AMI MICRO-PRODUCTS

7200 PROCESSOR

GENERAL FEATURES

- Microprogrammable
- 8-Bit or 16-Bit Parallel Processor
- Powerful Microprogram Instruction Set
- Easy ROM/RAM Expansion to 65K Bits
- 540 ns Microinstruction Execution Time
- 45 Registers
- Fully Parallel I/O Bus
- Multiprocessor Bus Capability
- TTL Logic Interface
- Interrupt Features
- +5, -12V, -3V Power Supplies
- 2Ø Non-Overlapping Clocks
- Interval Timer
- General Flags

TYPICAL APPLICATIONS

- Minicomputer
- Terminals
- Test Systems
- Measurement Systems
- Process Control Systems
- Accounting Machines
- Calculators
- I/O Processors

INTRODUCTION

The AMI 7200 CPU is fully parallel, highly flexible bus-organized system. It features a 16-bit Data Exchange Bus concept whereby the I/O devices, memory modules and central processors communicate with each other asynchronously.

The CPU contains 45 registers of which 32 may, under microprogram control, be utilized as a First In - Last Out (push-down) stack or as a file

of 32 general registers. The CPU has incorporated many architectural and logic design features to increase speeds of execution and allow for a wide range of applications.

GENERAL DESCRIPTION

The basic 8-bit 7200 processor consists of three Ion Implanted (I^2) P channel MOS LSI packages:

- Registers and ALU Chip
- Microcontrol Chip
- Microinstruction ROM Chip

The Registers and ALU Chip and the Microinstruction ROM Chip may be configured to form either 4- or 5-chip fully parallel 16-bit machines. The 7200 performs all of the functions commonly found in most minicomputers; the primary difference is the speed of execution which is approximately one third. In many cases the 7200 may be faster; for instance, the 32-byte stack allows for fast subroutine calls executed at microinstruction speeds rather than memory cycle speeds. The block diagram of Figure 1 depicts the basic machine architecture and partitioning. As shown in the figure, the system memory that contains macroinstructions and data may be core, semiconductor RAM, ROM or any combination operating at various rates.

A typical byte oriented application, such as the macroinstruction set described later, would use the complement of registers in the following manner:

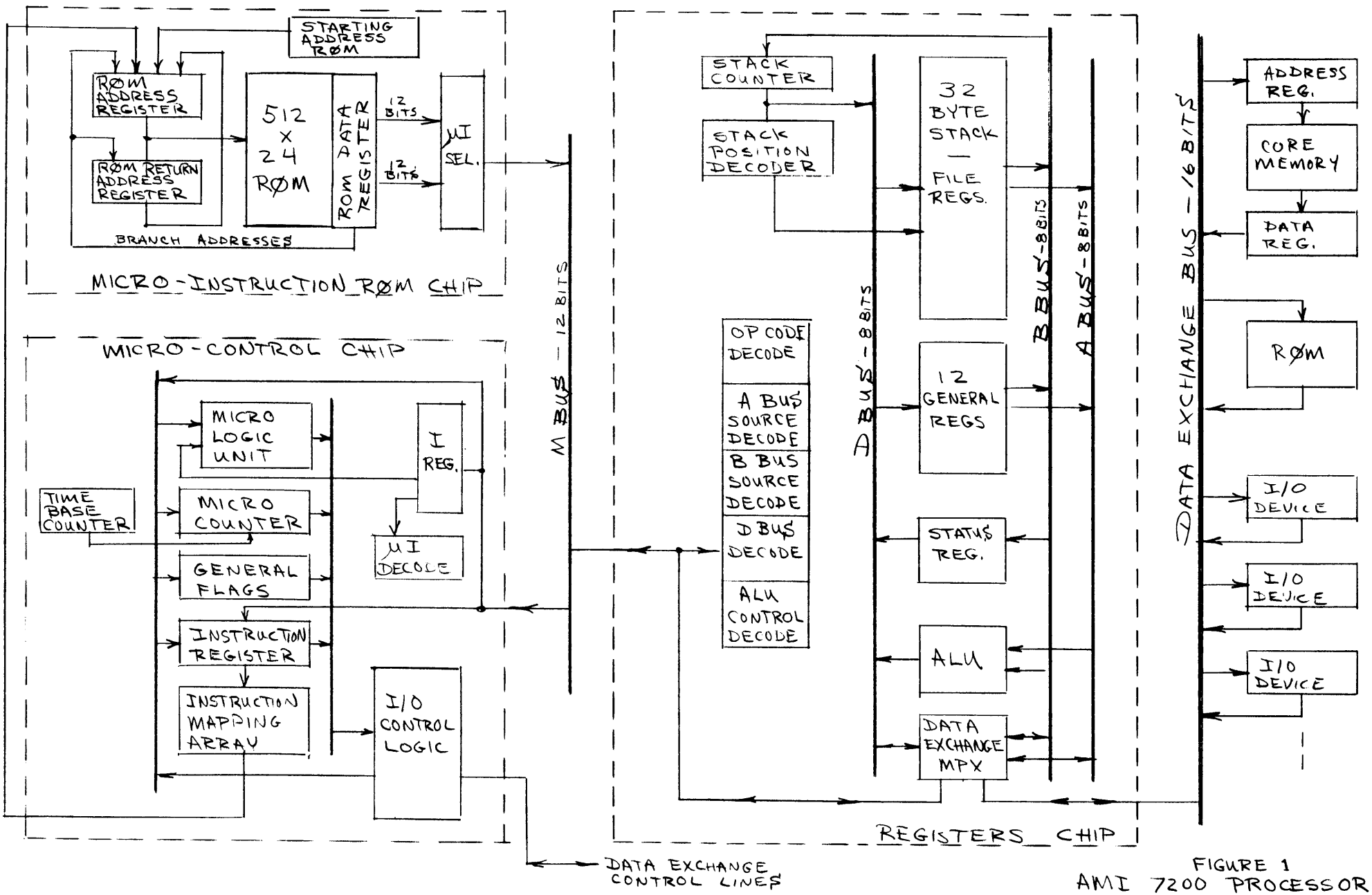


FIGURE 1
AMI 7200 PROCESSOR

4/10/72 G W. SCHULTZ

Number of 8-Bit Registers	Function	Number of Bits per Register
1	Accumulator 1	8
1	Accumulator 2	8
2	Index Register 1	16
2	Index Register 2	16
2	Program Counter	16
1	Temporary Register 1	8
1	Temporary Register 2	8
1	Temporary Register 3	8
1	Temporary Register 4	8
1	Status Register	8
32	Push-down Stack	8

The 32 register push-down stack may be utilized as a set of general registers. A foreground-background system could be implemented by utilizing the 32 registers as two groups of 16 general purpose registers.

The A and B Bus provide the sources of two operands and the D Bus provides the destination path for the ALU result. The ALU chip operates autonomously because it includes all microinstruction decoding logic along with the appropriate timing signals. An 8-bit status register incorporates the following functions:

1. General Flag 1
2. General Flag 2
3. Carry
4. Overflow Condition Flag
5. Zero Condition Flag
6. Minus Condition Flag
7. Interrupt Enable Flag
8. Special Interrupt Flag (Power Fail)

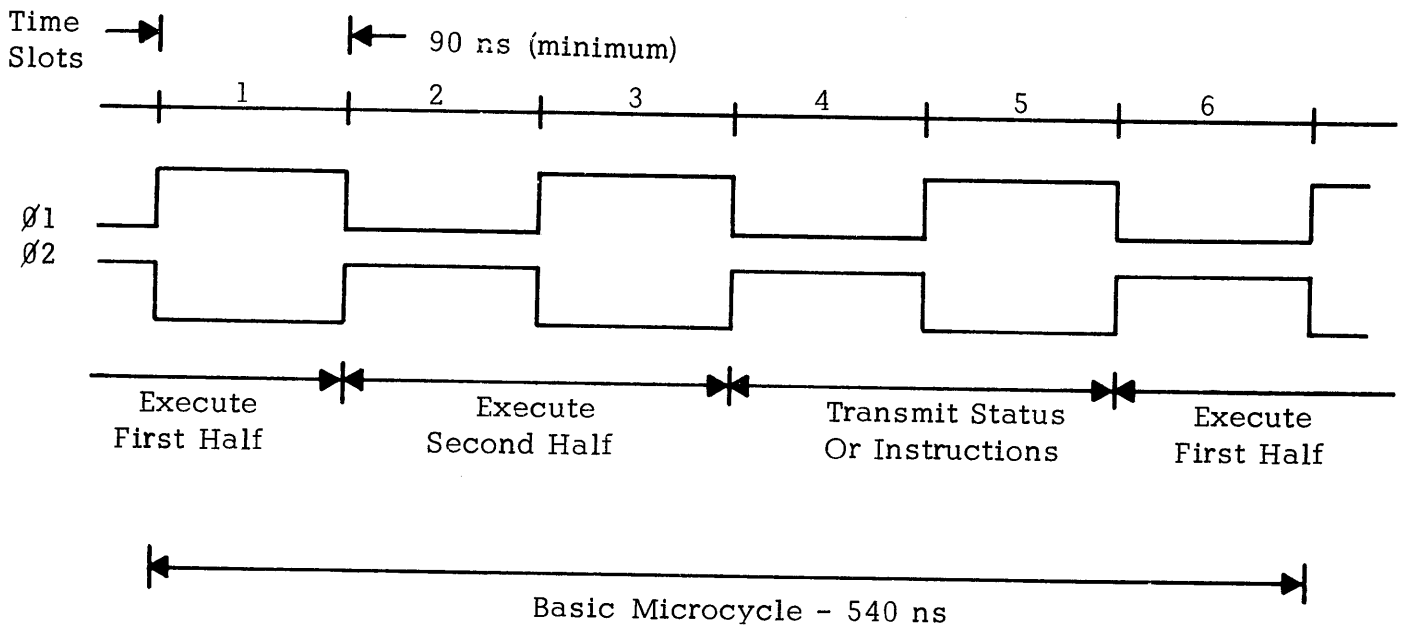
The busses which may also act as temporary storage registers, are connected directly to the ALU. The ALU performs its operations in five time slots; while the ALU is operating on one set of operands, the operands for the next operation are being accessed from the general file registers or stack. After the next operands have been accessed the last result is written into the register as specified by microprogram control.

The A and B busses interface with the external world through the Data Exchange Multiplexers. When addressing external memory the contents of the program counter are placed onto the A and B bus and clocked out to the Data Exchange Bus. When data is to be read from external memory the Data Exchange will drive the incoming data onto the A and B bus. Arithmetic operations can be performed immediately if desired. Macroinstructions are sent from the Data Exchange to the microinstruction M bus for decoding in the Instruction Mapping Array on the Control chip.

The Control section of the processor is implemented by microprogramming techniques. The microprogrammed control program is contained in the Microinstruction ROM which is 512 words by 24 bits. The instruction decoding function is performed by relating the instruction to a starting point in the microinstruction ROM. This decoding function is performed by the Instruction Mapping Array. The large microinstruction ROM and Instruction Mapping Array make it possible to implement virtually any minicomputer instruction set. Twelve general purpose flags are provided on the Micro-control Chip to make it easy to implement calculators and controllers. An interval timer with multiple ranges is also provided to aid in the design of control systems.

SYSTEM TIMING

The 7200 is clocked by a 2-phase non-overlapping clock. These clocks provide precise timing for the execution of each of 6 steps which form the microinstruction execution cycle. All microinstructions execute in the same sequence except when performing I/O from the Data Exchange. The basic microcycle is shown below:



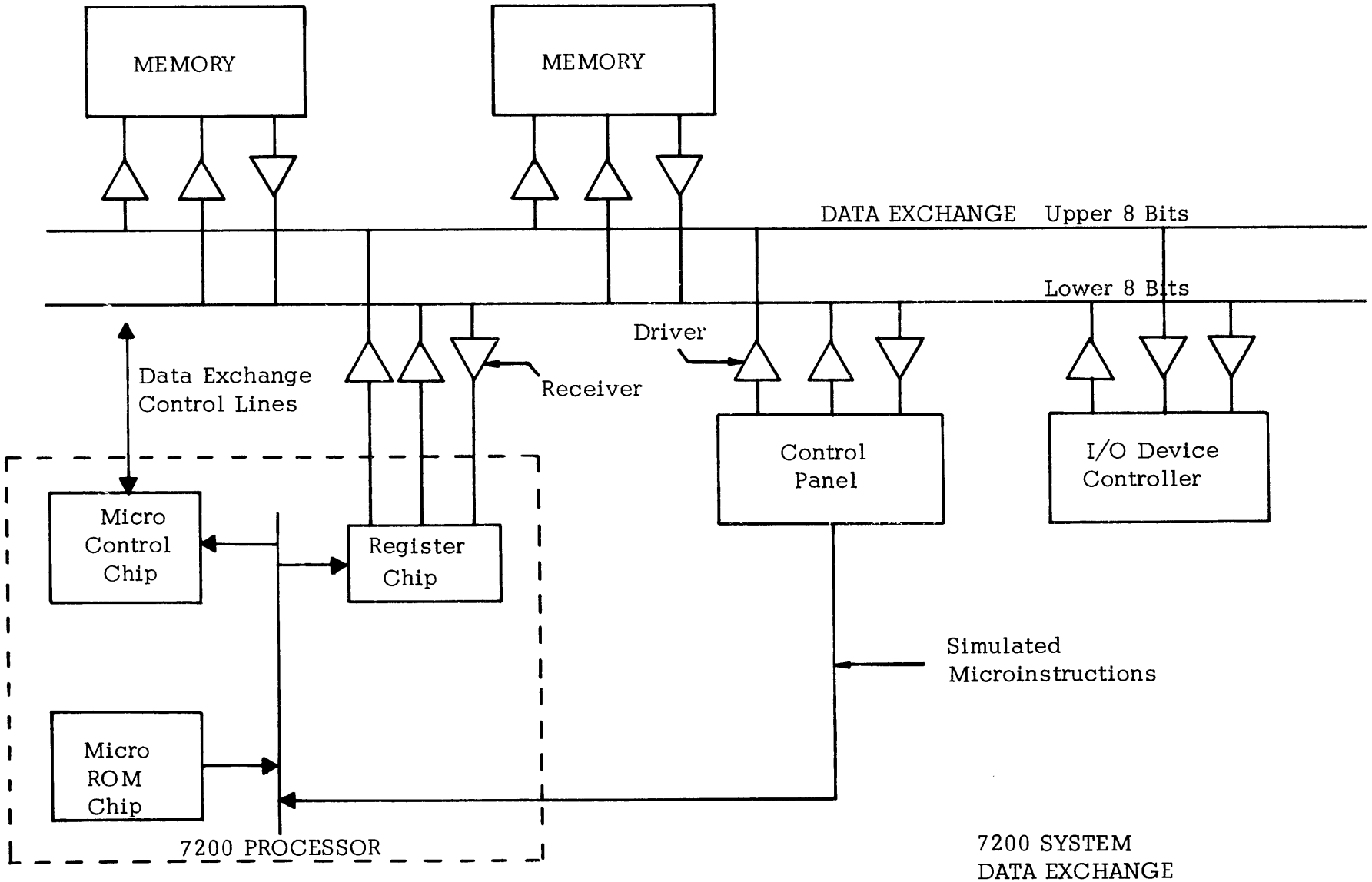
The microinstructions are accessed as 24-bit words but executed one-half at a time. During time slots 6 and 1 the first half is transmitted over the M bus; during 2 and 3 the second half is transmitted. Time slots 4 and 5 are used to receive status or, in the case of fetching, instructions are sent to the Microcontrol Chip.

DATA EXCHANGE

The Data Exchange (DE) Bus shown in Figure 2 is a fully parallel I/O Bus used to interconnect memory, processors, control panel and I/O controllers. Most systems will require only one processor; however, the Bus and its "handshake" lines are designed so that more than one processor may be attached to the Bus simultaneously in order to construct multiprocessor systems. Peripheral controllers may communicate with the processor or directly with memory. It is also possible for peripheral processors to communicate with each other.

The handshake lines are operated asynchronously so that memory and peripherals of varying speeds and responses may be freely mixed on the same Bus structure. These lines have been designed to be used directly to control the Driver/Receiver pairs that form the I/O channels of the Data Exchange in order to reduce the required logic external to the MOS packages.

Table 1 depicts the lines which form the Data Exchange Bus and represents their utilization for various I/O operations. The timing waveforms of the handshake lines for each type of operation are included after the Table.



7200 SYSTEM
DATA EXCHANGE

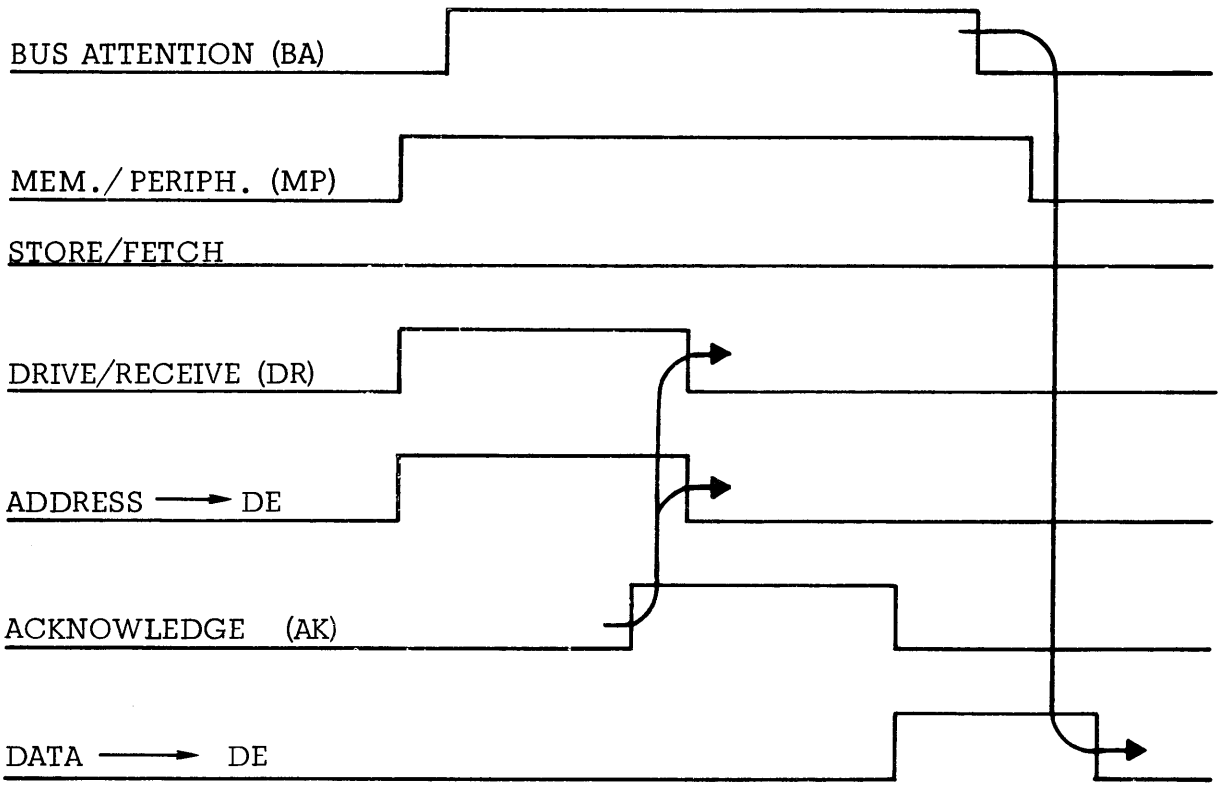
FIGURE 2

DATA EXCHANGE LINES - 8 BIT MACHINES

TABLE 1

Data Exchange Line	Memory Address	Memory Data In/Out	Peripheral I/O	Interrupt Address	Control Panel
Line 15	16 Bit Memory Address		Peripheral Command	Peripheral Command	Control Panel I/O Command
14					
13					
12					
11					
10					
9					
8		8 Bit Data			
7			8 Bit I/O Data	8 Bit Interrupt Address	8 Bit CPU Register
6					
5					
4					
3					
2					
1					
0					
Bus Attention					
Memory/ Peripheral					
Drive/ Receive					
Acknowledge					
Interrupt Attention					
Bus Priority Request					
Got Bus Priority					
Store/Fetch					
Reset					
Run					

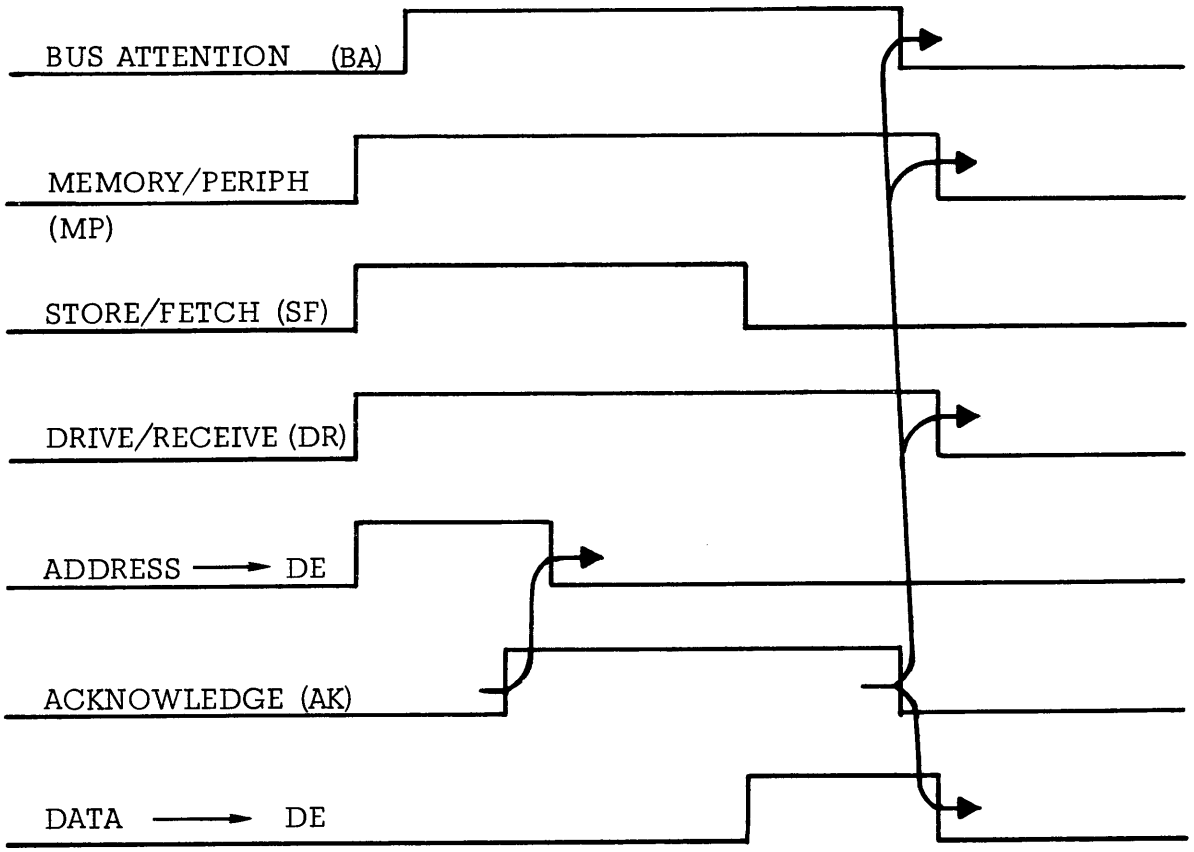
READ MEMORY



The Processor, Control Panel or Peripheral Controller reads memory by executing the following sequence:

1. The processor first drives Memory/Peripheral, Store/Fetch, and 16-bit Memory Address on the Data Exchange and then raises Bus Attention.
2. When Acknowledge is driven by the Memory, the Processor discontinues driving the Memory Address out and lowers Drive/Receive.
3. When Acknowledge is dropped by the Memory, the Processor clocks the data from the Data Exchange and lowers Bus Attention first and then Memory/Peripheral.
4. The Memory senses the dropping of Bus Attention and discontinues to drive data to the Data Exchange.

WRITE MEMORY

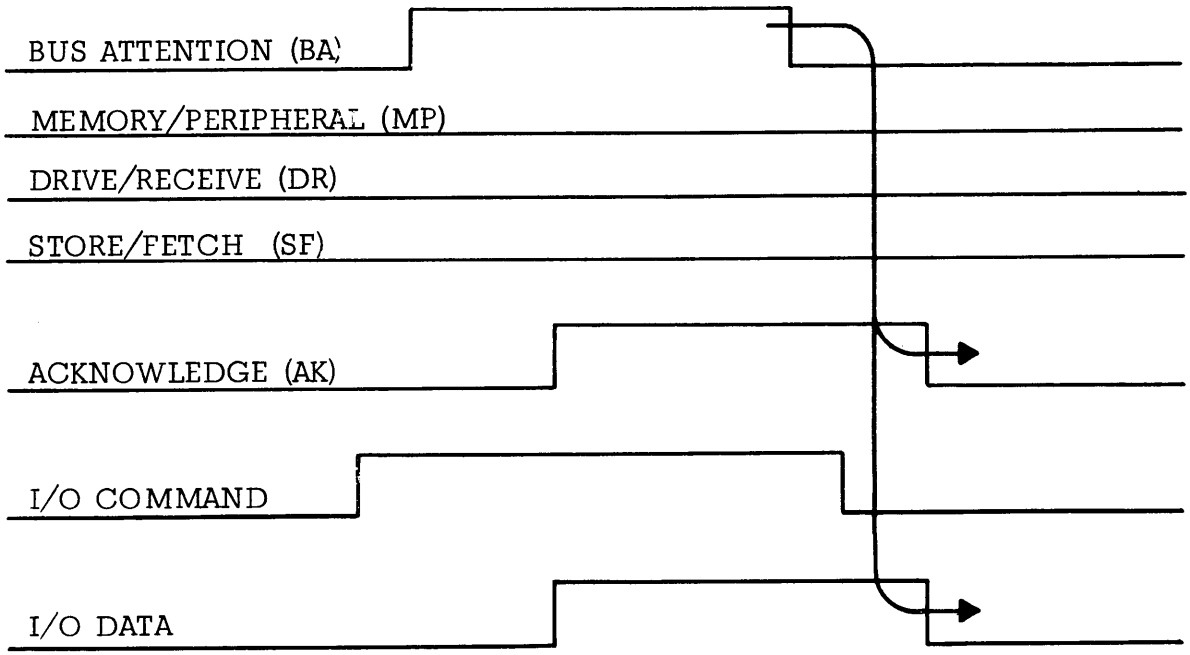


The Processor, Control Panel or Peripheral Controller writes Memory by executing the following sequence:

1. The processor drives Store/Fetch, Memory/Peripheral and Drive/Receive at the same time it places the 16-bit Memory Address on the Data Exchange. Then Bus Attention is raised.
2. When Acknowledge is driven by the Memory, the Processor discontinues to drive the Memory Address to the Data Exchange.

3. After the Processor has placed the data on the Data Exchange it lowers the Store/Fetch line.
4. The Memory senses the Store/Fetch line and clocks the data off the Data Exchange. When it has registered the data it lowers the Acknowledge line.
5. When the Processor senses that the Acknowledge line is no longer driven, it first lowers Bus Attention and then it discontinues to drive the Data Exchange and at the same time lowers the Memory/Peripheral and Drive/Receive.

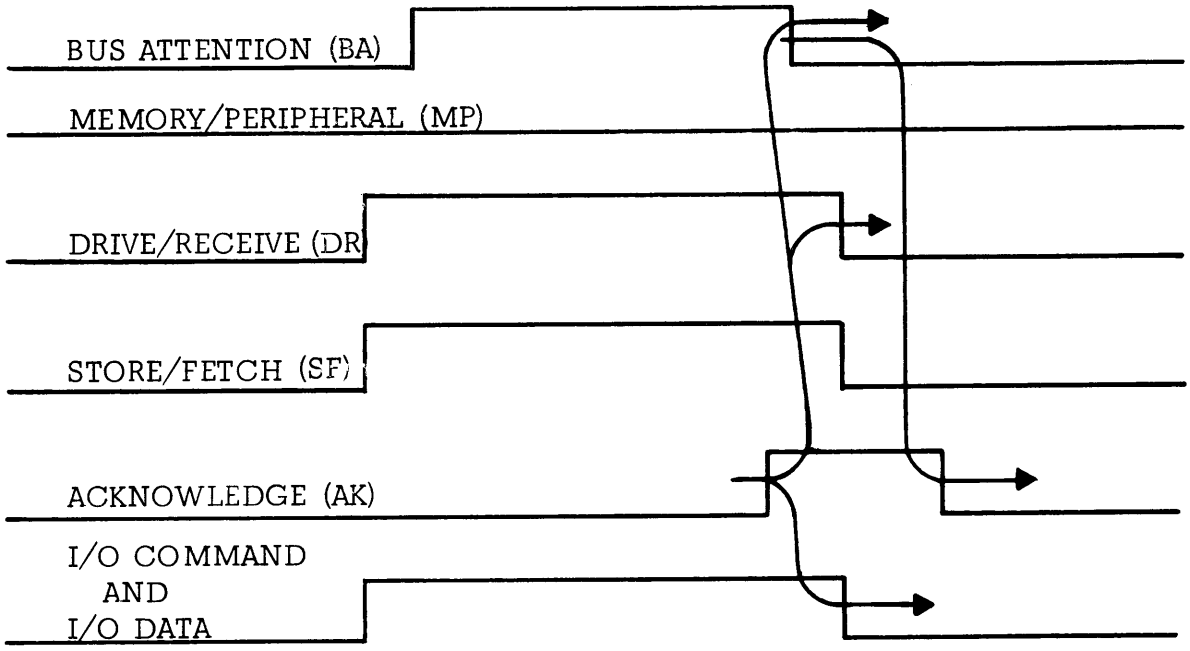
INPUT FROM PERIPHERAL



Input Communication with a Peripheral Controller or Control Panel utilizes the following steps:

1. The Processor places the I/O Command on the upper 8 bits of the Data Exchange and then raises Bus Attention.
2. All the Controllers consider the address portion of the command and the one that decodes the address also looks at the command. The Store/Fetch line can be used to specify Direction of Transfer.
3. The Peripheral drives its output data to the Data Exchange and at the same time drives Acknowledge.
4. When the Processor has registered the data byte it lowers Bus Attention and then removes the I/O Command from the upper 8 bits of the Data Exchange.
5. When the Peripheral Controller senses the loss of Bus Attention it no longer drives Acknowledge.

OUTPUT TO PERIPHERAL



Output Communications with a Peripheral or Control Panel utilize the following steps:

1. The Processor drives Drive/Receive and the I/O Command on the upper 8 bits of the Data Exchange and then raises Bus Attention.
2. The Peripheral decodes the command and registers the 8 bits from the lower portion of the Data Exchange. At the same time the Peripheral which decodes its Address and drives the Acknowledge Line Store/Fetch may be used to specify Direction of Transfer.
3. The Processor senses the Acknowledge line and first removes Bus Attention and then Drive/Receive, Store/Fetch and the I/O Command and I/O Data.
4. When Bus Attention is lowered the Peripheral lowers Acknowledge.

INTERRUPT PROCESSING

Interrupts may be initiated from four sources:

- Interrupt Attention line from the Data Exchange Bus
- Interval Timer
- Power Fail Interrupt line
- Initialization (power up clear) Interrupt Line

Two types of Interrupt schemes are possible utilizing the existing facilities of the Data Exchange.

Probably the simplest approach is to design the Controllers to drive the Interrupt Attention Line to Initiate an Interrupt. The Processor can then respond to the Interrupt by polling each Peripheral Controller in the order of priority desired to determine the source and type of interrupt.

Another way of implementing interrupt capability is to let the Controllers drive Interrupt Attention and at the same time request interrupt priority through a priority chaining network. The Processor will respond to the Interrupt Attention Line by locking up the state of the priority chaining network and performing a Peripheral Input operation as described above.

An I/O Address is set aside for the Interrupt System and all Controllers sense the same address which specifies that the Interrupting Controller with highest priority should place an 8-bit Interrupt Address on the lower 8 bits of the Data Exchange. This operation of gathering the Interrupt Address from the Interrupting Controller works exactly the same as the Peripheral Input process described above. More logic is required in the Controller but a much more efficient Interrupt System is generated.

DIRECT MEMORY ACCESS

Direct Memory Access or communications between peripheral processors will require a simple Memory Bus priority chaining Network to resolve priority among simultaneous accesses to the Data Exchange.

In most systems, the CPU is the lowest priority in the system and must be inhibited from using the bus when a peripheral controller is requesting the Bus. The Bus Priority Request line to the CPU accomplishes this function. When a peripheral requires use of the Bus, it requests priority which in turn causes the Bus Priority Request to be driven to the CPU. The controller must wait until it receives the Got Bus Priority Line before starting an I/O operation. If Bus Attention does not appear, the peripheral may use the Bus to perform any of the operations described above. The peripheral must continue to Drive Bus Priority Request during its I/O operation to hold the CPU off the Bus.

FULL 16-BIT MACHINES

The 3-chip system described thus far, sends a fully parallel 16-bit Memory Address to the Memory and receives an 8-bit byte for each Memory Access. A speed gain of approximately 20% can be achieved by utilizing the same chips and forming a full 16-bit parallel machine. The method of interconnect and number of chips required would depend

somewhat on the complexity of instruction repertoire to be microprogrammed. The primary problem area is in decoding an instruction format where the OP Code information is also contained in the second byte or possibly a third byte.

The first step in configuring a 16-bit wide machine is to parallel the Registers Chip while utilizing one microinstruction ROM chip to drive both. If a more powerful instruction set is required, the micro-instruction ROM chip would be paralleled to form a 5-chip system which can control each byte separately. Figures 3, 4 and 5 depict the complete interconnect diagrams for these three versions of the 7200.

BRANCH ADDRESS FORMAT (Format 4)

The Branch Address format contains two branch addresses to be used alone for immediate branching or with Format 3. When a Branch parameter is specified, one of the branch addresses is used if the branch result is a zero and the other is used if it is a one. Two bits of the format (RC0, RC1) are used to specify control of the RAR (ROM Address Register) and the RRAR (ROM Return Address Register) as follows:

- Load Branch Address 1 into RAR and leave RRAR undisturbed
- Load RAR + 1 into RRAR and Branch Address 1 into RAR
- Load Branch Address 1 into RAR and Branch Address 2 into RRAR
- Load Branch Address 1 into RRAR and increment RAR

MICROPROGRAM MACHINE STATES

An important feature of the Microinstruction ROM Chip is the Starting Address ROM. It may be programmed to supply starting addresses to the Microinstruction ROM for the following operations:

- Return to Instruction Fetch
- Peripheral Interrupts
- Timer Interrupts
- Special Interrupts (Power Fail)
- Power-on Clear (Reset)

The Power-on Clear (Reset) line will cause an immediate stop of the Microinstruction execution and will reset all important control flip-flops. Upon releasing the Clear line the microprogram control unit will begin executing instructions at the address prescribed by the Starting Address ROM.

MICRO-INSTRUCTION ROM CHIP - A03 PN C8817

ØIM, Ø2M, Vcc, VGG, VDD

*MBUS11, *MBUS10, *MBUS9, *MBUS8, *MBUS7, *MBUS6, *MBUS5, *MBUS4, *MBUS3, *MBUS2, *MBUS1, *MBUS0

*GP, RUN, *AK, *RESET, *BR, *SI, *IA, *K, *CI

*IMA8, *IMA7, *IMA6, *IMA5, *IMA4, *IMA3, *IMA2, *IMA1, *IMA0, *BBC, BLK

MICRO-CONTROL CHIP - A02 PN C8816

*MBUS11, *MBUS10, *MBUS9, *MBUS8, *MBUS7, *MBUS6, *MBUS5, *MBUS4, *MBUS3, *MBUS2, *MBUS1, *MBUS0, *CI, *K, DR, BA, MP, SF, *IA, *SI, *BR, *RESET, *AK, RUN

ØIM, Ø2M, Vcc, VGG, VDD

4.7K, VDD (-3)

MBUS

*MBUS11, *MBUS10, *MBUS9, *MBUS8, *MBUS7, *MBUS6, *MBUS5, *MBUS4, *MBUS3, *MBUS2, *MBUS1, *MBUS0

NOTE:
1. THIS DWG. REFERENCES EVERY SIGNAL TO TTL LOGIC LEVELS: '1' = 4 VDC, '0' = 0 VDC
2. * DENOTES COMPLIMENT SIGNAL NOTATION.

*CI, *K

LEAVE OPEN FOR 32 BIT MACH II

*ZERØ, *COUT, *CIN, *ITEM

ØIM, Ø2M, Vcc, VGG, VDD

REGISTERS CHIP - A01 C-8862

*DEH7, *DEH6, *DEH5, *DEH4, *DEH3, *DEH2, *DEH1, *DEH0, *DEL7, *DEL6, *DEL5, *DEL4, RØFF, *DEL3, *DEL2, *DEL1, *DEL0

8T26 (Vcc-16, GND-8)

DE15 (MSB), DE14, DE13, DE12, DE11, DE10, DE9, DE8, DE7, DE6, DE5, DE4, DE3, DE2, DE1, DE0 (LSB)

DATA EXCHANGE BUS

GOT BUS PRIORITY — *GP
DRIVE/RECEIVE — DR
BUS ATTENTION — BA
MEMORY/PERIPHERAL — MP
STORE/FETCH — SF

120Ω Vcc, 1K Vcc, 10K Vcc

DEGP, *DEBA, *DEMP, *DESF

INTERRUPT ATTENTION — *DEIA
SPECIAL INTERRUPT — *DESI
REQUEST BUS PRIORITY — *DEBR
RESET — *DERESET
ACKNOWLEDGE — *DEAK
RUN — DERUN

Ø2, 420Ω Vcc, GND, Vcc

SN74174 OR EQUIV.

CLOCK NETWORK: Ø1, Ø2, Ø1M, Ø2M — MØS CLØCKS (+5, -12, GND)

POWER SUPPLY: Vcc (+5 VDC), VGG (-12 VDC), VDD (-3 VDC), GND

FIGURE 3

REV 2 - 4-13-72

DRAWN	DATE 4-10-72
PROJ ENGR	GAYMOND SCHULTZ
SECT SUPV	
APPROVED	
APPROVED	
DOC. CONT	

AMERICAN MICRO-SYSTEMS, INC. 3800 HOMESTEAD ROAD, SANTA CLARA, CALIFORNIA 95051

7200 PROCESSOR INTERCONNECT 8 BIT MACHINE

CODE IDENT | DRAWING NUMBER | REV

AMI FORM 306-5/0 | SHEET OF

MICRO-INSTRUCTION ROM CHIP - A03

ØIM
Ø2M
Vcc
VDD
VGG
*MBUS11
*MBUS10
*MBUS9
*MBUS8
*MBUS7
*MBUS6
*MBUS5
*MBUS4
*MBUS3
*MBUS2
*MBUS1
*MBUS0
*GP
*IA
*SI
*BR
*RESET
*AK
RUN
*K
*CI

*IMA8 *IMA7 *IMA6 *IMA5 *IMA4 *IMA3 *IMA2 *IMA1 *IMA0 DCK BIK

MICRO CONTROL CHIP - A02

*MBUS11
*MBUS10
*MBUS9
*MBUS8
*MBUS7
*MBUS6
*MBUS5
*MBUS4
*MBUS3
*MBUS2
*MBUS1
*MBUS0
*CI
*K
DR
BA
MP
SF
*IA
*SI
*BR
*RESET
*AK
RUN

ØIM
Ø2M
Vcc
VDD
VGG

MBUS

VDD 4.7K (12 EACH LINES)

VDD 4.7K

REGISTERS CHIP - A01

ØIM
Ø2M
Vcc
VDD
VGG

*MBUS11
*MBUS10
*MBUS9
*MBUS8
*MBUS7
*MBUS6
*MBUS5
*MBUS4
*MBUS3
*MBUS2
*MBUS1
*MBUS0
*ZERO
*COUT
*CIN
*ITEM
*CI
*K

NOTE:
1. DEH0-7 ARE INPUTS ONLY
2. STATUS IS ON THIS CHIP

*DEH7
*DEH6
*DEH5
*DEH4
*DEH3
*DEH2
*DEH1
*DEH0
*DEL7
*DEL6
*DEL5
*DEL4
*DEL3
*DEL2
*DEL1
*DEL0
ROFF

REGISTERS CHIP A01

ØIM
Ø2M
Vcc
VDD
VGG

*MBUS11
*MBUS10
*MBUS9
*MBUS8
*MBUS7
*MBUS6
*MBUS5
*MBUS4
*MBUS3
*MBUS2
*MBUS1
*MBUS0
*CI
*K
*ZERO
*COUT
*CIN
*ITEM

NOTE:
1. DEH0-7 ARE INPUTS ONLY
2. NO STATUS OR D BUS IS RETURNED FROM THIS CHIP DURING TS45

*DEH7
*DEH6
*DEH5
*DEH4
*DEH3
*DEH2
*DEH1
*DEH0
*DEL7
*DEL6
*DEL5
*DEL4
*DEL3
*DEL2
*DEL1
*DEL0
ROFF

8T26 (Vcc - PIN 16, GND - PIN 8, SIGNETICS 8T26)

*DEH7 → DE15
*DEH6 → DE14
*DEH5 → DE13
*DEH4 → DE12
*DEH3 → DE11
*DEH2 → DE10
*DEH1 → DE9
*DEH0 → DE8
*DEL7 → DE7
*DEL6 → DE6
*DEL5 → DE5
*DEL4 → DE4
*DEL3 → DE3
*DEL2 → DE2
*DEL1 → DE1
*DEL0 → DE0

DATA EXCHANGE BUS

DR

10K Vcc

8T26

120Ω Vcc → DEGP
1K Vcc → *DEBA
*DEMP
*DESF

*DEIA
*DESI
*DEBR
*DERESET
*DEAK
DERUN

CLOCK NETWORK

ØI
Ø2
ØIM } MOS CLOCKS
Ø2M }

POWER SUPPLY

Vcc (+5VDC)
VDD (-3VDC)
VGG (-12VDC)
GND

120Ω ×6

Ø2 → CK
Vcc → 16
GND → 8
D1 D2 D3 D4 D5 D6
SN74174 OR EQUIV.
CLEAR
Q1 Q2 Q3 Q4 Q5 Q6
420 Vcc

REV2 - 4-13-72

DRAWN	DATE 4-13-72	AMERICAN MICRO-SYSTEMS, INC. 3800 HOMESTEAD ROAD SANTA CLARA, CALIFORNIA (408) 246-4330
PROJ ENGR GAY MOND SCHULTZ		7200 PROCESSOR INTERCONNECT DWG. 4 CHIP-16 BIT VERSION
SECT SUPV		
APPROVED		
APPROVED		
DOC. CONT		CODE IDENT / DRAWING NUMBER / REV
		AMI FORM 308-8/0 SHEET 1 OF 1

FIGURE 4

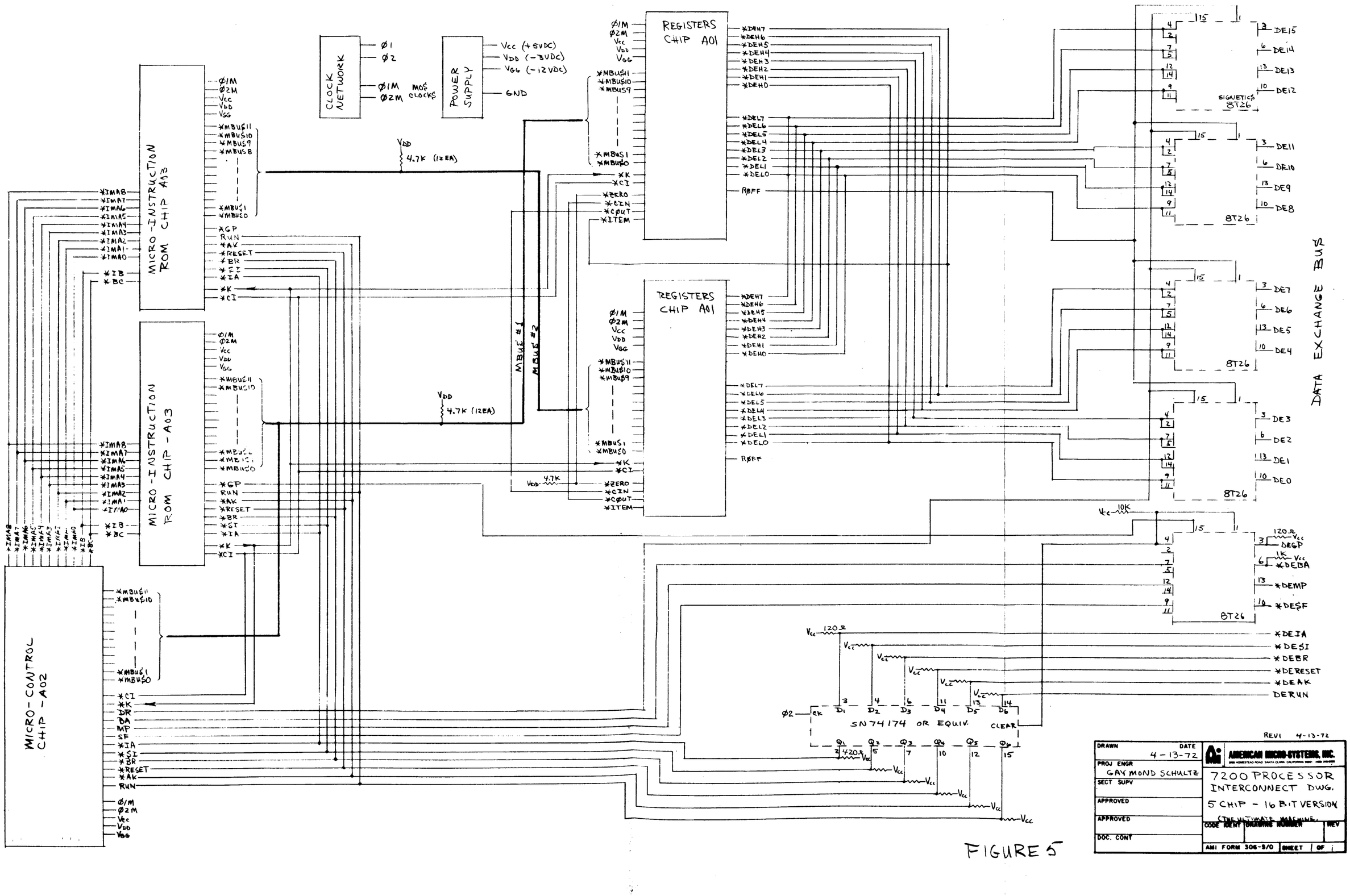
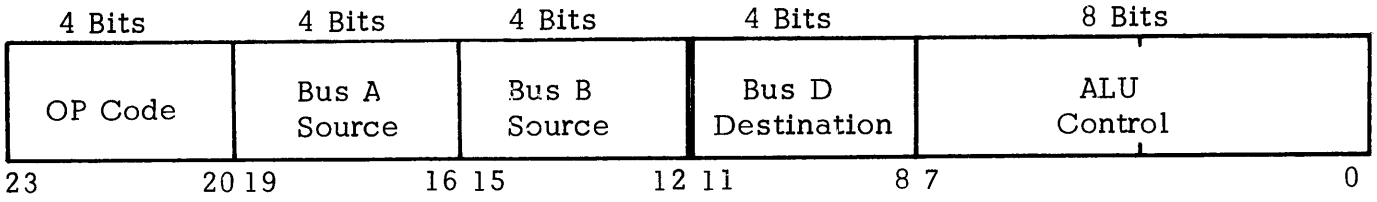

CLOCK NETWORK
Ø1
Ø2
ØIM
Ø2M
MOS CLOCKS
POWER SUPPLY
Vcc (+5VDC)
VDD (-3VDC)
VGG (-12VDC)
GND
MICRO-INSTRUCTION ROM CHIP A03
MICRO-INSTRUCTION ROM CHIP - A03
MICRO-CONTROL CHIP - A02
REGISTERS CHIP A01
REGISTERS CHIP A01
4.7K (12EA)
MBUS #1
MBUS #2
SIGNETICS 8T26
8T26
DATA EXCHANGE BUS
SN74174 OR EQUIV.
120Ω
10K
1K
REV1 4-13-72
DRAWN
DATE 4-13-72
PROJ ENGR GAYMOND SCHULTZ
SECT SUPV
APPROVED
APPROVED
DOC. CONT
AMERICAN MICRO-SYSTEMS, INC.
7200 PROCESSOR INTERCONNECT DWG.
5 CHIP - 16 BIT VERSION
(THE ULTIMATE MACHINE)
CODE IDENT
DRAWING NUMBER
REV
AMI FORM 306-S/O
SHEET
OF

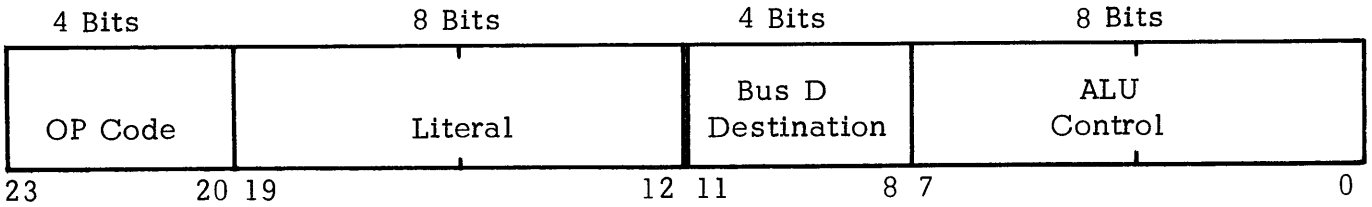
FIGURE 5

7200 MICROINSTRUCTION FORMATS

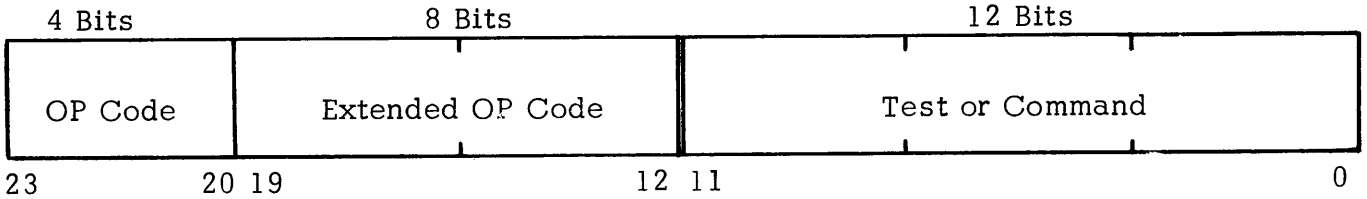
CONTROL FORMAT (Format 1)



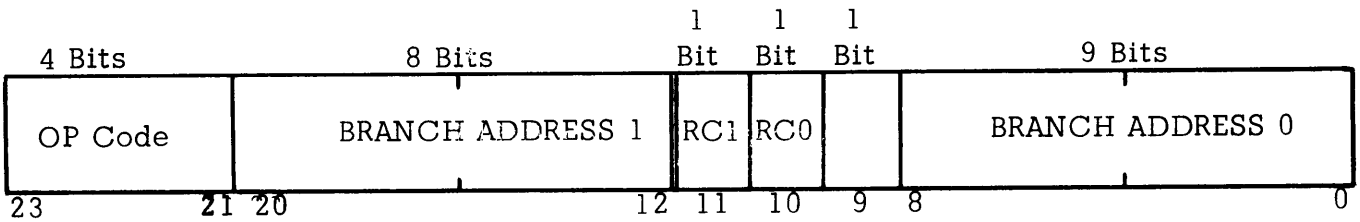
LITERAL FORMAT (Format 2)



TEST AND COMMAND FORMAT (Format 3)



BRANCH ADDRESS FORMAT (Format 4)



CONTROL FORMAT (Format 1)

The Control format is used to configure the registers on the Register Chip in executing register movements and arithmetic operations. The OP Code field specifies the format type and allows for instruction decoding, return to Instruction Fetch and RAR/RRAR Control. Source fields of the A and B Bus indicate one of 16 sources to be clocked onto these busses. Bus D destination represents the register used to store the result of a register movement or arithmetic operation. The 32-byte stack is popped each time it appears as a source and pushed when it appears as a destination. Thus, if it is desirable to simply read the top of the stack without popping it; the Stack code would appear in both source and destination fields. The ALU control field specifies control of the arithmetic unit as well as shifting operations and I/O operations.

LITERAL FORMAT (Format 2)

The Literal format allows an 8 bit byte of any configuration to be clocked onto the B Bus where it may be loaded into the destination specified by the Bus D destination field. If no destination is specified, the literal remains on the D Bus where it may be utilized by gating it to the A or B busses in the next microinstruction. The A Bus will automatically contain the last ALU (D Bus) results which may be utilized in conjunction with the literal.

TEST AND COMMAND FORMAT (Format 3)

The Test and Command format utilizes an extended OP Code field to further specify the setting, resetting and testing of the microcontrol registers:

- Status Register
- General Flags
- Instruction Register
- Microcounter/Timer
- Interrupt System and Data Exchange Control Lines

The other four operations are tested at Return to Instruction Fetch time. The order of priority is shown above, with Power-on Clear having highest priority.

This technique saves space in the Microinstruction ROM, gains speed and allows the flexibility of microprogramming the machine to react exactly as desired based on specific system operational requirements.

MACROINSTRUCTIONS

INTRODUCTION

A wide variety of macroinstruction sets may be designed to meet custom design requirements for a relatively low development cost. On the other hand, both development costs and time can be saved by using an off-the-shelf product such as that described here. Hopefully many users will find the instruction set described below adequate or close to that required.

The instruction set is a two accumulator byte-oriented structure. Most arithmetic and logical operations are performed on 8-bit bytes but provision is also made for limited 16-bit bytes. The register array was listed above, and includes two 16-bit index registers which allow easy access to any two blocks of addresses. Bit testing is facilitated by special Test and Branch instructions which eliminates the need for a flag register. Subroutine and interrupt returns are stored in the Push Down Stack, the head of which may be used as an Autoindex Register to facilitate subroutine parameter pick-ups. The instruction set allows for memory expansion to 65K bytes with all locations simultaneously addressable.

ADDRESSING SCHEMES

Several different forms of addressing are used as described below:

Immediate Addressing

In Immediate Mode, the Effective Address (EA) of the operand is contained in the program counter. After fetching each byte, the program counter is incremented by one.

Indexed Addressing

Indexed Address Mode specifies either of the two index registers as a base. A second byte of instruction is added to the specified index register to form the EA.

Autoindexed Addressing

In Autoindex mode the top (last-in) two bytes on the General Stack, referred to as the Stack Head (SH), are used as the EA. After fetching or storing each byte the Stack Head is incremented by one.

Indirect Addressing

Indirect Addressing may be specified in conjunction with the above modes. The EA, calculated as described above, is used to fetch two bytes which become the EA of the operand.

Relative Addressing

Branch and Bit Test instructions always use relative addressing to specify the address at which program execution is to resume if the branch test is successful. In relative addressing a second byte of instruction is fetched and is regarded as a signed, 8-bit two's complement quantity (sign bit plus 7 mantissa bits). This byte, called the offset, is then added to the program counter to form the effective address. Relative addressing allows any location within -128 to +127 bytes of the location following the offset byte of the Branch instruction to be specified as the Branch Address.

INSTRUCTIONS

The instruction formats require from 1 to 5 bytes of memory for OP Code and address or data. The symbols and abbreviations used and a summary of the instructions are shown on the following tables.

SYMBOLS AND ABBREVIATIONS

Registers and Flags

RØ	Contents of the RØ accumulator
R1	Contents of the R1 accumulator
XØ	Contents of the XØ index register
X1	Contents of the X1 index register
PC	Contents of the Program Counter
GS	General Stack
SH	Contents of the top byte of the GS
SR	Contents of the Status Register
I	Interrupt Enable bit of the SR
C	Carry bit of the SR
M	Minus bit of the SR
Z	Zero of the SR

INSTRUCTION SUMMARY

Memory Reference Instructions

M Nemonic	Name
LDXØ	Load XØ
LDX1	Load X1
LDRØ	Load RØ
LDR1	Load R1
STRØ	Store RØ
STR1	Store R1
ADRØ	Add to RØ
ADR1	Add to R1
SBRØ	Subtract from RØ
SBR1	Subtract from R1
ANRØ	AND to RØ
ORRØ	OR to RØ
ORR1	OR to R1
PUM	Push Memory
PUMD	Push Memory Double
POM	Pop Memory
POMD	Pop Memory Double
JMP	Jump
PJMP	Push Jump

Bit Test Instructions

TBBO	Test Bits and Branch on Ones
TBBZ	Test Bits and Branch on Zero

Branch Instructions

IBZXØ	Increment and Branch if Zero XØ
IBZX1	Increment and Branch if Zero X1
IBNXØ	Increment and Branch on Non-Zero XØ
IBNX1	Increment and Branch on Non-Zero X1
DBZXØ	Decrement and Branch if Zero XØ
DBZX1	Decrement and Branch if Zero X1
DBNX0	Decrement and Branch if Non-Zero XØ
DBNX1	Decrement and Branch if Non-Zero X1
BRZ	Branch-on Z (Zero)
BNZ	Branch-on Not Z
BRC	Branch-on C (Carry)

Branch Instructions (Continued)

BNC	Branch-on Not C
BRM	Branch-on M (Minus)
BNM	Branch-on Not M
BRI	Branch-on I (Interrupt Enable)
BRN	Branch
BTOC	Branch-on Time Out and Clear TO and TE
BPUC	Branch-on Power Up and Clear PU
BPD	Branch-on Power Down

Operate Group

M Nemonic	Name
PUXØ	Push XØ
PUX1	
POXØ	Pop XØ
POX1	
SWXØ	Swap SH with XØ
SWX1	
CLRØ	Clear RØ
CLR1	
CMRØ	Complement RØ
CMR1	
NGRØ	Negate RØ
NGR1	
RLRØ	Rotate left RØ
RLR1	
RRRØ	Rotate Right RØ
RRR1	
PURØ	Push RØ
PUR1	
PORØ	Pop RØ
POR1	
INRØ	Increment RØ
INR1	
DCRØ	Decrement RØ
DCR1	
ACRØ	Add Carry to RØ
ACR1	
SCRØ	Subtract Carry from RØ
SCR1	
SCRØ	Swap Digits of RØ
SDR1	
ASRØ	Add SH to RØ
ASR1	

Operate Group (Continued)

M Nemonic	Name
SSRØ	Subtract SH from RØ
SSR1	
RLD	Rotate Left Double
RRD	Rotate Right Double
SWR	Swap R1 & RØ
INSH	Increment SH
DCSH	Decrement SH
PUSR	Push SR
POSR	Pop SR
CLC	Clear Carry
ION	Interrupt Enable ON
IOF	Interrupt Enable OFF
RTS	Return from Subroutine
RTI	Return from Interrupt
CCAL	Co-Call
INXØ	Increment XØ
INX1	Increment X1
DSHT	Decrement Stack Head by Two

IOT Instructions

M Nemonic	Name
STO	Status Out
STI	Status In
DAO	Data Out
DAI	Data In

Interval Timer Instructions

LITI	Load Interval Timer and Interrupt on Time-Out
LIC	Load Interval Timer

Added Instructions

- Multiply
- Divide
- Search Memory
- Extract Decimal Digit