV variables.

Examples: XMIT DATA,IVR
XMIT OUTPUT,IVL
XMIT -11,AUX
XMIT -00001011B,AUX
XMIT -13H,AUX

NOTE

The first example selects a previously declared WS variable by transmitting its address to the IVR register. The second example selects a previously declared IV variable by transmitting its address to the IVL register. The last three examples all result in the generation of the same machine code. They all load the AUX register with -11₁₀. In the first case, the programmer has written the number in base 10. In the second case, the programmer has written the number in binary and has indicated this by placing a B after the number. In the third case, the number has been written in octal as indicated by an H after the number.

XEC Statement*Use*

To select and execute one instruction out of a list of instructions in program memory as determined by the value of a data variable, and then continue the sequential execution of the program beginning with the statement immediately following the XEC unless the selected instruction is a JMP or NZT statement.

Examples:

JTABLE JMP XEC JTABLE(R1),3
GR8ERTHAN
JMP LESSTHAN
JMP EQUALTO

XEC SEND(INPUT),4
"NEXT INSTRUCTION"
"NEXT INSTRUCTION"

SEND XMIT 11011011B,AUX
XMIT 11111111B,AUX
XMIT 10101010B,AUX
XMIT 00000000B,AUX

NOTE

In the first example, the execution of the program will be transferred to one of three labeled instructions on the basis of whether register R1 contains 0, 1 or 2. In the second example, the XEC statement causes the execution of a statement which transmits a special bit pattern to the AUX register in response to an input signal which is either 0, 1, 2 or 3. After the pattern is transmitted, the execution of the program continues with the next instruction after the XEC.

NZT Statement*Use*

To carry out a conditional branch on the basis of whether or not a register, WS variable, or IV variable is zero or non-zero.

Examples: NZT R1,*+2
NZT SIGN,NEG

NOTE

In the first example, if the contents of R1 are non-zero, then program execution will continue with the instruction, whose address is the sum of the address of the NZT statement and 2. If the contents of R1 are 0, the program execution continues with the next instruction after the NZT statement. In the second example, if the contents of a WS or IV variable called SIGN is non-zero, then program execution will continue beginning with the instruction whose address is NEG. Otherwise execution continues with the next instruction after the NZT statement.

JMP Statement*Use*

To transfer execution of the program to the statement whose address is the operand of the JMP statement.

Examples: JMP START
JMP *-2

NOTE

In the first example, execution of the program continues sequentially beginning with the instruction labeled START. In the second example, program execution continues beginning with the instruction whose address is the JMP instruction's address minus 2.

SEL Statement*Use*

Select a variable in Working Storage or on the Interface Vector, so that subsequent machine instructions may reference that variable.

Examples: SEL DATA
SEL OUTPUT

NOTE

It is the programmer's responsibility to assure that the proper page has been addressed before calling the SEL statement if the variable may be in Working Storage. The SEL statement causes a single instruction, XMIT, to be assembled into the user program. The operand of the XMIT instruction is the byte address of the named variable (argument of the reference) as it has been allocated in Working Storage or on the Interface Vector.

PROGRAMMING EXAMPLES

This section contains programming examples which demonstrate how the 8X300's instructions can be assembled to perform some simple, commonly required functions. These examples are written as program

fragments. They are not complete programs as the Data Declaration and Directive statements have been omitted. Otherwise, they follow standard MCCAP conventions.

Looping

Looping is terminated by incrementing a counter and testing for zero. Register R1 is used as counter register and is loaded with a negative number so that the program counts up to zero. Figure 4 illustrates the process.

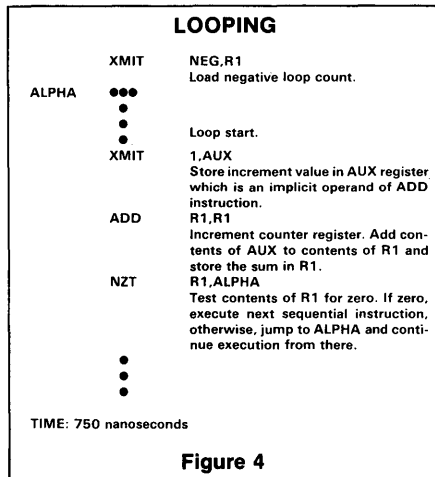


Figure 4

Inclusive-OR (8 Bits)

Generate inclusive-OR of the contents of R1 and R2. Store the logical result in R3. Although the 8X300 does not have an OR instruction, it can be quickly implemented by making use of the fact that $(A + B) + (A \oplus B)$ is logically equivalent to $A \oplus B$.

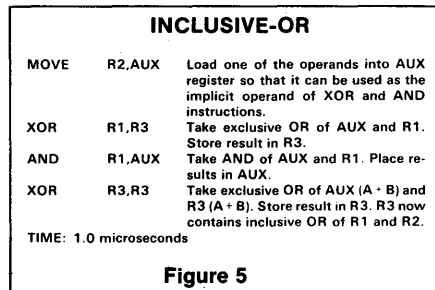


Figure 5

Two's Complement (8-Bits)

Generate the two's complement of the contents of R2. Store the result in R3. Assume that R2 does not contain 200.

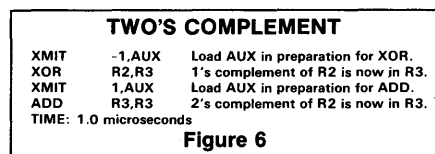


Figure 6

8-Bit Subtract

Subtract the contents of R2 from the contents of R1 by taking the two's complement

of R2 and adding R1. Store the difference in R3.

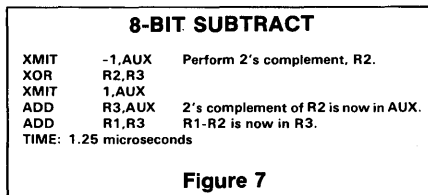


Figure 7

16-Bit ADD, Register to Register

Add a 16-bit value stored in R1 and R2 to a 16-bit value in R3 and R4. Store the result in R1 and R2.

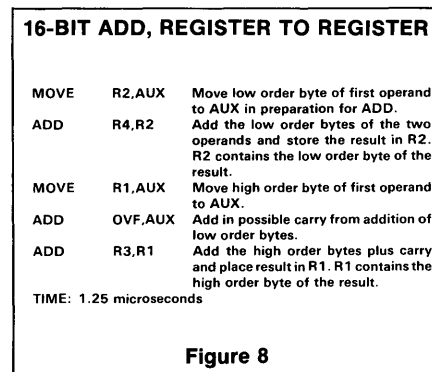


Figure 8

16-Bit ADD, Memory to Memory

Add a 16-bit value in Working Storage, OPERAND1, to a 16-bit value in Working Storage, OPERAND2, and store result in Working Storage OPERAND1. H1 and L1 represent the high and low order of bytes OPERAND1. H2 and L2 represent the high and low order bytes of OPERAND2.

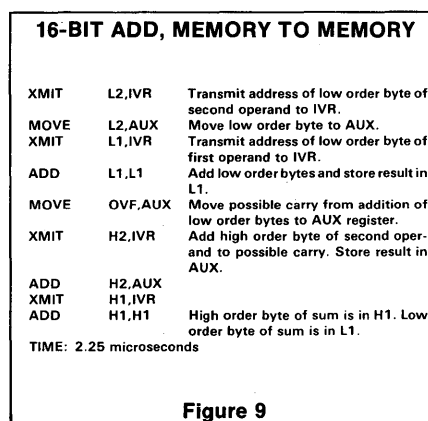


Figure 9

Byte Assembly From Bit Serial Input

This is typical of problems associated with interfacing to serial communications lines. An 8-bit byte is assembled from bit inputs that arrive sequentially at the Interface Vector. A single bit on the Interface Vector

named STROBE is used to define bit timing, and a second bit, named INPBIT, is used as the bit data interface. Figure 10 illustrates the byte assembly.

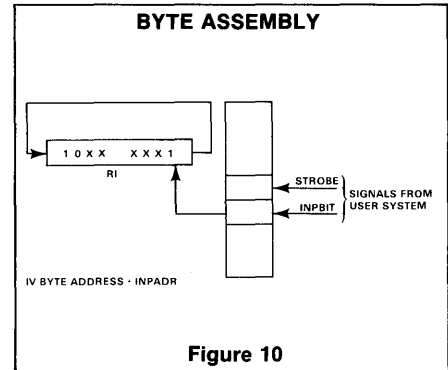


Figure 10

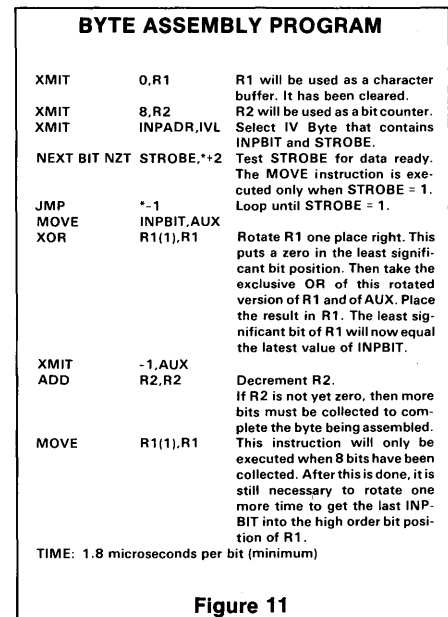


Figure 11

Rotate Left

The 8X300 has no instructions which explicitly rotate data to the left. Such an instruction would be redundant because of the circular nature of the rotate operation. For example, a rotate of two places to the left is identical to a rotate of six places to the right. The rotate n places to the left in an 8-bit register, rotate 8-n places to the right. This example illustrates a rotate of the contents of R4 three places to the left.

```
MOVE R4(5),R4
TIME: 250 nanoseconds
```

Three Way Compare

The contents of R1 are compared to the contents of R2. A branch is taken to one of three points in the program depending upon whether $R1 = R2$, $R1 < R2$, or $R1 > R2$.

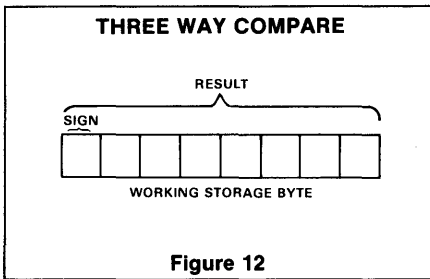


Figure 12

THREE WAY COMPARE PROGRAM

```

XMIT RESULT,IVR    Choose a working Storage byte
                    by transmitting its address to
                    IVR register.
XMIT  -1,AUX        Load AUX with all 1's, in prepa-
                    ration for complementing con-
                    tents of R2.
XOR   R2,RESULT    Store complement of R2 in RE-
                    SULT.
XMIT  1,AUX
ADD   RESULT,AUX  AUX now contains 2's comple-
                    ment of R2.
ADD   R1,RESULT   RESULT now contains R1-R2.
NZT  RESULT,NEQUAL If RESULT ≠ 0, then R1 ≠ R2.
JMP   EQUAL
NEQUAL NZT SIGN,LESS Sign Bit = 1 only when R1 < R2.
GREATER Continue

EQUAL Continue

LESS Continue
TIME: 2.0 microseconds
    
```

Figure 13

Interrupt Polling

Three external interrupt signals are connected to three IV bits. The three bits are scanned by the program to determine the presence of an interrupt request. A branch is taken to one of eight program locations depending upon whether any or all of the interrupt request signals are present. The IV bits associated with the interrupt requests are wired to the low order three bits of the IV byte named Control. Figures 14 and 15 illustrate the interrupt polling.

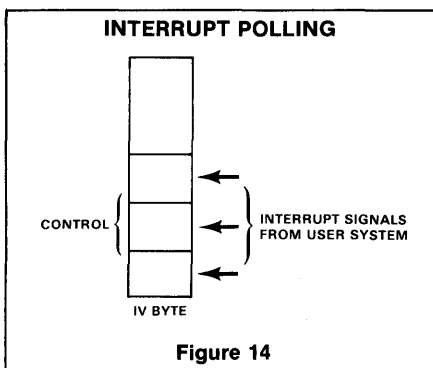


Figure 14

INTERRUPT POLLING PROGRAM

```

XMIT CONTROL,IVL    Choose proper IV Byte
                    by transmitting its address
                    to IVL register.
XEC   JTABLE (CONTROL),8 Execute the one in-
                    struction whose address is
                    the sum of JTABLE and the
                    contents of CONTROL. The
                    8 indicates the length of
                    the table.

JTABLE JMP ALPHA1
      •
      •
      •
      •
      •
      •
      •
      •
      JMP ALPHA8
TIME: 750 nanoseconds.
    
```

Figure 15

Bit Pattern Detection In An I/O Field

Test input field called Input for specific bit pattern, for example: 1 0 1 1. If pattern is not found, branch to NFOUND, otherwise continue sequential execution. Figures 16 and 17 illustrate the procedure.

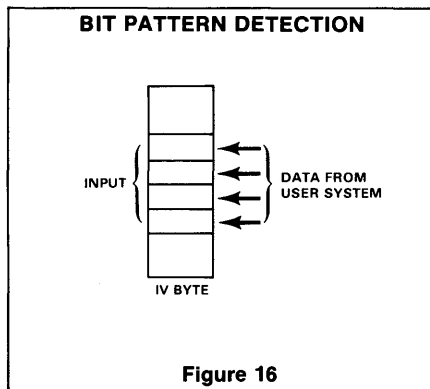


Figure 16

BIT PATTERN DETECTION PROGRAM

```

XMITI INPUT,IVL    Choose proper IV Byte by trans-
                    mitting its address to IVL
                    register.
XMIT  1011B,AUX    Store desired bit pattern in
                    AUX register for use as im-
                    plicit operand of XOR in-
                    struction.
XOR   INPUT,AUX    Take exclusive OR of the
                    contents of INPUT and
                    AUX. Store the result in
                    AUX. Now the contents of
                    AUX will be zero if the
                    contents of INPUT are
                    1011.
NZT  AUX,NFOUND   Test AUX for zero. Branch
                    to NFOUND if non-zero.

•
•
•
•
NFOUND Continue
TIME: 1.0 microseconds
    
```

Figure 17

Control Sequence #1

Set an output bit when an input bit goes high (is set) (see Figure 18).

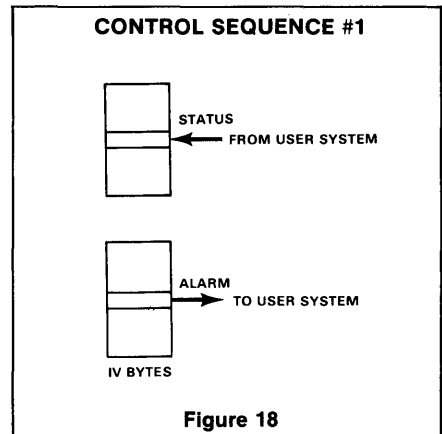


Figure 18

CONTROL SEQUENCE #1 PROGRAM

```

XMIT STATUS,IVL    Choose input IV byte by trans-
                    mitting its address to IVL.
NZT  STATUS,*2     Test input bit to determine
                    whether it is still zero.
                    Skip next instruction if
                    it is not zero.
JMP  *-1
XMIT ALARM,IVL    Jump to previous instruction.
                    Choose output IV byte.
XMIT 1,ALARM      Set output bit by loading
                    ALARM with 1.
TIME: 1.0 microseconds (minimum)
    
```

Figure 19

Control Sequence #2

Output a specific 5-bit pattern in response to a specified 3-bit input field.

Subprogram Calls and Returns

The mechanism for managing subprogram calls and returns is based on assigning a return link value to each subprogram caller; this return link value is then used, on exit from the subprogram, to index into the return jump table which returns control to the callers of the subprogram. Figure 21 is an example of a subprogram called from four different locations in the main program.

As seen from Figure 21, each subprogram (or procedure) caller is assigned a "tag" or index values ranging from 0 to 3, or a total of four index values for the four callers. Before jumping to the subprogram, the index value is placed in a previously agreed upon location, register R11 in this case. Upon exit from the subroutine, the index value stored in R11 is used as an offset to the Program Counter in order to execute the proper JMP instruction. The key to returning to the proper caller is the index jump table. Figure 22 gives a detailed description of the return operation.

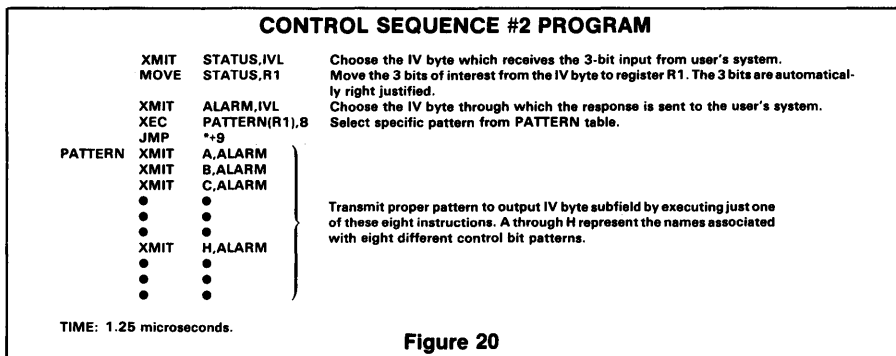


Figure 20

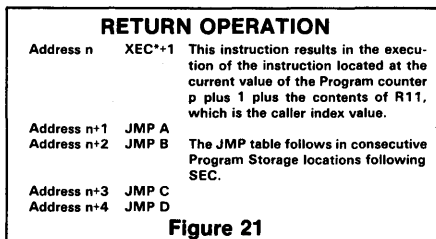


Figure 21

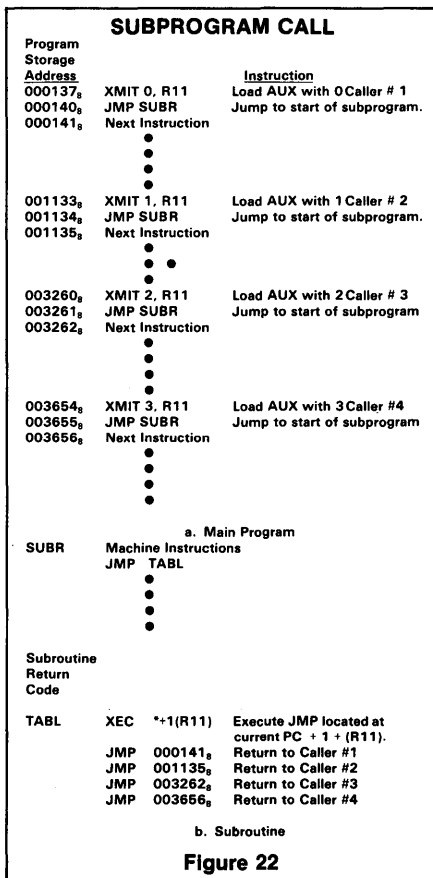


Figure 22

ABSTRACT

Many possible applications for microprocessors demand a very quick response to requests for action or information. While MOS microprocessors are relatively cheap, they do not generally possess the necessary speed. Although bipolar microprocessors tend to possess greater speed, they are mostly designed as general purpose devices, which means that they are not ideally suited to the requirements of a fast real-time microcomputer system. The Signetics 8X300 microprocessor has been specifically designed to fulfill this role. This article describes the architecture and instruction set of the 8X300 and, by the use of examples, explains the capabilities and applications of the device.

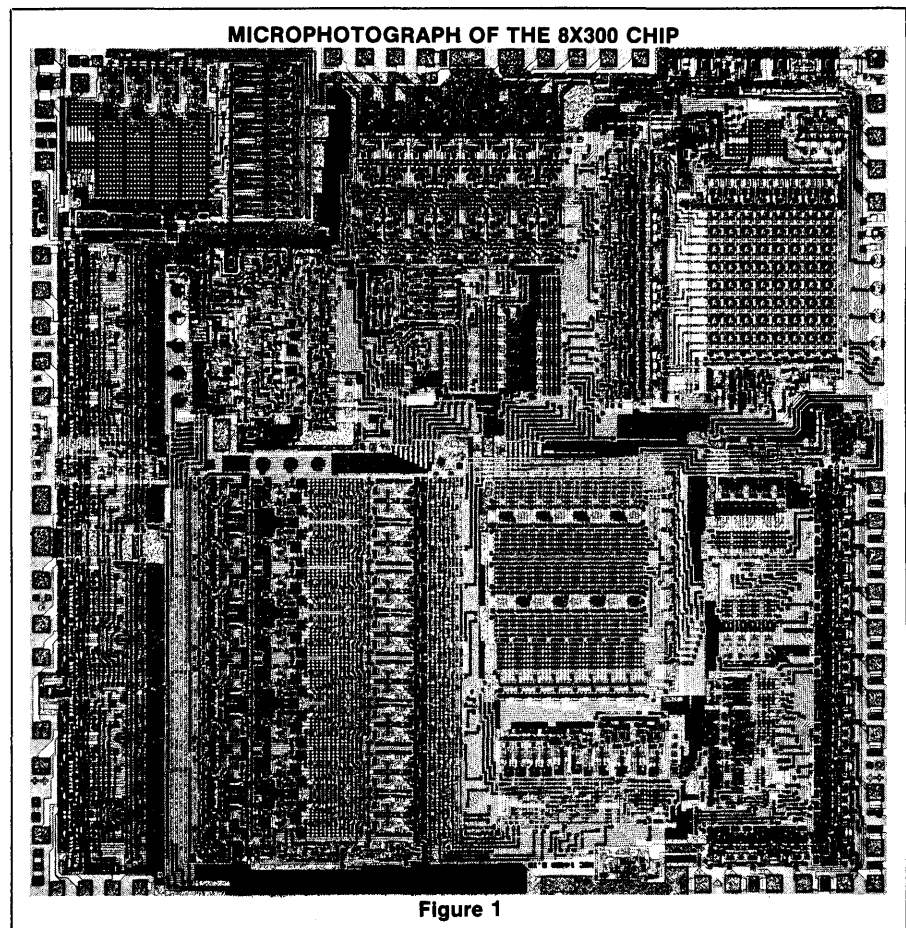
Considerably improved data throughput is obtained from the use of separate data and address buses for the program memory, coupled with extremely flexible I/O control. Data may be input, modified and output all in the same instruction by the use of the two independent, parallel I/O ports.

INTRODUCTION

As semiconductor technology improved, allowing greater die area with economically acceptable yield, the amount of logic that could be put on a marketable integrated circuit increased. It naturally followed that rather than provide more individual elementary gates on a single die, these gates would be interconnected to afford the user of these chips more complicated logic functions in a single package. The attractiveness of the more complex integrated circuits compelled semiconductor manufacturers to strive for increasing circuit density.

The prospect of putting an entire, although elementary, computer CPU on a single die focused attention on those fabrication processes which allowed the greatest densities. Therefore, the MOS process was the first to yield an entire microprocessor on a chip. Unfortunately, a price was paid in that the MOS processes did not produce as high a speed of logic element as the usual bipolar processes. Because of density limitations, the bipolar process could only produce the less dense parts of chip microprocessors—the bit slices.

Now, however, the improvements in bipolar technology permit the construction of single chip microprocessors with all of the performance advantages of bipolar Schottky technology. Such a circuit has been fabricated and is being produced with significantly high yield to allow commercial availability of quantity parts. The product is the 8X300 microprocessor produced by Signetics. It is the purpose of this paper to



present the 8X300 by discussing the architecture and some of the key fabrication and technology features of the microprocessor. This paper concludes with a brief review of some of the present as well as potential applications of this device.

The 8X300 was optimized for control applications rather than for extensive numerical processing, so before the main presentation begins, it is advisable to describe the basic requirements in the envisaged application field of the 8X300.

Control here applies to a wide variety of areas and is not necessarily limited to those specific areas itemized below. The action of control may be the sole purpose of a stand-alone microprocessor. In such a task, the microprocessor examines statuses at a particular rate and issues command words or bits to the external circuitry to effect the function of the whole machine as it is described in the control program. Thus, the microprocessor selects specific bits defined by the program, tests the bits, and responds or directs by setting or clearing other bits. Although elementary on the surface, this task may be quite complex involving timing, interval measurement, and various forms of

decision making, all at potentially high speed. Control may also take the form of bit or word manipulation and data movement such as in data concentrators, communication controllers, disk and tape controllers and similar devices. Here the data destined for storage, transfer or transmission may require alteration (for example bit packing, preamble addition or error detection/correction); consequently the control also involves calculation or data generation. Consider an industrial metal cutter required to form a complex shape as directed by some external data input. Matrix multiplications may be a very necessary part of this controller's process in order to carry out its function.

Thus, we see that controllers in this context may perform a wide variety of bit and arithmetic processing depending upon the type of controller one is discussing. The 8X300 is capable of good performance in all of these control areas.

ARCHITECTURE

The architecture of a microprocessor is intimately connected to the technology used to produce the device, for one could

define architectures which are realizable only with certain fabrication approaches. Also, a microprocessor's architecture is described by its instruction set and its input/output structure. So, in this section, the 8X300 will be examined both from the inside—technology, block diagram, etc., and from the outside—instruction set, I/O bus, timing, etc.

The 8X300 is fabricated using standard Schottky technology. Dual layer metallization is used to minimize die area, reduce capacitance and hence maximize the speed of the processor. A microphotograph of the 8X300 die is shown in Figure 1. The die measures 250 mils square and is the largest bipolar microprocessor in existence. The 8X300 is a complete processor on a single chip and, as will be seen later, results in a minimum circuit count processor system. Linear elements are also provided on the die as shown in Figure 2.

One functional entity is the clock generator circuit, which oscillates at a frequency determined by an external crystal or timing capacitor. This circuit generates all timing signals required internally by the 8X300 and externally for bus timing. Secondly, a voltage regulator in combination with an externally connected (user-provided) pass transistor, provides a stable low voltage source for the operation of selected internal segments. This voltage is approximately 3 volts and is used in areas where power conservation rather than speed is a prime concern. (The 3 volts does not imply I²-L utilization.) Maximum current used is 450mA (300mA typical) with 150mA used in the 5 volt (V_{CC}) connection and 300mA used in the 3 volt (V_{CR}).

With the regulator, the entire processor operates from a single +5 volt supply over the commercial temperature range (0°C to

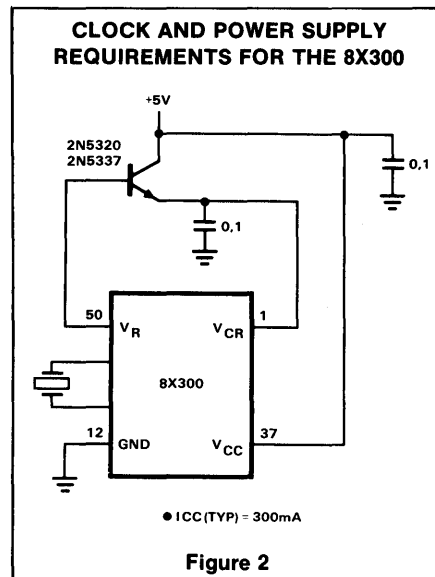


Figure 2

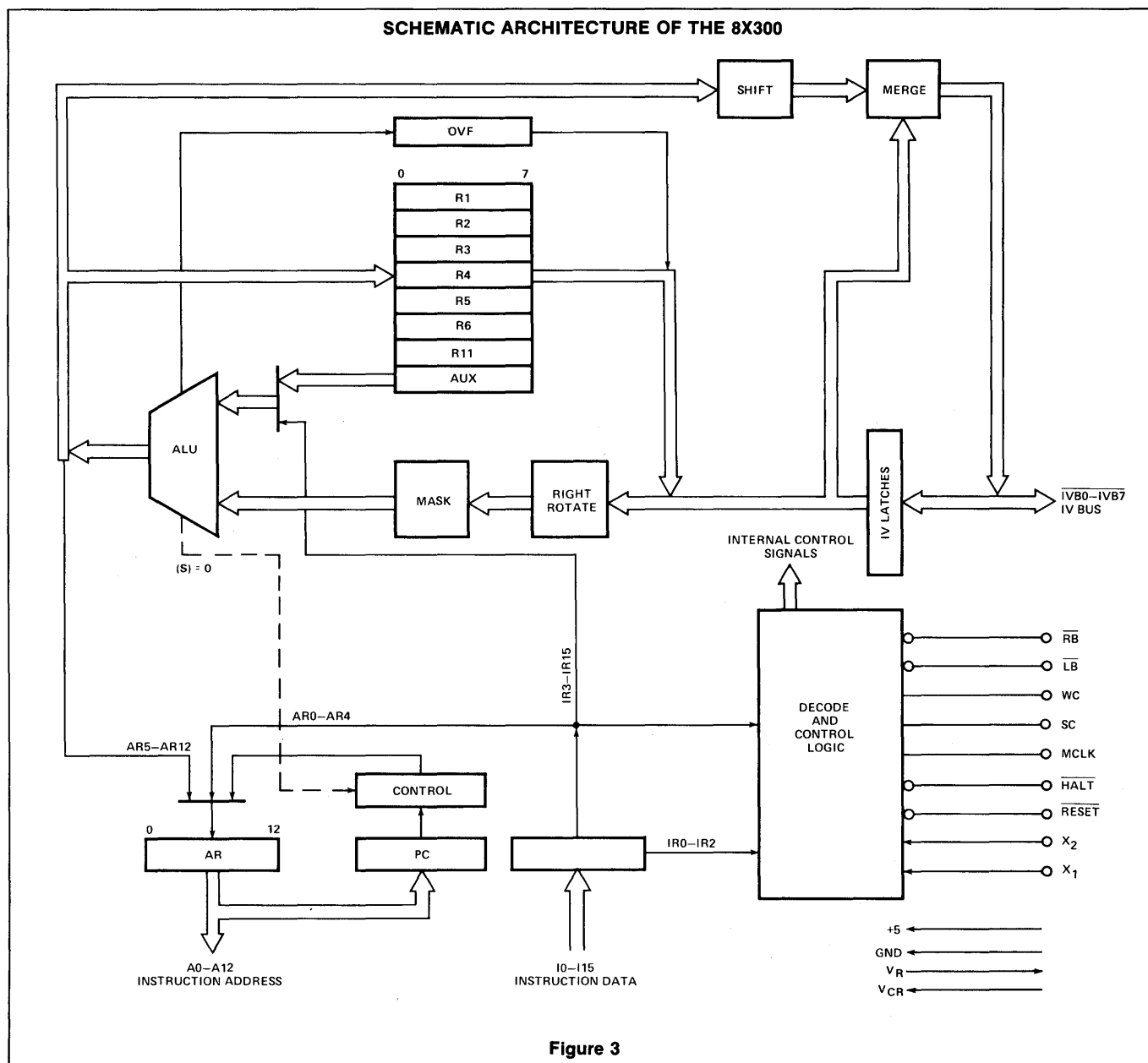


Figure 3

+70°C). The 8X300 is packaged in a 50-pin dual-in-line ceramic package.

The block diagram of the 8X300 processor is shown in Figure 3. It does not show the circuitry just described. First, note that full instruction decoding logic is provided to interpret the instruction classes and perform the indicated operation. This will be discussed in more detail later. This decoding and control logic provides all internal signals required as well as certain control lines for data input and output. These lines are RB, LB, WC, SC and MCLK. External control may be applied to hold the 8X300 in a non-processing or wait state (via halt) or force the processor to instruction address zero (reset). The processor also contains its own program counter (PC) which is automatically incremented upon instruction execution or, in certain cases, is not incremented or is loaded with a new value. Current address control, provided by the address register (AR) may be derived all or in part from the program counter, the instruction data (AR0-AR4) or from the output of the ALU (AR5-AR12). Thus, the present and future instruction to be executed may be altered through instructions or the condition of selected data.

Input/Output

Separate buses are provided for instruction address and instruction data. The current contents of the address register (AR) are presented on a 13-bit bus (A0-A12) to the program memory to fetch the 16-bit instruction word. The 8X300 possesses the capability of directly addressing 8K of program storage. The instruction word enters the processor via the instruction bus (I0-I15) and is stored in the instruction register (R).

The processing part of the 8X300 is shown in the upper half of Figure 3. The entire processor is oriented about 8-bit data manipulation; therefore interfaces to external circuitry use an 8-bit bus, designated the Interface Vector (IV) bus (IV0-IV7). For internal storage of data, eight 8-bit read/write registers are provided, designated R1-R6, R11 and AUX (auxiliary). The auxiliary register contains one of the operands that are used in two operand instructions such as ADD, AND and XOR (Exclusive-OR). A 1-bit overflow register (OVF) is provided to store the overflow resulting from add operations. The IV latch is not addressable, but stores original data brought in from the IV bus to be used in the merge operation prior to output. At the heart of the processing is the ALU which performs various arithmetic and logic operations on data. The ALU, when combined with the rotate, mask, shift and merge elements, permits unique data operations.

Before proceeding, it is essential that the IV bus concept be explained. From this, we shall go back and discuss the architecture and instruction set in greater detail. The IV bus serves both as an address and data bus

and is accompanied by the bus control signals shown in Figure 4. Since the bus carries addresses as well as data, I/O ports must be enabled before data transfers may take place. This is usually accomplished by presenting an address on the bus under program control. The control line SC is used to indicate address content of the bus. When presented with an address, an I/O port either enables itself (becomes active on the bus to accept or present data) if the address presented is its own, or disables itself (becomes inactive) if the address presented does not match its own address.

Together with this, processor I/O ports have been designed which allow 1 of 512 interface vector bytes to be selected without decoders. Having two ports, one for the user and the other to the microprocessor, these IV bytes are completely bidirectional. The unique feature of these bytes is the way in which they are addressed.

Each IV byte has an 8-bit field programmable address, which is used to enable the microprocessor port, allowing data transfer through it.

To effect input and output data transfer, the 8X300 IV outputs are three-state drivers. Additionally, to control external devices, the 8X300 issues the write command, WC, which indicates whether data transfers are read (into the 8X300) or write (out of the 8X300). The bus direction is entirely under control of the 8X300.

A unique feature of the 8X300 is the partitioning of the bus into two banks, designated left bank (LB) and right bank (RB). Using the LB and RB signals from the processor as master enables for the I/O ports, the processor may dynamically select ports as Figure 5 illustrates. Two I/O ports may be active during one cycle provided that they are on opposite banks. To do this,

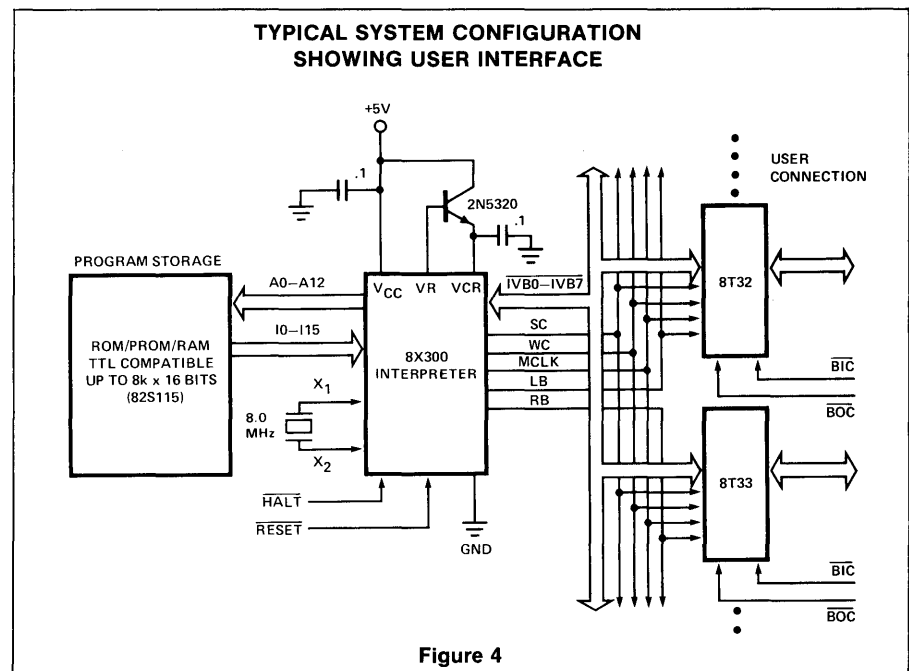


Figure 4

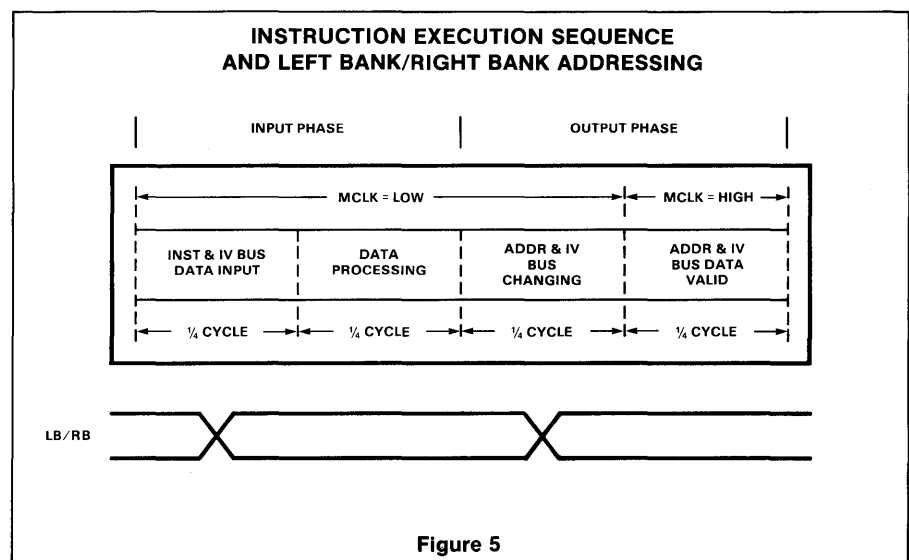


Figure 5

I/O ports recognize addresses, data or controls only when enabled by the bank signal to which they are connected. Clearly, the bank partitioning may be considered as a ninth address bit which is alterable by the processor within an instruction. (The 8X300, therefore, has 512 direct I/O port address capability.) A general data operation between two I/O ports could follow the following steps. First, an address is presented to one bank enabling a selected I/O port and disabling all others on that bank. Secondly, another address is presented to the opposite bank effecting a similar selection there. Subsequently, in one instruction cycle, the 8X300 may accept data from one port (on one bank), operate on the data and deposit the result in the other port in the second bank. If the working storage of the registers is not sufficient, additional storage can be added using an I/O port address to add another 256X8 words of RAM. See Figure 6.

In order to fully appreciate the speed of the last operation, accepting data from one port and depositing it on the other, it is necessary to explore the details of the instruction cycle. Each 8X300 operation is executed in one instruction cycle which is subdivided into four quarter cycles. The quarter cycles are shown in Figure 5. The instruction address for an operation is presented at the output during the third quarter of the previous instruction cycle. With a memory of sufficient speed, the instruction is returned and accepted by the processor during the first quarter of the cycle in which that instruction is to be executed. The instruction is decoded and used to direct the operation of the processor throughout the cycle.

For data processing, the instruction cycle may be viewed as having two halves. During the first half of the cycle, data to be processed is brought into the processor and stored in the IV latch. This is accomplished during the first quarter cycle. The next quarter cycle of this first half is used to bring the data through the ALU, thereby processing the data as required by the instruction. The second half cycle is the output phase during which the data is presented to the IV bus and finally clocked into the appropriate I/O port after bus stabilization. The processor issues MCLK for this purpose.

Bank selection during input and output phases is independent, thus data may be input from the right bank and deposited in the left bank or vice-versa, or to and from the same bank if the same IV is used. Bank selection during instruction cycle phases is specified by the instruction. Therefore, the processor may input data from one port, operate on the data and return it to a second port in one instruction cycle time. Remember that instruction fetching is concurrent with data operations. The cycle time is 250ns, making the 8X300 comparable in

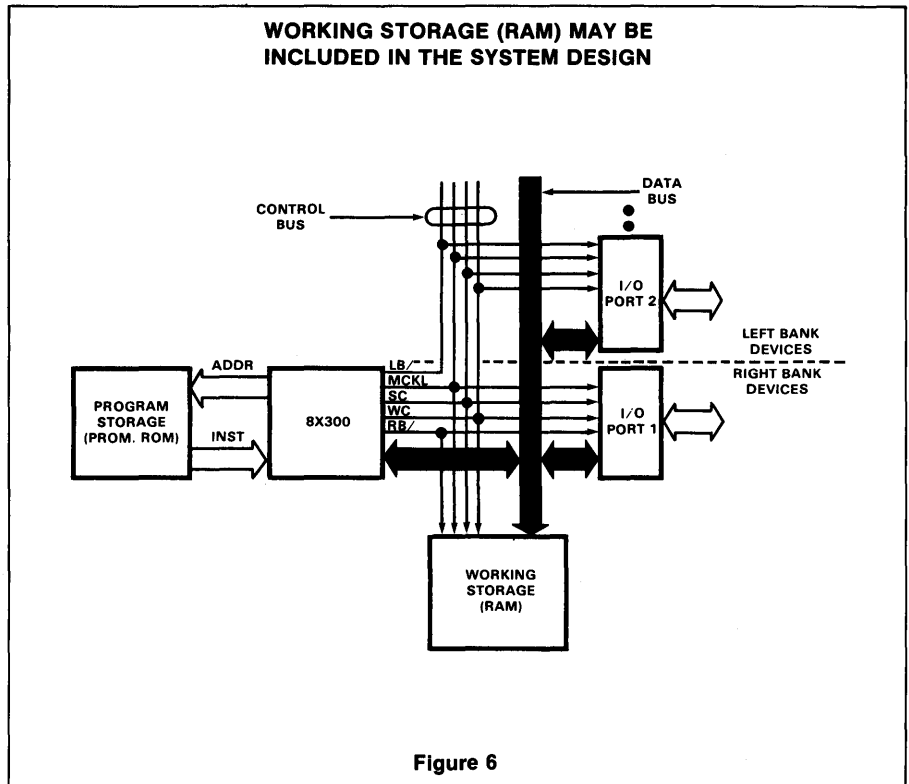


Figure 6

speed on a microcycle basis, to bipolar slice systems.

Instruction Set

The power of the 8X300 architecture is embodied in the instruction set which controls the ALU, rotate, mask, shift and merge functions to provide for various data operations. Each 16-bit instruction word is subdivided into several fields. The arithmetic and logical instructions follow the format shown in Figure 7. There are eight instruction classes each with variations depending upon the operand specifications. These instructions provide for:

- Arithmetic and logic operations—
 - Add, And and XOR
- Data movement—
 - Move and XMIT (transmit)
- Context alteration—
 - JMP (unconditional jump), NZT (test and branch on non-zero) and XEC (execute the instruction at the address specified without program counter alteration)

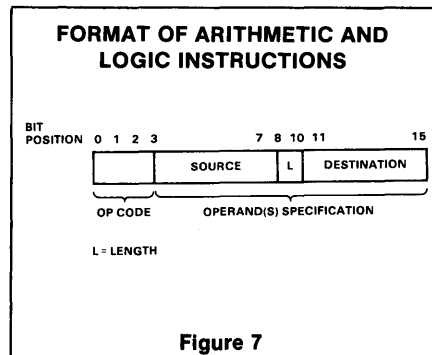


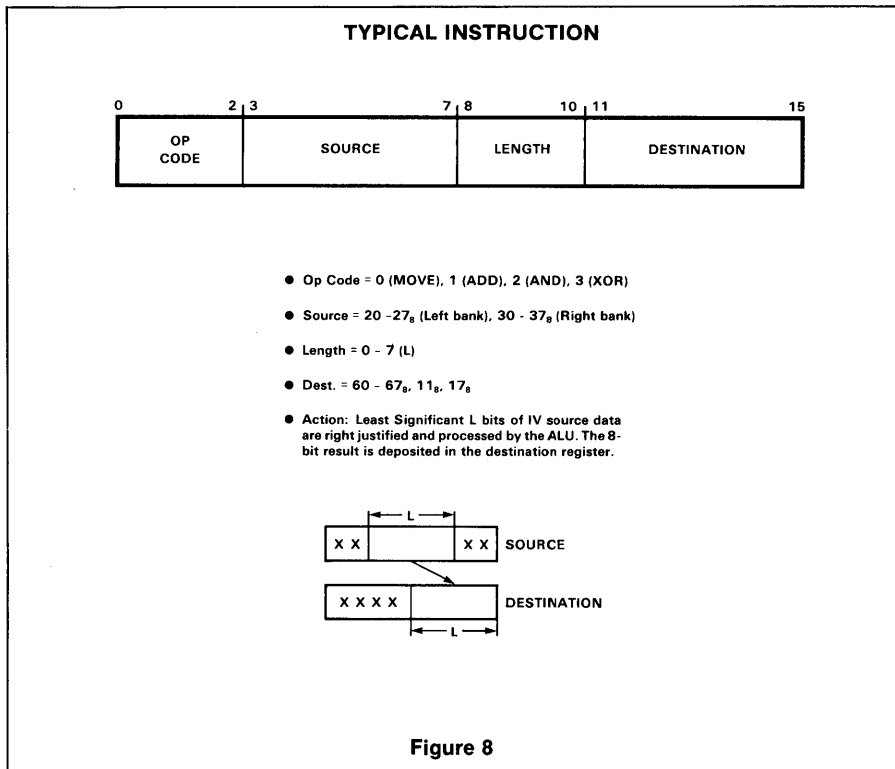
Figure 7

The operand fields specify the source of the data as one of the internal registers or from the IV bus as left bank or right bank, and the destination of the data as one of the internal registers, left bank or right bank or as left bank or right bank addresses. Additionally, these fields specify the length and position of the data which is to be processed. As an example, see Figure 8.

Before going through an example, some features of this instruction should be explained. The first 3 bits are used for the op-code. The 5 source bits contain two separate information groups: The first 2 bits (3 and 4) define the actual source while the next 3 bits (5, 6 and 7) define the least significant bit of the variable length field of the source. These are represented in the diagram by two digits—the first modulo 4 and the second modulo 8. In the example the first digit being 2 selects left bank I/O (right bank = 3).

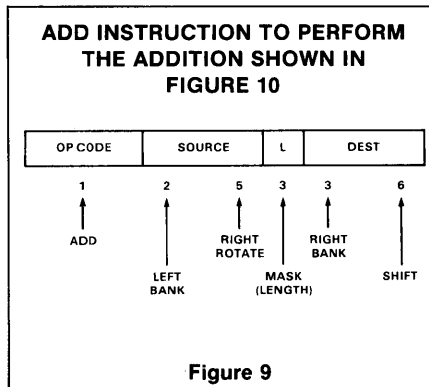
The length of the source data field is specified by the length (bits 8, 9 and 10). The 5 destination bits are represented by two digits—the first modulo 4 and the second modulo 8, as the source. In Figure 8, the destination is an internal register, specified by the first digit being 0 or 1. The actual register is specified by the value of the second digit. These operand fields control the rotate, mask and shift operations as data proceeds through the microprocessor.

Rather than go through the details of the complete instruction set, it is more instructive to proceed with an example which will serve to illustrate what may be done with a single instruction. What shall be done in this



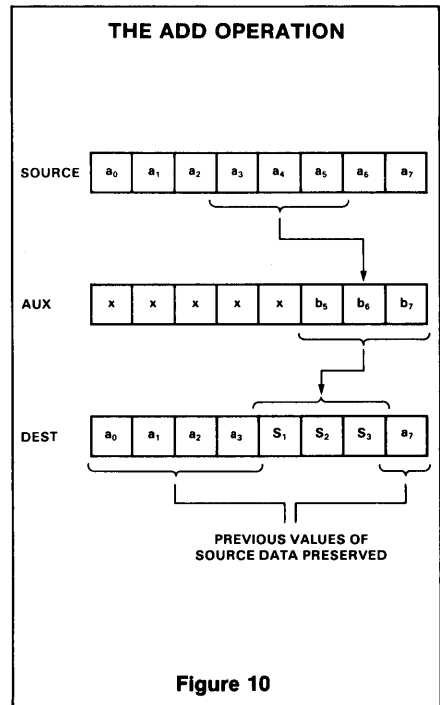
example is to select two I/O ports, add the contents of the AUX register to a specified segment of the source, merge the result with the original data and deposit the result at the destination.

Suppose the source of the data is in IV port, address 5 on the left bank, and the destination address is contained in internal register R3. Further suppose that the AUX register already contains the required value to be added. First, the I/O ports are selected: XMIT 5, IVL (transmit the number 5 to the bus as left bank address). MOVE R3,IVR (move contents of R3 to the bus as a right bank address). The I/O ports have now been enabled using two instructions—500ns total thus far. Now perform ADD LB, RB (add left bank to AUX and deposit in right bank port). The add instruction is shown in Figure 9 where the add operand fields specify the selection of bits throughout rotate and length (mask), and after addition specify the position of merge (shift) in the original data. Although the source, length and destination fields shown here are unique to the MOVE, ADD, AND and XOR instructions, the comments made about these fields also apply to the fields of all the other instructions. Port 5 on the left bank is assumed to contain a_0 - a_7 (Figure 10) and the AUX register is assumed to contain b_0 - b_7 . The source field specifies selecting bits starting with a_5 and the length field specifies taking 3 bits to the left. Thus, a_5 , a_4 and a_3 are masked off and right justified. Note that this requires that only contiguous bits be selected for operation. Next, the selected bits are added to the same length of bits (beginning at the right) from the AUX register.



Thus, the sum (a_3 , a_4 , a_5) + (b_5 , b_6 , b_7) is computed producing a 3-bit sum, S_1 , S_2 , S_3 (and a possible overflow). The destination field of the add instruction then specifies a shift of 1 bit to the left. The shift is made and the 3 bits of the sum are merged with the original source data. Note that the same length specification (3 in the example) applies in source selection, operation and merge functions and is not alterable within one instruction cycle. The destination contains the set of bits shown in Figure 10 after the add operation. Note that the entire set of rotate, mask, add, shift and merge functions took place in one instruction cycle time.

The content of each field can be represented by a set of digits. These digits have a direct relation to the specific operations which the data undergoes as it is directed along the 8X300 internal data paths. The op code for add is 1. This is followed by a two digit source field. The source field is in fact two fields (in this particular case) in which



the first digit, 2, specifies left bank, while the second digit specifies the rotate operation which is to be performed on the incoming data. The L or length field specifies the number of bits to be accepted for ALU operation. This is the mask function specification and selects a quantity of bits counting from the right. The masking operation takes place after the rotate. The destination field, like the source field, specifies the bank or internal register (right bank in this case), and for the bank destination, also specifies the shift operation.

There is one important point to note about the instruction format. Since the fields are easily represented by octal digits and since these digits have a direct relation to the function specified by the field, programming the 8X300 is very easy. Simple mnemonic representation of each of the field specifications, such as ADD for the add function, LB and RB for left bank and right bank and so forth, are easily translated into the octal representation. With this convenience, several hundred lines of program code can be easily generated by hand from the mnemonic representation. Consequently, for small tasks (i.e., less than 500 instructions), an assembler is not essential for efficient programming. A simple conversion is required to generate the actual program memory content.

The above example is typical of what can be done with the MOVE, ADD, AND and XOR instructions. However, the control functions perform differently and are worthy of further attention. Specifically, the XEC (execute) instruction is powerful in that it may be register or I/O vectored. The XEC instruction temporarily changes the contents of the address register for the one instruction cycle following the XEC while allowing sub-

sequent control to be resumed through the program counter. In this light, XEC may be viewed as calling a single instruction subroutine. The XEC instruction performs the vectoring by concatenating the higher order program counter contents with a number determined, in part, by the contents of one of the internal registers or by the content of an I/O port. Thus, the XEC instruction may be used to sequence through a list where the list counter is an internal register, or it may be used to branch to a specific service routine based on some external status reflected in a selected I/O port.

APPLICATIONS

The 8X300 may be exploited in a variety of applications where high speed is required and where the architecture fits the particular requirement. The 8X300 may serve in disk controllers, communications data concentrators and demultiplexers, tape controllers, industrial process controllers, video controllers including entertainment and games, as well as CRT/keyboard terminals, plus a variety of other applications. Principally, the 8X300 affords its greatest service to the user in high speed, relatively sophisticated systems. For example, a low speed MOS processor might be used to control a CRT display and do so economically. However, add the requirement to do data processing for, say, graphics or color display, then the 8X300 becomes increasingly attractive. With 8-bit parallel to serial conversion, the 8X300 may easily process data and directly produce video for alphanumeric display. Generally, one may conclude that the 8X300 serves well where the control processor is required to be in the data path. In controlling computer peripherals, one alternative is to use a single 8X300 processor to control a number of peripherals, as opposed to having one lower speed, less costly processor with separate memory and auxiliary circuits in each peripheral.

Economically, the 8X300 is certainly competitive with the bit slice approach. For those who need the performance, the 8X300 affords a complete, single chip processor at a power consumption of only 1.5 watts in contrast to three to four chips for a bit slice equivalent using nearly 5 watts.

The typical system configuration for the 8X300 is shown in Figure 4. The 8X300 interfaces to the external world through a convenient number of I/O ports connected to the IV bus. Program storage is provided by a suitable ROM or PROM, but RAM could be used here also depending upon the user's application. However, in the more common control applications, the function of the processor is dedicated and, consequently, there is no need to have alterable program storage. This reasoning is also evident in the 8X300 architecture, as exem-

plified in Figure 4. It is clear that there is no direct connection between the program store and the I/O system, as opposed to other microprocessors (the MOS microprocessor in particular) in which instructions are fetched over the same bus on which data and I/O transfers take place.

Figure 4 also emphasizes the compact nature of the processor system. Note that the CPU and program store are realized in as few as three packages (e.g. 8X300 with two 82S115 chips). I/O ports are added as required for the particular system configuration.

Connections to the IV bus are not restricted to the 8T32 type of addressable bidirectional I/O port. Depending upon requirements, a number of devices may be employed. Working storage in the form of RAM may be interfaced directly to the IV bus with an 8T31 or other suitable device used as an address latch. This affords the user temporary storage for data and status information. ROM may also be provided in order to access fixed constants for use by the processor. Examples of such ROM include sine function look-up tables, coordinate translation constants, sensor linearization curves, etc.

Some users have objected to the overhead cost in addressing I/O ports prior to an operation. As in the example used in this paper, 500ns (two instructions) were taken up in selecting I/O ports prior to the major data operation. This is acceptable if ports continue to be accessed for a number of times and thereby reduce the addressing overhead. However, for those who see this as a limitation, there is a convenient alternative. The instruction memory may be extended such that an extra field appears as an additional bus which is applied to each I/O port. Port selection (addressing) would then be done upon instruction fetch. No latch addressable I/O ports would be used, but the normal active-on-address-decode scheme would be employed. The address field may be as wide as required to serve all system I/O ports and if necessary memory. Bus left bank and right bank partitioning would still be used, so the address field would contain two addresses, one for each bank. With this scheme, an entire operation such as described earlier, including the selection of I/O ports, could be accomplished in 250ns.



FEATURES

- **Totally self-contained with keyboard alpha-numeric display, tape reader, TTY output**
- **Dual MicroControllers: one to run instrument, one dedicated to execute user's program**
- **Real time instruction execution**
- **Control of program execution-Halt Single Step and Run Modes**
- **Direct program/register/IV examination and modification through keyboard**
- **Three breakpoint types—Normal, Halt and Insert Instruction**
- **Up to 4K words of high speed read/write program storage**
- **Format compatible with papertape output of MicroController Cross Assembly Program**

DESCRIPTION

The SMS MicroController Simulator, MCSIM, is a hardware development instrument designed to perform in the user's system exactly as an SMS MicroController. It directly supports the SMS 300 (8X300) as well as MicroController prototyping systems. MCSIM gives the user an Interpreter system with a modifiable Program Storage and a Control Panel which together provide a means of entering, running, monitoring, debugging and changing a program. It features ease of use due to its high level interactive display/keyboard and simple operating system. MCSIM allows the user to run his program on-line

in real time. Through the Control Panel, data stored in internal registers, Working Storage and IV Bytes can be examined and modified. An extensive breakpoint capability permits the user to quickly locate program faults.

MCSIM contains two MicroControllers: one for running the instrument and a second totally dedicated to the user's program and prototyping system. This insures that when program development is complete and the design specifications have been met, production systems will perform identically with the prototype.

Operation

MCSIM operation is separated into six modes, each mode consisting of a related set of functions and status displays. Access to a mode is made using one of the six mode selection keys. The currently selected mode is displayed at the extreme left part of the display; the currently selected function is displayed at the extreme right of the display. Selection among the available functions is made using the ▲ and ▼ keys, incrementing and decrementing to the next function. The function selected at last exit is automatically re-selected when the mode is re-entered except for the Breakpoint Mode which selects the Current display.

An operation is generally performed using the following four steps:

1. Select a mode by depressing one of the six Mode Select Keys.
2. Select a function in that mode by positioning the function roll with one of the two Function Select Keys.
3. Set up any necessary conditions, such as entering data or readying papertape in the reader.
4. Activate the function by depressing the Function Acknowledge Key (FCN/ACK).

MCSIM Operator Controlled Functions

Mode	Functions
Manual Examination and alteration of Program Storage	Change Address Store Instruction
Tape Load or dump all or part of Program Storage	Load Verify Begin Punch Address End Punch Address Punch Program Identification
Register Examination and alteration of the Interpreter's internal registers	Change Address Store Octal Data Store Binary Data Complement Overflow
Interface Vector Examination and alteration of the locations on the IV Bus	Display Current Enabled Bytes Change IV Address Store Binary Data in IV Location
Breakpoint Set one of three types of breakpoint for program debugging	Display Currently Active Breakpoint Set Normal (Sync) Breakpoint Set Stop Breakpoint Replace Instruction at Breakpoint
Execute Control and monitor execution of a program	Halt Run Program Insert Instruction Single Step

Specifications

Control Panel

The Mode Select Keys and the Function Select Keys used in combination give the user 26 possible functions to accomplish complete loading and testing of the program. Each Mode Select Key accesses a set of functions, one of which is chosen using the Function Select Keys.

Status Messages

Message displays are used to indicate special conditions which may result from the operation of MCSIM:

POWER ON, MCSIM SYSTEM START
 ???MORE THAN ONE KEY PRESSED
 UNABLE TO READ TAPE. RESTART
 UNDEFINED MEMORY ADDRESS (*octal address*)
 INVALID INSTRUCTION (*instruction code*)
 LOADING INTERRUPTED, RESTART
 V'FYING INTERRUPTED, RESTART
 P'CHING INTERRUPTED, RESTART
 NO VERIFY (*octal address, tape data, memory data*)
 CLEAR RESET LINE TO START RUNNING
 CLEAR HALT LINE TO START RUNNING
 INVALID 8-BIT OCTAL VALUE (*octal value*)
 UNDEFINED IV BYTE ADDRESS (*octal address*)

Data Input/Output

Panel	36 character self scan display for messages and data readout 12 key numeric keyboard for data entry/modification
Tape Reader	120 character/second tape reader for high speed data entry (ASCII format)
TTY	20 ma current loop output for listing of program code on Teletype® terminal

Sync Output

Output pulse whenever address breakpoint is reached.

System Interface

MCSIM is connected to the user's prototyping system via a single ribbon cable. This cable terminates in either a MicroController Simulation Module or a dual in-line plug which is pin for pin compatible with the SMS 300 Interpreter. There are two different simulation modules to exactly match presently available MicroController systems. Selection of the proper input/output connector makes a MCSIM appear to be physically and electrically equivalent to a production MicroController or Interpreter.

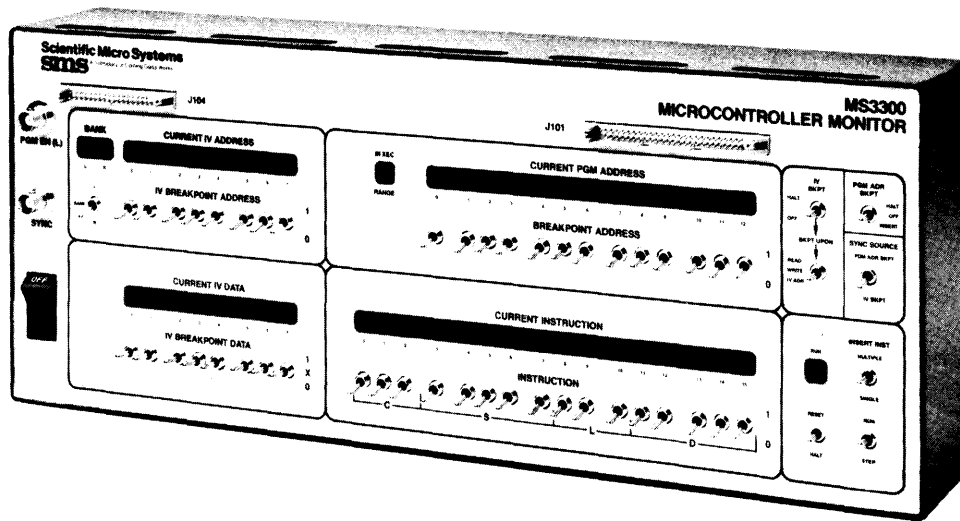
MCCAP

The MicroController Cross Assembly Program (MCCAP) is designed to translate symbolic instructions into object code that can be executed by the SMS 300. This program will run on most computers that have a FORTRAN compiler with a computer word length of at least 16 bits and a random access capability. MCCAP features:

- Symbolic address assignment and references
- Automatic Procedure Handling
- Predefined System Procedures
- Forward References
- Expression Evaluation
- Flexibility to handle MicroController component configurations as well as standard systems
- Generation of object code for MCSIM, the SMS ROM Simulator, or most PROM programmers.

Physical Characteristics

Power	115 V or 230 V ± 10% 50 or 60 Hz ± 10% 350 watts/min. to 750 watts/max. (Power dissipation dependent on the number of Simulation Modules configured in the system)
Dimensions	7 inches high x 17 inches wide x 19 inches deep
Installation	May be used on table or may be installed in standard 19 inch rack with mount adapters
Weight	65 lbs. net, 75 lbs shipping
Ventilation	Air Flow 120 CFM
Environmental	0° to 55°C, Relative Humidity to 90%.



FEATURES

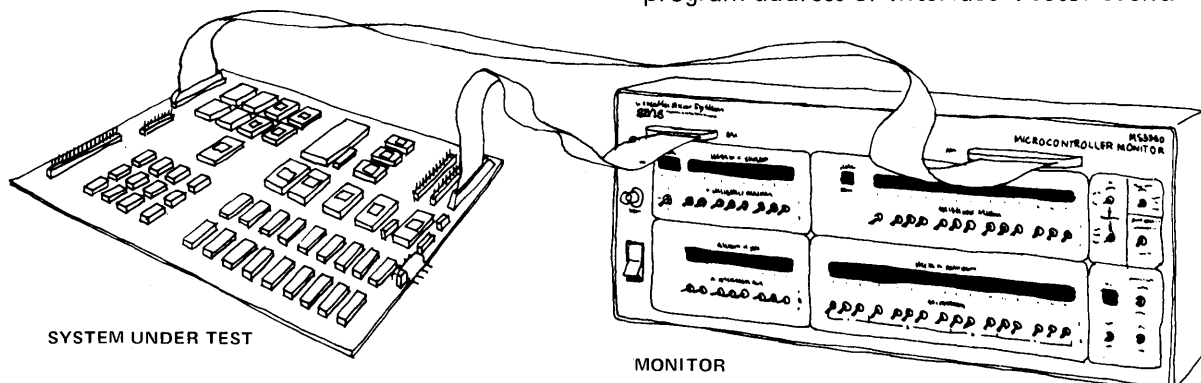
- Real time monitoring instrument for SMS 300
- Totally self contained
- Displays IV address and data
- Displays current program address
- Displays current instruction
- Control of RESET and HALT
- Single step capability
- Real time instruction insertion
- Two real time breakpoints
- Breakpoint output signal

DESCRIPTION

The MicroController Monitor is a self contained debug and maintenance tool for use with all systems containing the SMS 300 (8X300). It provides a control panel allowing an operator to observe, modify, and control program execution in real time permitting system faults to be rapidly traced. The Monitor displays the status of the Address, Instruction, and IV Bus of an Interpreter. Switch registers associated with each display can be used to set breakpoint addresses and enter instructions or data from the Monitor. Program execution can be halted, single stepped, or altered from Monitor controls. The Monitor can be used to insert instructions thereby examining or modifying the contents of

internal registers, Working Store, and IV Bytes. In addition, it can be used to start program execution from any address and enable the operator to set program loops allowing a program segment to be checked independently of normal program flow. Breakpoints on IV data and program address allow the operator to halt program execution or insert instructions in real time.

As shown in the diagram below, the Monitor connects to the system under test through flat ribbon cables (provided) or an adaptor (optional) which plugs into an Interpreter socket. These connections are buffered and present a negligible load to the system. Other features include a Sync output which provides pulses at a selectable program address or Interface Vector event.



Operation

The MicroController Monitor provides the user with two basic modes of system operation, RUN and STEP. When in the RUN mode, indicator LED's provide a continuous real time display of the three major system busses: Address, Instruction, and Data.

Program execution can be halted by one of two actions: depressing the Halt switch or selecting the Halt on Breakpoint function. When halted, it is possible to execute one instruction at a time by pressing the RUN/STEP control to the STEP position. The readout shows the current (static) bus information and the next instruction to be executed.

For checkout purposes it is useful to be able to set a program loop or modify the normal program flow in some other manner. This is accomplished by inserting in real time the instruction set up on the Instruction Switch Register whenever the Program Address Breakpoint is reached. To check for system noise or similar malfunction a separate Insert Instruction switch can be set to the Multiple position, forcing one instruction repetitively. When halted, a single instruction can be inserted by toggling the Insert Instruction switch to the Single position. When an instruction is being inserted, system ROM is temporarily disabled.

To aid in the diagnosis of a malfunction an advanced breakpoint capability is provided. In addition to the Halt and Insert functions when a Program Address Breakpoint is reached, a breakpoint can be generated on the Interface Vector (IV) Bus data or address. This permits program execution to be halted or a Sync pulse generated in response to external data or the contents of RAM. Three modes of IV Breakpoint generation are selectable: Breakpoint on IV Address; Breakpoint on IV Address and IV Write Data; and Breakpoint on IV Address and IV Read Data. The IV Breakpoint Data switches have a "don't care" position to permit generation of a breakpoint on a data subfield. A Sync pulse output is provided on either the IV Breakpoint or Program Address Breakpoint (switch selectable).

Specifications

Controls

RUN/STEP	Sets operation mode, continuous execution (RUN) or single step (STEP).
INSERT INST	Unconditionally causes execution of the instruction in the Instruction Switch Register.
RESET/HALT	Three position switch used to unconditionally HALT program execution or RESET Interpreter's Program Counter to zero.
PGM ADR BKPT	Selects appropriate function to be performed when Program Address Breakpoint is reached: HALT — halt on breakpoint, INSERT — replace normal instruction with instruction set into Instruction Switch Register, OFF — program execution unchanged by breakpoint.
IV BKPT	Selects appropriate function to be performed when IV Breakpoint is reached: HALT — halt on breakpoint, OFF — program execution unchanged by breakpoint.
BKPT UPON	Selects IV Breakpoint on one of three conditions: specified IV address, specified write data at IV address, specified read data at the IV address.

SYNC SOURCE Selects source for output Sync pulse, IV Breakpoint or Program Address Breakpoint. Sync pulse always available at breakpoint regardless of function selected by breakpoint controls.

Switch Registers

INSTRUCTION	Sixteen two position switches for entering instruction to be inserted (during breakpoint or insert instruction control).
BREAKPOINT ADDRESS	Thirteen two position switches to select program breakpoint address.
IV BREAKPOINT ADDRESS	Eight two position switches to select IV byte breakpoint address or Working Store breakpoint address.
BANK	Selects display of LB (Left Bank) or RB (Right Bank) IV address information. Also selects either LB or RB for IV breakpoint.
IV BREAKPOINT DATA	Eight three position switches, 0, 1, X (don't care), to select IV Data Breakpoint.

Displays

RUN	LED display Indicating when Interpreter not halted.
CURRENT INSTRUCTION	LED display of the binary value of next instruction to be executed.
CURRENT PROGRAM ADDRESS	LED display of the binary address of the next instruction to be performed.
IN XEC RANGE	LED display indicating that the value of the Interpreter's program counter and address register may not match because the previous instruction command was "Execute".
BANK	Displays bank addressed by the current instruction (left bank or right bank).
CURRENT IV ADDRESS	Shows the binary address of the currently enabled IV Byte or Working Store location.
CURRENT IV DATA	Binary display of the data on the IV bus.

Outputs

SYNC	Pulse output whenever the switch selected breakpoint occurs.
PGM EN(L)	A low true signal used to enable or disable system program ROM (required only when Interpreter adaptor is used).

Cycle Time

The Monitor can be adjusted to work with any MicroController-based system with a cycle time of 200 ns to 2 μ s.

Physical Characteristics

Power	115 VAC \pm 10% @ 0.75 amp max. 230 VAC \pm 10% @ 0.38 amp max. (optional) 47-63 Hz
Dimensions	7" H x 17.5" W x 3" D
Installation	May be used on table or may be installed in standard 19 inch rack (using optional adaptors).
Weight	10 lbs. net, 13 lbs. shipping
Ventilation	Forced air, 120 CFM.
Environmental	0° to 50°C, relative humidity to 90%
Cables	2 provided, 3 feet long

NOTES

Signetics

a subsidiary of **U.S. Philips Corporation**

Signetics Corporation
811 East Arques Avenue
Sunnyvale, California 94086
Telephone 408/739-7700