Architecture of microcontroller system

by MICHAEL LICCARDO
Scientific Micro Systems
Mountain View, California

SYSTEM OVERVIEW

A microcomputer designed for control

The SMS MicroController is a microcomputer designed for control. It features:

Execution speed

- 300 nanosecond instruction execution time.
- Direct address capability—up to 4096 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 two instruction cycle times.
- MicroController 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 MicroController 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 MicroController ideal in switching, controlling, and editing applications.

Interface simplicity and expandability

- Direct connection to TTL (3-state) I/O (Open Collector outputs are optional).
- I/O expandable to 224 connection points with storage buffer at each point.
- User defined data flow direction with each group of 8 I/O points.

External device signals may be accessed with minimal interface circuitry. The MicroController input/output system provides a direct register interface to external devices. Unlike classical minicomputer bus structures, external devices do not require the logic for providing addresses to the input/output system. The address of an external device is determined programmatically within the MicroController.

Direct processing of external data

Data from external devices may be processed (tested, shifted, added to, etc.) without first moving them to internal storage. This is because its I/O system appears to the MicroController as a set of internal registers. 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 MicroController 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. Data storage for the MicroController is implemented with read/write memory because data in control and other real time applications is dynamic and variable.

High density packaging and reliable operation

- The MicroController is implemented completely with LSI circuits.
- The MicroController CPU consists of a single integrated circuit.
- Single +5.0 volt power supply operation.

The MicroController is provided packaged on one of four basic boards. The smallest packaging scheme is the System 10 which is 6.875 inches by 2.675 inches. This board can accommodate CPU, 2K words of program storage, and 32 I/O points. The largest package, the System 40, is 6.875 inches by 13.475 inches and accommodates a fully expanded system consisting of CPU, 4K words of program storage, 224 I/O points and 256 bytes of read/write data storage.

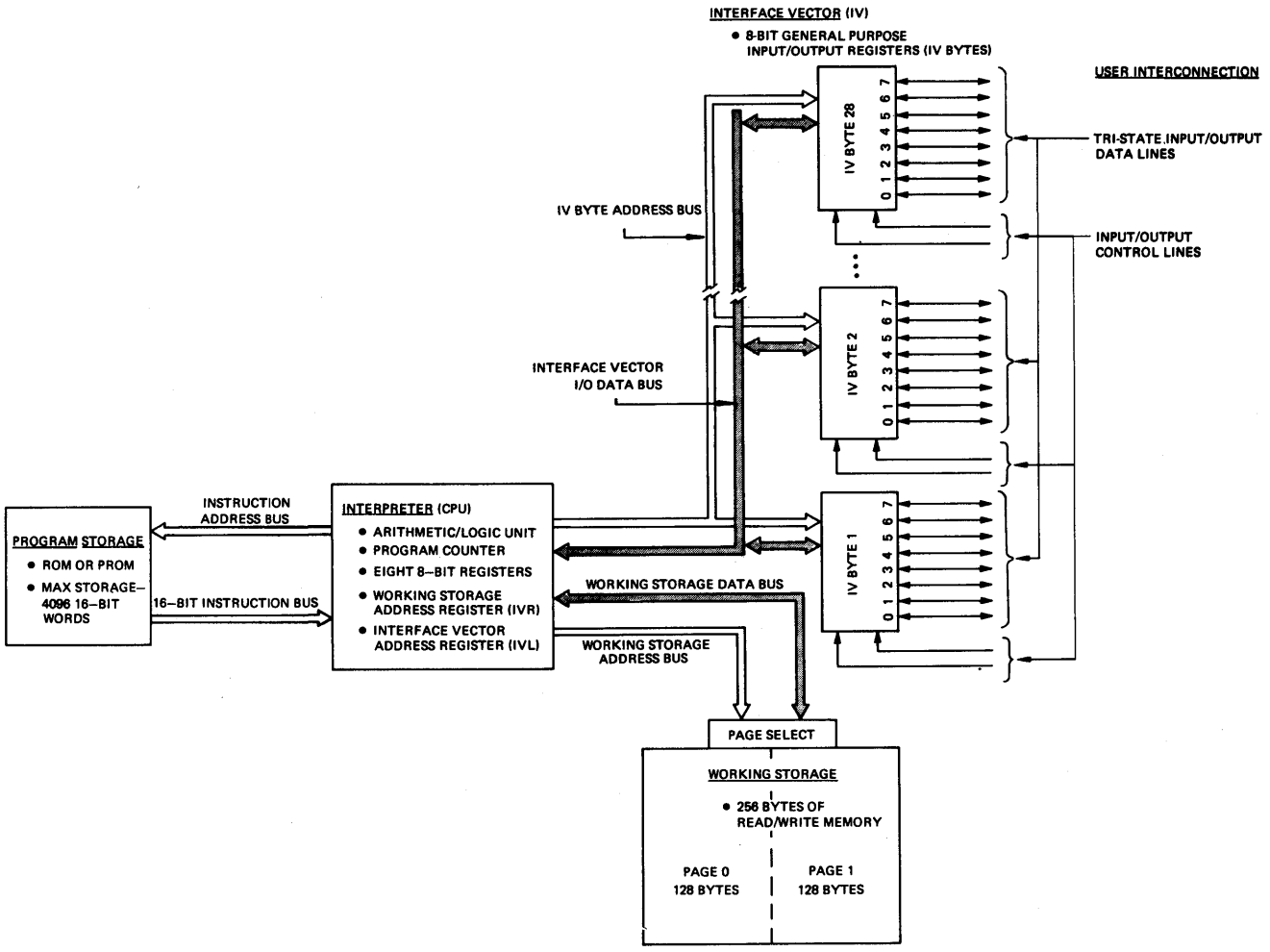


Figure 1—Microcontroller system diagram

MicroController functional components

The MicroController is a complete microcomputer system consisting of:

- A central processing unit called the Interpreter.
- Read-only program storage.
- Optional read/write data storage called Working Storage with variable field address of from 1 to 8 bits.
- A complete input/output system called the Interface Vector.

The MicroController System is shown in Figure 1.

Figure 2 illustrates the MicroController architecture. The MicroController CPU contains an Arithmetic Logic Unit (ALU), Program Counter, Interface Vector Address Register (IVL), and Working Storage Address Register (IVR). Eight 8-bit general purpose registers are provided, including seven working registers and an Auxiliary register which performs as a working register and also provides an implied operand for many instructions. The MicroController registers are shown in Figure 2 and are summarized below:

Control Registers include:

Instruction—A 16-bit register containing the current instruction.

Program Storage Address (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.

IV Byte Address (IVL)—An 8-bit register containing the address of the current byte being accessed from the Interface Vector. IVL is under program control.

Working Storage Address (IVR)—An 8-bit register containing the address of the current byte being accessed from Working Storage. IVR is under program control.

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. Arithmetically treated as 2°.
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 provides eight instructions.

The 16-bit MicroController instructions are stored in 512 to 4096 words of read-only Program Storage. Program Storage can be implemented with either mask coded ROMs (Read-Only Memory) or PROMs (Programmable Read-Only Memory).

The input/output system, called the Interface Vector, serves as the data path over which information is transferred into and out of the MicroController. The basic elements of the Interface Vector are:

- The general purpose 8-bit input/output registers, or Interface Vector (IV) Bytes, whose tri-state data path serves as the connection points to the user system.
- The IVL register which contains the address of the IV Byte currently being accessed.
- Variable field selection which permits 1 to 8-bit field access of a selected IV Byte in a single instruction.

The Interface Vector eliminates the need for costly interface logic and presents a simple, well-defined interconnection point to the user system.

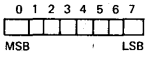
Working Storage is available as an option that provides 256 bytes of read/write memory for program data or input/output data buffering. Working Storage consists of:

- 256 8-bit bytes of read/write memory organized as two pages (banks), Page 0 and Page 1, of 128 bytes each.
- The Working Storage address register, IVR which holds the address of the byte currently accessed in either Page 0 or Page 1, depending on the state of the Page Select Register.
- The Page Select Register, addressed through IVR, is a single bit register used to select Page 0 or Page 1 of Working Storage.
- Variable Field Select which permits 1 to 8-bit field transfers to or from an addressed Working Storage byte in a single instruction.

MICROCONTROLLER INSTRUCTION SET

The MicroController has a repertoire of eight instructions 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 executed completely in 300 nanoseconds.

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



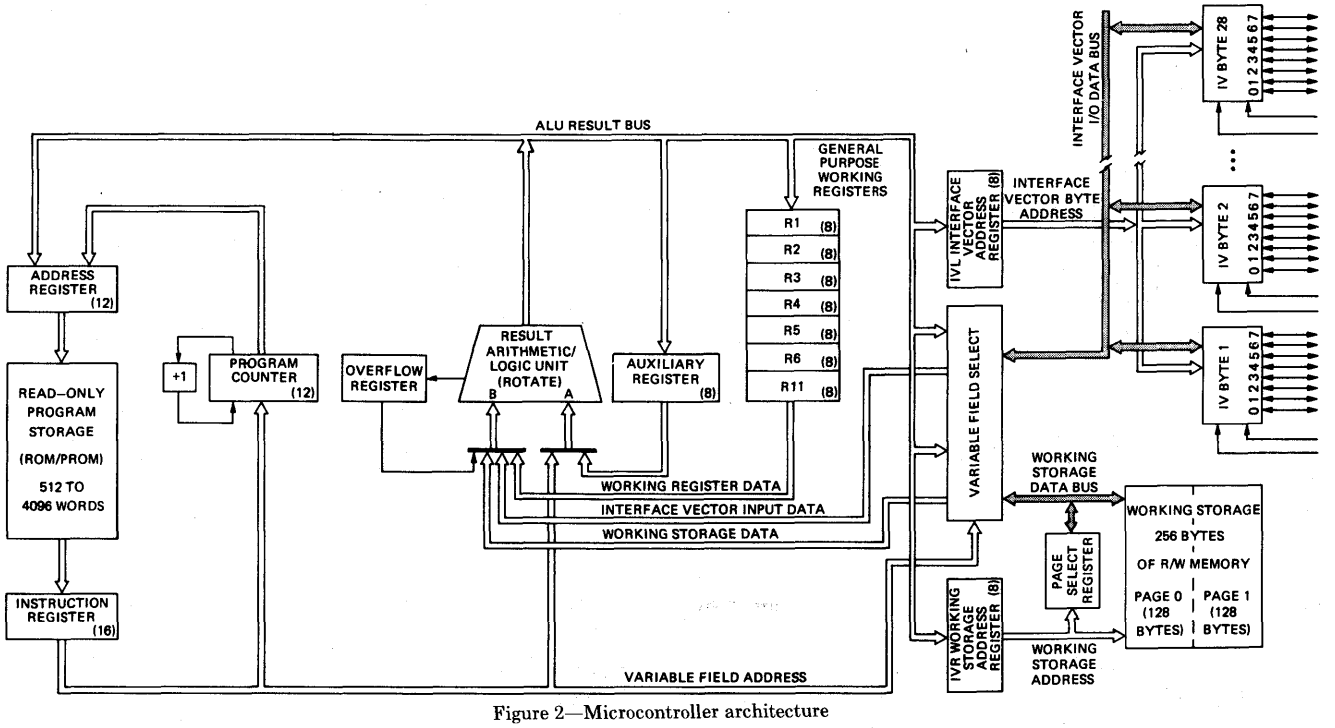


Figure 2—Microcontroller architecture

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 MicroController instruction format is:

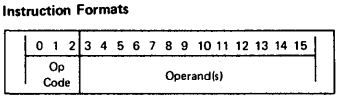


Table I contains a summary of the MicroController instruction set and description of the operand fields.

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 (R) Field, Length (L) Field, Immediate (I) Operand Field, and (Program Storage) Address (A) Field.

The instructions are divided into five format types based on the Op Code and the form of the operand(s).

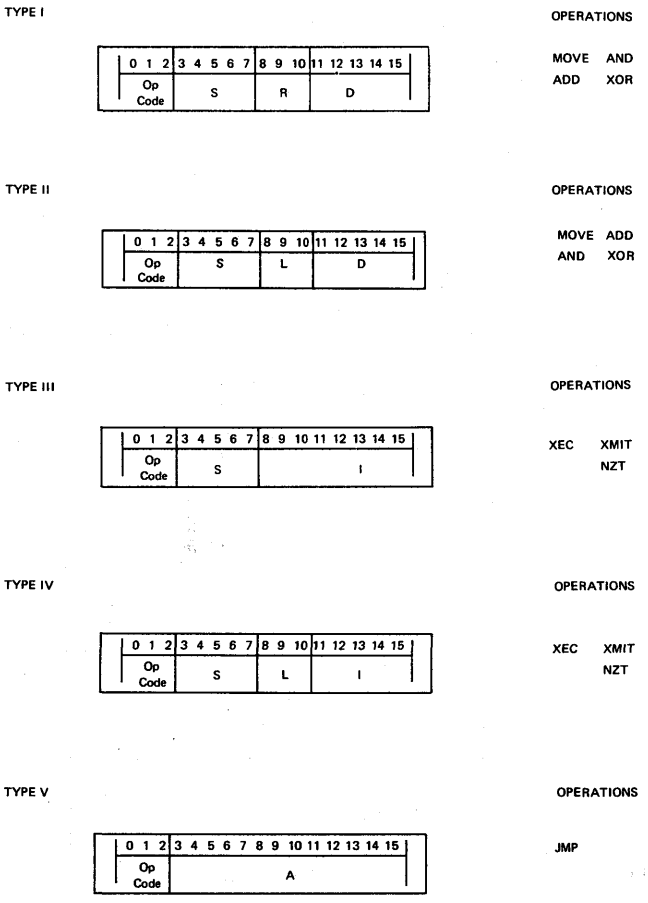


TABLE I—MicroController Instruction Summary

OPERATION	FORMAT	RESULT	NOTES
MOVE	C S L D	Content of data field addressed by S, L replaces data in field specified by D, L.	If S and D both are register addresses then L specifies a right rotate of L places applied to the register specified by S.
ADD		Sum of AUX and data specified by S, L replaces data in field specified by D, L.	
AND		Logical AND of AUX and data specified by S, L replaces data in field specified by D, L.	
XOR		Logical exclusive OR of AUX and data specified by S, L replaces data in field specified by D, L.	
XMIT	C S L I	The literal value I replaces the data in the field specified by S, L.	If S is IV or WS address then I limited to range 00-37. Otherwise I limited to range 000-377.
NZT	C S I	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 - 37. I is limited to the range 000 - 377 otherwise.
XEC	C S L I	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	C I	The literal value I replaces contents of the Program Counter.	I limited to the range 00000 - 07777.

Instruction fields

Op code field—3-bit field

The Op Code field is used to specify one of eight MicroController instructions.

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

S,D fields—5-bit fields

The S and D fields specify the source and destination of 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 three variables are referenced in 300 nanoseconds.

0_8-17_8 is used to specify one of seven working registers (R1-R6, R11), Auxiliary Register, Overflow Register, IVL and IVR registers.

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

20_8-27_8 is used to specify the least significant bit of a variable length field within the IV Byte selected by the address in the IVL register. The length of the field is determined by L.

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

30_8-37_8 is used to specify the least significant bit of a variable length field within the Working Storage Byte selected by the address in IVR Register. The length of the field is determined by L.

OCTAL VALUE	
30	Field within selected W.S. Byte; position of LSB=0
31	=1
32	=2
33	=3
34	=4
35	=5
36	=6
37	=7

L/R field—3-bit field

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

A. When both S and D specify registers, the R 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.)

B. When either or both the S and D fields specify and IV or Working Storage Byte, the L field is used to specify the length of the field (within the byte) accessed, as shown below:

OCTAL VALUE
0—Field length=8 bits
1—Field length=1 bit
2—Field length=2 bits
3—Field length=3 bits
4—Field length=4 bits
5—Field length=5 bits
6—Field length=6 bits
7—Field length=7 bits

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 Byte or to specify the low order bits of the Program Counter.

The length of the I field is based on S field:

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 Byte, 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. In current systems, however, only 12 bits are used, resulting in storage capacity of 4096 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 operation (MOVE, ADD, AND, XOR) result are stored. The operation is performed on the entire 8-bit source for a MOVE, or between the 8-bit AUX and the source register for ADD, AND, XOR operations. The least significant bits of the result are stored in the IV or Working Storage field specified in the instruction.

When an IV or Working Storage field of one to eight bits is specified as the source, and a register as the destination, the 8-bit result of the operation (MOVE, ADD, AND, XOR) is stored in the register. The operations ADD, AND, XOR actually use the IV or Working Storage data field (1-8 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 registers can be specified as destination fields only, (see description of S, D fields), 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.

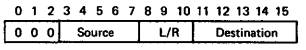
The OVF (Overflow) Register only can be used as a source field; therefore, it cannot be set or reset in a single instruction.

Instruction descriptions

The following instruction descriptions employ MCMAC (the MicroController Machine Compiler, described in a later section) programming notation. This notation varies somewhat from the instruction descriptions provided earlier. Thus, for example, explicit L field definition as shown is not required by MCMAC for machine instructions; MCMAC 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 MicroController instruction set is described below with examples illustrating instruction use.

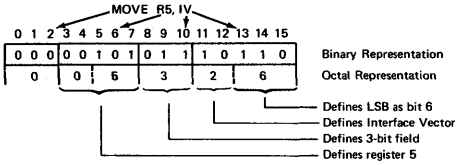
MOVE S,D or
MOVE S (R), D

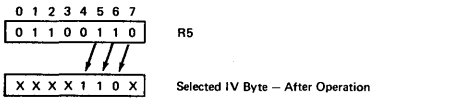


OPERATION: (S) → (D)

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 specifies a right rotate of the source data during the move. Otherwise, L is implicit and specifies the length of the source and destination fields. If the MOVE is between an IV Byte and a Working Storage Byte, an 8-bit field is always moved.

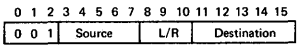
EXAMPLE: Store the least significant 3 bits of register 5 (R5) in bits 4, 5 and 6 of the IV Byte addressed by IVL register.





Note: X's in the IV Byte denote bits unaffected by the MOVE operation.

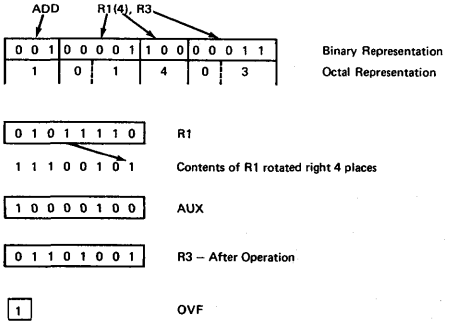
ADD S, D or
ADD S (R), D



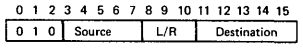
OPERATION: (S) plus (AUX) → D; set OVF if carry from most significant bit occurs.

DESCRIPTION: Unsigned two'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 set. If both S and D are registers, R specifies a right rotate of the source (S) field before the operation. Otherwise L is implicit and specifies the length of the source and destination 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.



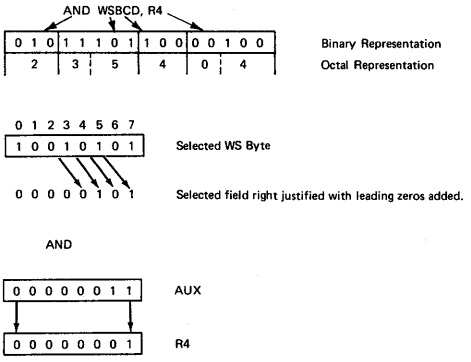
AND S, D or
AND S (R), D



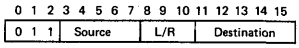
OPERATION: (S) ∧ (AUX) → D

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 specifies a right rotate of the source (S) field before the AND operation. Otherwise L is implicit and specifies the length of the source and destination fields. S and AUX are unaffected unless specified as a destination.

EXAMPLE: Store the AND of the selected Working Storage Byte and AUX in R4. The Working Storage Byte field is called WSBCD and is 4 bits long and located in bits 2, 3, 4 and 5.



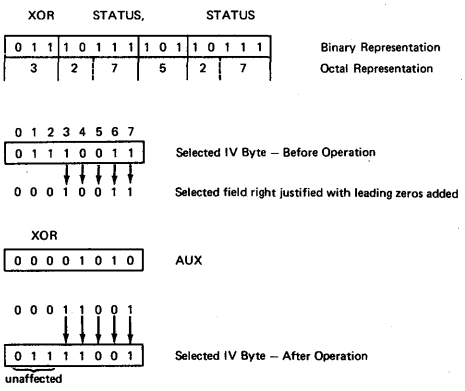
XOR S, D or
XOR S (R), D

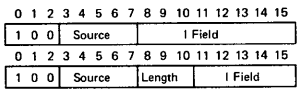


OPERATION: (S) ⊕ (AUX) → D

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 specifies a right rotate of the source (S) field before the XOR operation. Otherwise L is implicit and specifies the length of the source and destination fields. S and AUX are unaffected unless specified as a destination.

EXAMPLE: Replace the selected IV Byte field with the XOR of that 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.



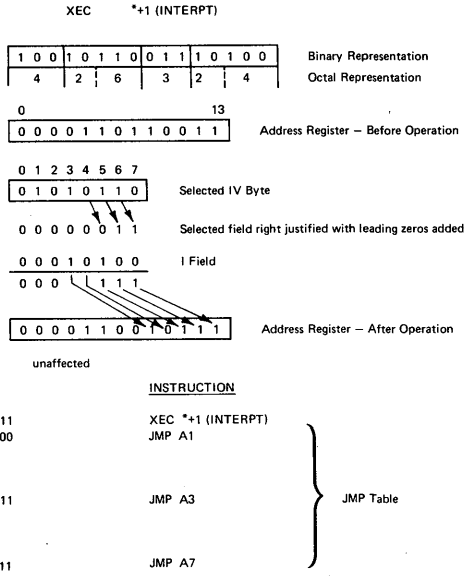


XEC I(S)

OPERATION: Execute instruction at (Address Register) offset by (S) + I.

DESCRIPTION: Execute the instruction at the address determined by replacing the low order bits of the Address Register (AR) (which contains the current value of the Program Counter) 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 Byte, 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).

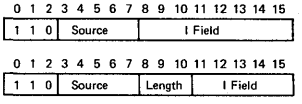
EXAMPLE: Execute a JMP in a table of JMP instructions determined by the value of the selected IV Byte field. The table follows immediately after the XEC instruction and the IV field is called INTERPT and is a 3 bit field located in bits 4, 5 and 6.



ADDRESS	INSTRUCTION	
0000110110011	XEC *+1 (INTERPT)	
0000110110100	JMP A1	JMP Table
•		
•		
•		
0000110110111	JMP A3	
•		
•		
•		
0000110111011	JMP A7	

JMP A3 is executed because IV field INTERPT = 3

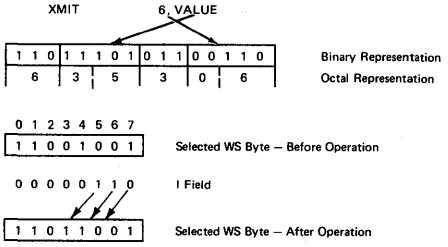
XMIT I, S



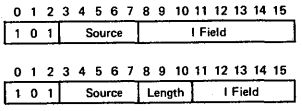
OPERATION: I → S

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

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



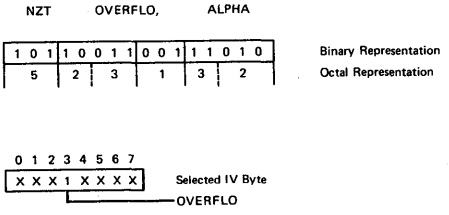
NZT S, I



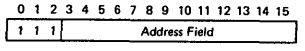
OPERATION: Non Zero Transfer If (S) ≠ 0, PC offset by I → PC; otherwise PC + 1 → PC.

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 are replaced; if S is an IV or Working Storage Byte, the low order 5 bits 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.



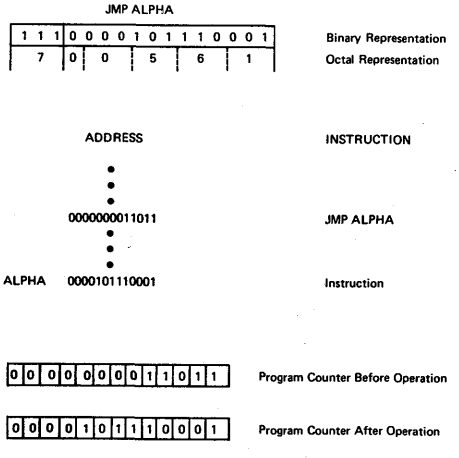
JMP A



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 - 7777_8 in current systems (0 - 4095).

EXAMPLE: Jump to location ALPHA (0000101110001)



INPUT/OUTPUT SYSTEM

As seen from previous sections, the Interface Vector is the MicroController's input/output system. It provides a simple interconnection to the user status, control and data lines.

Addressing data on the interface vector

The Interface Vector is comprised of general purpose I/O registers called Interface Vector (IV) Bytes. In the present MicroController offering, the Interface Vector may consist of up to 28 IV Bytes.

As seen from Figure 2 the IVL register serves as the address register to the IV Bytes. In order for an instruction access (read or write) an IV Byte, the address of that byte must first be placed into the IVL register.

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

XMIT ADDRESS, IVL
MACHINE INSTRUCTION

Once the IV Byte is selected (addressed) it will remain selected until the IVL register is loaded with another address. 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.

Although the address range of IVL is $0\text{-}377_8$, only 28 IV Bytes are available on current system offerings. The addressing for the 28 IV Bytes is 01_8 to 34_8.

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 two input/output control lines, called Byte Input Control (BIC) and Byte Output Control (BOC) (Figure 3). The control lines functions are summarized in Table II.

READ/WRITE MEMORY

In MicroController applications, data may be stored in a read/write memory system called Working Storage.

TABLE II—Functions of the BIC and BOC Lines

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 III contains a summary of the electrical characteristics of the IV Byte.

TABLE III—IV BYTE Terminal Electrical Characteristics

Characteristic	Symbol	Limits Min	Typ	Max	Units	Conditions
"1" Input Current*	I_{1in}			100	uA	V_{1in}=5.5 V
"0" Input Current*	$I_{\phi in}$			−800	uA	$V_{\phi in}$=0.50 V
"1" Input Voltage	V_{1in}	2		5.5	Volts	
"0" Input Voltage	$V_{\phi in}$	−1		0.8	Volts	
Input Clamp Voltage	V_{cin}			−1	Volts	$I_{\phi in}$ = −5ma
High Output Voltage	V_{1out}	2.4			Volts	I_{1out}=1ma
Low Output Voltage	$V_{\phi out}$			0.5	Volts	$I_{\phi out}$ = −16ma
Output Short Circuit Current	I_{so}	−20		−200	ma	$V_{\phi out}$=OV
Data Input Capacitance	C_{in}			12	pf	$V_{\phi in}$=OV

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

Working Storage is accessed in much the same manner as IV Bytes. Figure 2 shows that IVR register is the Working Storage address register. It should also be noted from Figure 2 that a Page Select Register determines the page currently addressed by the IVR register. In order to access the Page Select Register, IVR must be set to 177_8, which is the address of the Page Select Register. Either a 1 or 0 can be transferred into bit 7 to select page 1 and page 0 respectively. Once the proper page is selected, IVR can be loaded with the address of the Working Storage Byte requiring access.

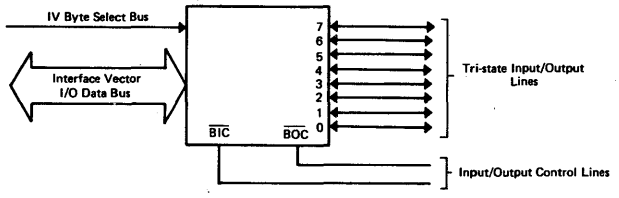


Figure 3.1
IV BYTE PROVIDING DYNAMICALLY DEFINED DATA FLOW

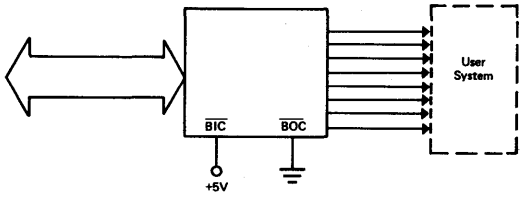


Figure 3.2
IV BYTE WIRED FOR USER OUTPUT ONLY

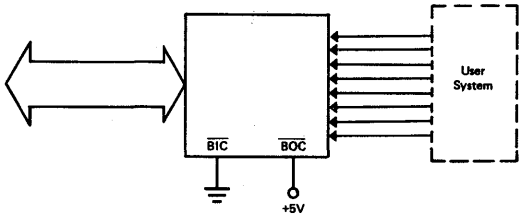


Figure 3—IV byte wired for input only

Because the two 128 byte pages of Working Storage are selected by the Page Select Register, the address loaded into IVR to access a byte in either page is identical 200_8-377_8. In effect, IVR holds the low order address bits and Page Select holds the high order address bit.

Operating on data in Working Storage requires two steps:

a. Selecting the 128 byte page which contains the data.
b. Accessing and operating on the actual data byte(s).

Page selection requires two instructions:

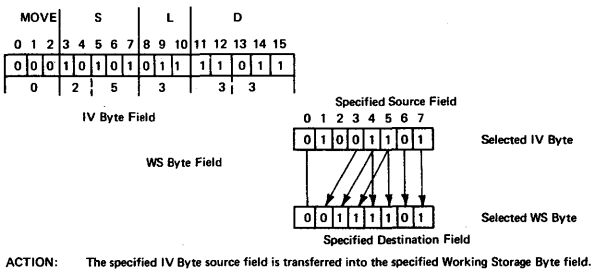
XMIT	177H, IVR	Enables Page Select Register
XMIT	PAGE, PSR	Selects Page

Thereafter all references to bytes within the selected page require two instructions:

XMIT ADDRESS, IVR Selects byte
MACHINE INSTRUCTION

When using instructions that involve the transfer of fields of less than 8 bits between an IV Byte and Working Storage, the following results should be noted.

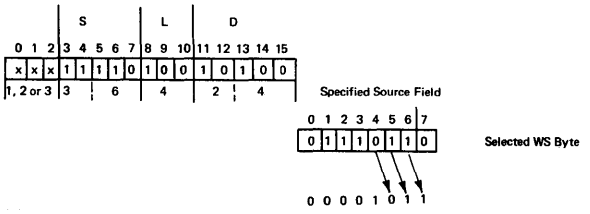
EXAMPLE: A MOVE instruction that specifies an IV Byte and Working Storage will have the following result:



ACTION: The specified IV Byte source field is transferred into the specified Working Storage Byte field. The remainder of the destination byte is filled by the contents of the corresponding bit positions in the source bytes.

EXAMPLE: An ADD, XOR or AND instruction that specifies an IV Byte and Working Storage will have the following result:

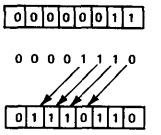
ADD
AND
XOR



Specified Source Field

Selected WS Byte

ADD



Auxiliary Register

Specified Destination Field
Corresponding bits of Source Field

ACTION: The specified source field (right justified with leading zero's inserted) is added/anded/exclusive or'ed with the Auxiliary Register and the result placed in the destination field. The remainder of the destination byte is filled with the corresponding bit positions of the source byte.