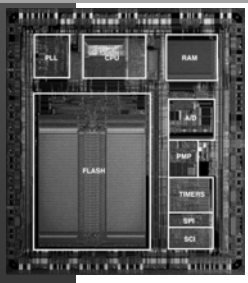
CPU08

The CPU08 Central Processing Unit

M68HC08 and HCS08 microcontrollers

Babak Kia
Adjunct Professor
Boston University
College of Engineering
Email: bkia -at- bu.edu

Introduction



- **Central Processing Unit is one of the components that makes up a computer.**
  - **CPU**
  - **Memory Subsystem**
  - **Bus structure**
- **CPU is made up of Control and Execution units, and handles**
  - **Arithmetic & logical operations**
  - **Code execution & Branching**
  - **Data transfer to & from memory**
  - **Interrupt management**

Architecture of a Microcontroller

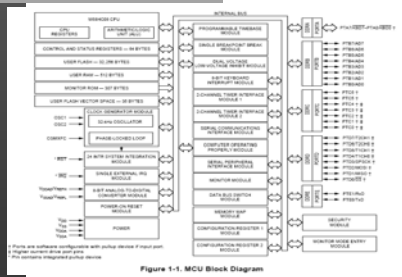


Figure 1-1. MCU Block Diagram

Topics of Discussion

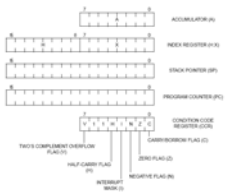
- **Features of CPU08**
- **Architecture Overview**
- **Addressing Modes**
- **A note on assemblers**
- **CPU08 Instruction Summary**

Features of CPU08

- **CPU08 is an 8-bit architecture**
- **8-MHz CPU standard bus frequency**
- **64-Kbyte memory space**
- **Fully object-code compatible with the M68HC05**
- **Instructions designed around stack manipulation**
- **16 addressing modes**
- **Instructions capable of moving data from memory-to-memory without using the accumulator**
- **Extensible addressing range beyond the 64K boundary**

Architecture Overview

- **The CPU08 is an 8-bit architecture utilizing the following registers:**
  - **A - Accumulator**
  - **CCR - Condition Code Register**
  - **H - High byte of Index Register**
  - **X - Low byte of Index Register**
  - **PC - Program Counter**
  - **PCH - High byte of PC**
  - **PCL - Low byte of PC**
  - **SP - Stack Pointer**



CPU08 Registers

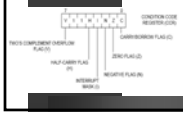
- **Accumulator (8 bit)**
  - **A general purpose 8-bit register used by the CPU to hold operands, or results of arithmetic or logical operations**
- **Index Register (16 bit)**
  - **Can be accessed either as a 16 bit (H:X) register, or independently as two 8 bit H, or X registers**
  - **Used by the CPU to index into, or address a 64KB memory range**

CPU08 Registers

- **Stack Pointer (16 bit)**
  - **The Stack Pointer always points to the next available location of the stack**
  - **Address decrements as data is moved (pushed to) the stack, and increments as data is moved out (pulled) from the stack**
- **Program Counter (16 bit)**
  - **Holds the address of the next instruction to be fetched**
  - **Automatically incremented, unless a jump or a branch takes place**
  - **Is loaded with the contents of the Reset Vector at time of reset**

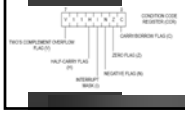
Condition Code Register

- **The Condition Code Register is an 8-bit register which is comprised of the Interrupt Mask, and 5 flags which contain the results of the last executed command**
- **V – Overflow Flag**
  - **Set when two's complement overflow occurs**
  - **Set by ASL, ASR, LSL, LSR, ROL, ROR**
  - **Used by BGT, BGE, BLE, BLT**



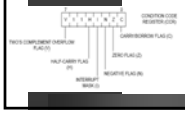
Condition Code Register

- H – Half-Carry Flag
  - Set when a carry occurs between bits 3 and 4 of the accumulator
  - The DAA uses the value of H to make appropriate adjustments (BCD)
- I – Interrupt Mask
  - When set, all interrupts are disabled
  - When an interrupt occurs, Interrupt Mask is automatically set
  - Further interrupts are latched until an RTI is issued



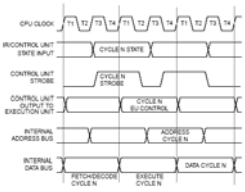
Condition Code Register

- N – Negative Flag
  - Set whenever an arithmetic or logical operation produces a negative result
- Z – Zero Flag
  - Set whenever an arithmetic or logical operation produces a zero result
- C – Carry or Borrow Flag
  - Set when an arithmetic operation produces a carry flag, or a subtraction operation requires a borrow
  - Some other operations such as bit test can also set or clear this flag

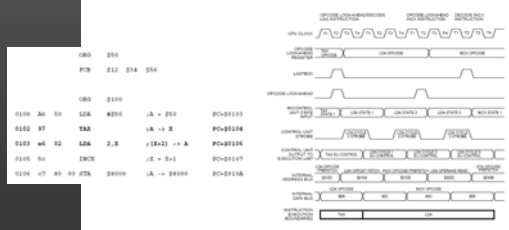


Control Unit

- The control unit is comprised of the Sequencer, the Control Store, and control logic
- It is primarily reponsible for "preparing" the instruction for execution (i.e. fetching, decoding, address resolution)
- Sequencer provides the next state of the machine to the control store based on the contents of the Instruction Register (IR)
- Control store provides the next state condition to the Execution Unit (EU)



Instruction Execution



Addressing modes

- Inherent
- Immediate
- Direct
- Extended
- Indexed
  - No offset, post increment
  - 8-bit offset, post increment
  - 16-bit offset, post increment
- Stack Pointer
  - 8-bit offset
  - 16-bit offset
- Relative
- Memory to Memory (4 modes)

Inherent Addressing

- Have no operand associated with them (addresses are inherent)
- They require no memory address
  - INCA – Increment Accumulator
  - CLRA – Clear Accumulator
  - PSHH – Push Index High onto Stack
- Example

```
LDA $55 ; A = $55
INCA    ; A = $56
```

Immediate Addressing

- Operand immediately follows the opcode
- Operand sizes are 1 or 2 bytes long
  - ADD – Add immediate value to Accumulator
  - AIX – Add immediate to Index register (H:X)
- Example

```
CLRH        ; H = 0
CLRX        ; X = 0
AIX #$1FF   ; H = $01
            ; X = $FF (H:X = $1FF)
```

Direct Addressing

- Direct Addressing is limited to operands in the $0000 - $00FF area of memory (page 0)
- Operands are the low byte of the memory address, the high byte is assumed to be $00
- Fast execution – one clock cycle
- Important since most RAM is originated at $0000
- Examples are
  - Arithmetic Shifts (ASL, ASR, etc.)
  - Loads and stores (LDA, STA, etc.)

Extended Addressing

- Used to access any address in a 64K boundary
- Any address above direct or zero page requires this instruction
- Example

```
        org $E000
        db $55
        …
Start:  LDA $E000    ; A = $55
```

Indexed Addressing

- Have 1, 2, or 3 operands for no offset, 8-bit, or 16-bit offset
- Indexed addressing is of great value as Instructions can seamlessly move a pointer through a table
- Incrementing the pointer is done automatically through post-increment
- Using 8 or 16-bit offset, we can easily access any element in an array

Indexed Addressing Example

- The following example shows the three different modes of index addressing

```
LDA ,x          ; no offset
                ; load value pointed to by HX
LDA $FF,x ; 8-bit offset
                ; load value pointed to by
                ; HX + $FF
LDA $1000,x     ; 16-bit offset
                ; load value pointed to by
                ; HX + $1000
```

Stack Pointer Addressing

- Indexing off of the Stack Pointer is done much the same way as Indexed Addressing
- If interrupts are off, this mode allows the Stack Pointer to be used as a second index register

Relative Addressing

- Relative addressing is done relative to the current value of the Program Counter (PC)
- The CPU automatically adds a signed offset (-128 to 127) to the PC
- The offset gives the relative address starting from the address location *following* the current branch instruction
- Only conditional branch instructions use this addressing mode

Memory to Memory Addressing

- Memory to Memory is expensive
  - Is a three-byte, four cycle operation
- But is less expensive than having to save the contents of the accumulator
- Moving Memory with Accumulator takes 9 cycles:

```
PSHA      ; (2 cy) save contents of A
LDA #$55  ; (2 cy) load A with data
STA $10   ; (3 cy) save A into memory
PULA      ; (2 cy) restore A
```

- Moving Memory with MOV takes 4 cycles:

```
MOV #$55, $10    ; (4 cy)
```

Memory to Memory cont.

- Moving memory using Direct to Direct mode also saves time
  - 10 cycles for doing so with accumulator
  - 5 cycles for doing so using MOV
- This is a valuable addressing mode because the contents of the accumulator are not changed
- Savings can be substantial with a lot of data movement

A Note on Assemblers

- The programmer generally does not need to keep in mind which mode of Index, Relative, or SP addressing mode to use
- For instance, the assembler can automatically select no offset, 8-bit, or 16-bit Index addressing mode
- The assembler can also calculate offsets for Relative Addressing mode, and verify that they are within range

Resets and Interrupts

- CPU is designed in a way as to execute instructions sequentially
- However, external events such as interrupts and resets are asynchronous to program execution and require special handling by the CPU
- Reset is the mechanism by which we force (initialize) the CPU into a known state. This includes loading the Program Counter (PC) from a pre-defined non-volatile memory location to start execution from a known state
- Interrupts temporarily suspend normal program execution and cause the CPU to switch context to service the interrupt
- The CPU08 has up to 128 different interrupt sources

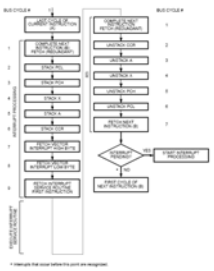
Resets and Interrupts

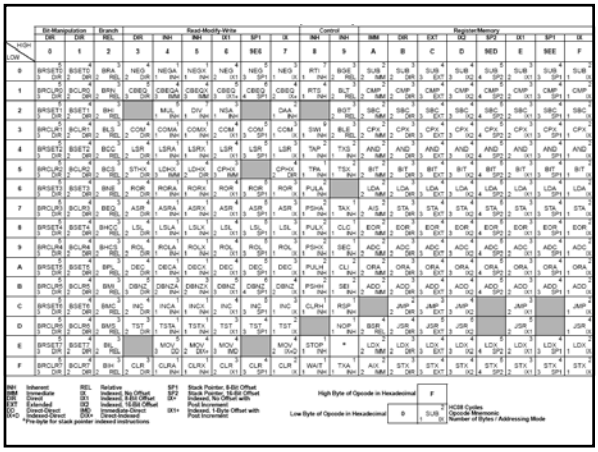
- Interrupts come in two flavors, maskable and non-maskable. The only non-maskable "interrupt" on the CPU08 is reset
- Interrupts on the CPU08 are prioritized according to a hardware defined priority scheme
- Once reset occurs, the 16-bit reset vector is fetched from $FFFE, and the stack pointer is set to $00FF
- Some reset sources are the RESET# pin, COP watchdog timer, Illegal opcode, and Low Voltage Inhibit (LVI)

Address	Reset	Priority
FFFE	Reset	1
FFFC	SWI	2
FFFA	IRQ[0]	3
FF02	IRQ[124]	127
FF00	IRQ[125]	128

Interrupt Handling

- **Interrupt Processing is comprised of 5 steps:**
  - **Recognition – Interrupt source must be unmask**
  - **Arbitration – Interrupt priority is evaluated, arbitrate between pending INTs, etc.**
  - **Stacking – Saving the current state of the processor onto the stack**
  - **Execution – Fetching the interrupt vector & handling it**
  - **Returning – Restoring the state of the processor**





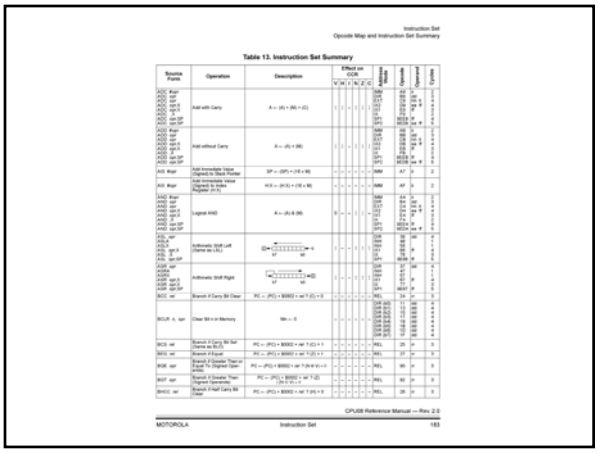


Table 13. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Cycles
BHCS rel	Branch if Half Carry Bit Set	PC ← (PC) + $0002 + rel ? (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI rel	Branch if Higher	PC ← (PC) + $0002 + rel ? (C) | (Z) = 0	–	–	–	–	–	–	REL	22	rr	3
BHS rel	Branch if Higher or Same (Same as BCC)	PC ← (PC) + $0002 + rel ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH rel	Branch if IRQ Pin High	PC ← (PC) + $0002 + rel ? IRQ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL rel	Branch if IRQ Pin Low	PC ← (PC) + $0002 + rel ? IRQ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #opr BIT opr BIT ,opr BIT opr,S BIT ,S BIT ,opr,SP BIT ,opr,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9E15 9E16	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 5 4 3 5 4
BLE opr	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + $0002 + rel ? (Z) | (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO rel	Branch if Lower (Same as BCS)	PC ← (PC) + $0002 + rel ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS rel	Branch if Lower or Same	PC ← (PC) + $0002 + rel ? (C) | (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT opr	Branch if Less Than (Signed Operands)	PC ← (PC) + $0002 + rel ? (N ⊕ V) = 1	–	–	–	–	–	–	REL	91	rr	3
BMC rel	Branch if Interrupt Mask Clear	PC ← (PC) + $0002 + rel ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI rel	Branch if Minus	PC ← (PC) + $0002 + rel ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS rel	Branch if Interrupt Mask Set	PC ← (PC) + $0002 + rel ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE rel	Branch if Not Equal	PC ← (PC) + $0002 + rel ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL rel	Branch if Plus	PC ← (PC) + $0002 + rel ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA rel	Branch Always	PC ← (PC) + $0002 + rel	–	–	–	–	–	–	REL	20	rr	3
BRCLR n,opr,rel	Branch if Bit n in Memory Clear	PC ← (PC) + $0003 + rel ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRSET n,opr,rel	Branch if Bit n in Memory Set	PC ← (PC) + $0003 + rel ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET n, opr	Set Bit n in Memory	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4

Table 13. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Cycles
BSR rel	Branch to Subroutine	PC ← (PC) + $0002 ; push (PCL) SP ← (SP) − $0001 ; push (PCH) SP ← (SP) − $0001 PC ← (PC) + rel	–	–	–	–	–	–	REL	AD	rr	4
CBEQ #opr,rel CBEQ opr,rel CBEQX #opr,rel CBEQ ,opr,rel CBEQ opr,S,rel CBEQ ,opr,SP,rel	Compare and Branch if Equal	PC ← (PC) + $0003 + rel ? (A) − (M) = $00 PC ← (PC) + $0003 + rel ? (A) − (M) = $00 PC ← (PC) + $0003 + rel ? (A) − (M) = $00 PC ← (PC) + $0003 + rel ? (A) − (M) = $00 PC ← (PC) + $0004 + rel ? (A) − (M) = $00 PC ← (PC) + $0004 + rel ? (A) − (M) = $00	–	–	–	–	–	–	IMM DIR IMM IX1 IX SP1	41 51 61 71 31 9E61	ii rr dd rr ii rr ff rr rr rr ff rr	4 5 4 5 5 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask Bit	I ← 0	–	–	0	–	–	–	INH	9A		2
CLR opr CLRA CLRX CLRH CLR ,opr,X CLR ,X CLR ,opr,SP	Clear	M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 7F 6F 9E6F	dd ff ff	5 1 1 1 4 3 5
CMP #opr CMP opr CMP ,opr CMP opr,S CMP ,X CMP ,opr,SP CMP ,opr,SP	Compare Accumulator with Memory	(A) − (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9E61 9ED1	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 5 4 3 5 4
COM opr COMA COMX COM ,opr,X COM ,X COM ,opr,SP	Complement (One's Complement)	M ← ($FF − (M)) A ← ($FF − (A)) X ← ($FF − (X)) M ← ($FF − (M)) M ← ($FF − (M)) M ← ($FF − (M))	0	–	–	↕	↕	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	5 1 1 4 3 5
CPHX #opr CPHX opr	Compare Index Register (H:X) with Memory	(H:X) − (M:M + $0001)	↕	–	–	↕	↕	↕	IMM DIR	65 75	ii ii dd	3 4
CPX #opr CPX opr CPX ,opr CPX opr,S CPX ,X CPX ,opr,SP CPX ,opr,SP	Compare X (Index Register Low) with Memory	(X) − (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9E63 9ED3	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 5 4 3 5 4
DAA	Decimal Adjust Accumulator	(PIA)	U	–	–	↕	↕	↕	INH	72		2
DBNZ opr,rel DBNZA rel DBNZX rel DBNZ ,X,rel DBNZ ,opr,SP,rel	Decrement and Branch if Not Zero	A ← (A)−$01 or M ← (M)−$01 or X ← (X)−$01 PC ← (PC) + $0003 + rel ? (result) ≠ 0 for DBNZ opr and DBNZ ,X rel PC ← (PC) + $0002 + rel ? (result) ≠ 0 for DBNZA and DBNZX PC ← (PC) + $0004 + rel ? (result) ≠ 0 for DBNZ ,opr,SP	–	–	–	–	–	–	DIR INH INH IX1 SP1	3B 4B 5B 6B 9E6B	dd rr ff ff rr	5 3 3 5 6
DEC opr DECA DECX DEC ,X DEC opr,S DEC ,opr,SP	Decrement	M ← (M) − $01 A ← (A) − $01 X ← (X) − $01 M ← (M) − $01 M ← (M) − $01 M ← (M) − $01	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	5 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	↕	↕	↕	INH	52		7

Table 13. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Cycles
EOR #opr EOR opr EOR ,opr EOR opr,S EOR ,X EOR ,opr,SP EOR ,opr,SP	Exclusive OR Memory with Accumulator	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9E68 9ED8	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 5 4 3 5 4
INC opr INCA INCX INC ,opr,X INC ,X INC ,opr,SP	Increment	M ← (M) + $01 A ← (A) + $01 X ← (X) + $01 M ← (M) + $01 M ← (M) + $01 M ← (M) + $01	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	5 1 1 4 3 5
JMP opr JMP opr JMP ,opr JMP opr,S JMP ,X JMP ,opr,SP	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX SP2	BC CC DC EC FC 9EDC	dd hh ll ee ff ff ff	3 4 4 3 2 4
JSR opr JSR opr JSR ,opr JSR opr,S JSR ,X JSR ,opr,SP	Jump to Subroutine	PC ← (PC) + n (n = 1, 2, or 3) Push (PCL); SP ← (SP) − $0001 Push (PCH); SP ← (SP) − $0001 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX SP2	BD CD DD ED FD 9EDD	dd hh ll ee ff ff ff	5 6 6 5 4 6
LDA #opr LDA opr LDA ,opr LDA opr,S LDA ,X LDA ,opr,SP LDA ,opr,SP	Load Accumulator from Memory	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9E66 9ED6	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 5 4 3 5 4
LDHX #opr LDHX opr	Load Index Register (H:X) from Memory	H:X ← (M:M + $0001)	0	–	–	↕	↕	–	IMM DIR	45 55	jj kk dd	3 4
LDX #opr LDX opr LDX ,opr LDX opr,S LDX ,X LDX ,opr,SP LDX ,opr,SP	Load X (Index Register Low) from Memory	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9E6E 9EDE	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 5 4 3 5 4
LSL opr LSLA LSLX LSL ,opr,X LSL ,X LSL ,opr,SP	Logical Shift Left (Same as ASL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 4 3 5
LSR opr LSRA LSRX LSR ,opr,X LSR ,X LSR ,opr,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	5 1 1 4 3 5
MOV #opr,opr MOV opr,X+ MOV #opr,X+ MOV X+,opr	Move	(M)destination ← (M)source H:X ← (H:X) + $0001 in IX+/DAIX+ Mode	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 5 4 4
MUL	Unsigned multiply	X : A ← (X) × (A)	–	0	–	–	–	0	INH	42		5

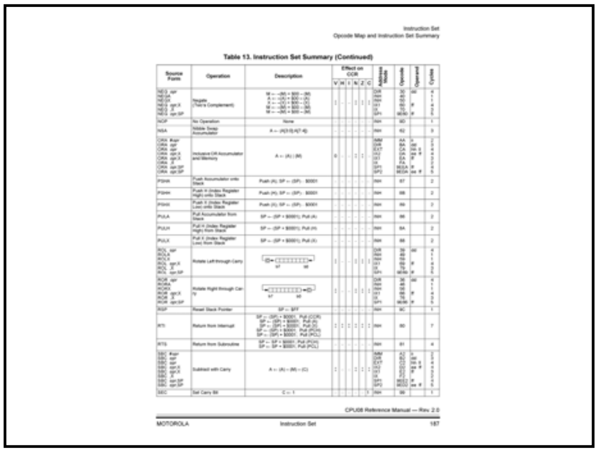
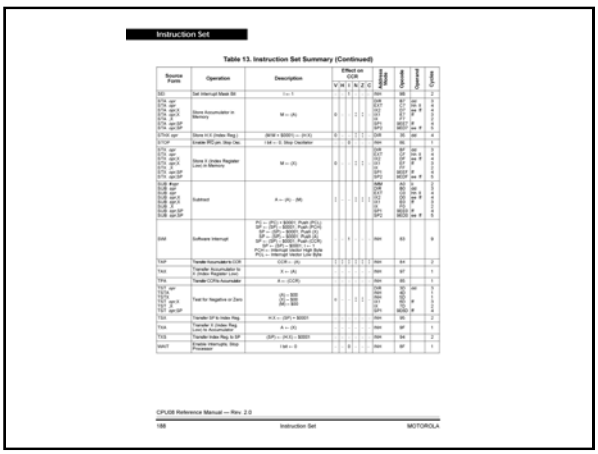
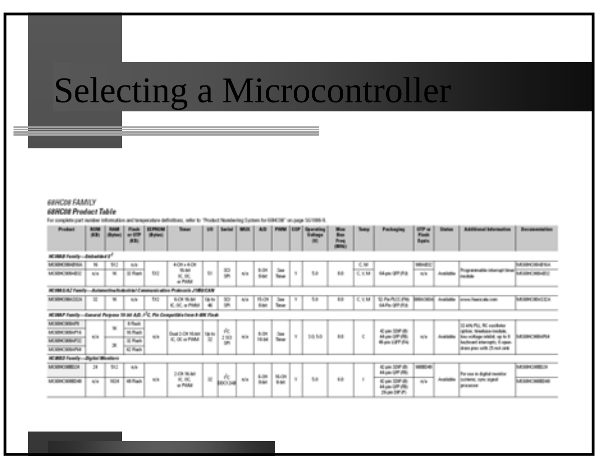
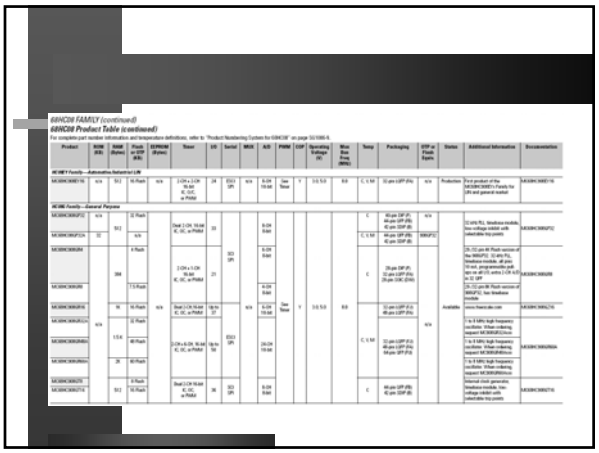


Table 13. Instruction Set Summary (Continued)


Instruction Set
Table 13. Instruction Set Summary (Continued)


Selecting a Microcontroller

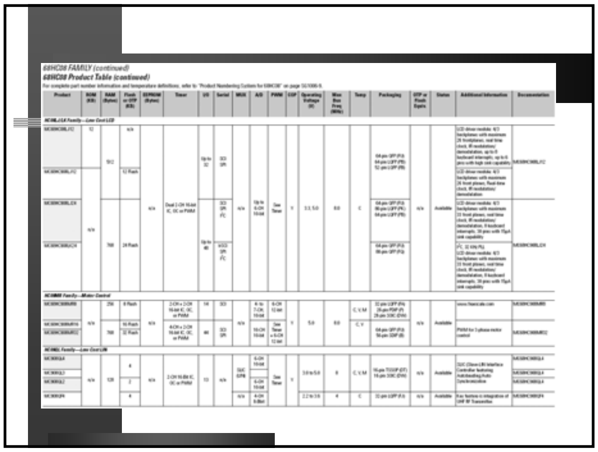


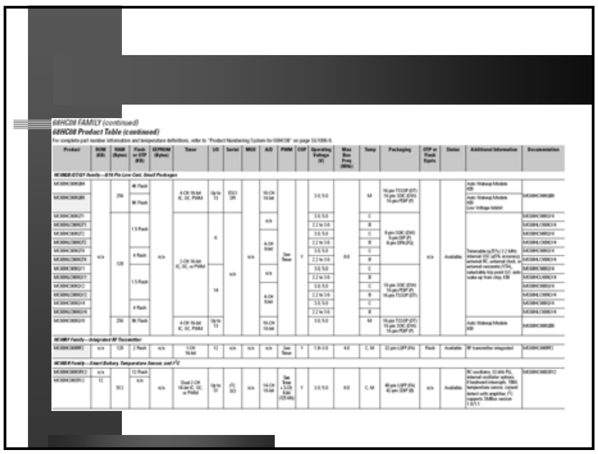
68HC08 FAMILY (continued)

68HC08 Product Table (continued)

For complete part number information and temperature definitions, refer to "Product Numbering System for 68HC08" on page 5U1000-5.

Product	ROM (KB)	RAM (Bytes)	Flash or OTP (KB)	EEPROM (Bytes)	Timer	I/O	Serial	MOC	A/D	PWM	COP	Operating Voltage (V)	Max Bus Freq (MHz)	Temp	Packaging	OTP or Flash Equiv.	Status	Additional Information	Documentation
HC16F1 Family—Automotive, Industrial LIN																			
MC68HC908EY16	n/a	512	16 Flash	n/a	2-CH x 2-CH 16-bit IC, OC, or PWM	24	ESCI SPI	n/a	8-CH 10-bit	See Timer	Y	3.0, 5.0	8.0	C, I, M	32-pin LQFP (FA)	n/a	Production	First product of the MC68HC908EY16 family for LIN and general market	MC68HC908EY16
HC08J Family—General Purpose																			
MC68HC908JW32	n/a	512	32 Flash		Dual 2-CH 16-bit IC, OC, or PWM	33			8-CH 8-bit					C	40-pin DIP (P), 44-pin LQFP (FB), 42-pin SDIP (B)	n/a		32 kHz PLL, Wireless module, low-voltage inhibit with selectable trip points	MC68HC908JW32
MC68HC908JW32A	32		n/a											C, I, M	44-pin LQFP (FB), 42-pin SDIP (B)	908JW32			
MC68HC908JB8	384		8 Flash		2-CH x 1-CH 16-bit IC, OC, or PWM	21	SCI SPI		8-CH 8-bit					C	28-pin DIP (P), 32-pin LQFP (FA), 28-pin SOIC (DW)	n/a	Available	25–32-pin 4K Flash version of the 908JP32; 32 kHz PLL, Wireless module, all pins have programmable pull-up; 3 bits are all 16-bit timer, 2-CH x 16-bit SPI	MC68HC908JB8
MC68HC908JB8			7.5 Flash						8-CH 8-bit									25–32-pin 8K Flash version of the 908JP32; 32 kHz PLL, Wireless module	
MC68HC908JP16	n/a	1K	16 Flash	n/a	Dual 2-CH 16-bit IC, OC, or PWM	Up to 37	n/a	n/a	8-CH 10-bit	See Timer	Y	3.0, 5.0	8.0		32-pin LQFP (FA), 48-pin LQFP (FA)	n/a		www.freescale.com	MC68HC908JL16
MC68HC908JL16		1.5 K	32 Flash															1- to 8-MHz high-frequency oscillator. When ordering, request MC908JB16xxx.	
MC68HC908JB8A			40 Flash		2-CH x 6-CH, 16-bit IC, OC, or PWM	Up to 50	ESCI SPI		10-CH 10-bit					C, I, M	32-pin LQFP (FA), 48-pin LQFP (FA), 64-pin QFP (FU)			1- to 8-MHz high-frequency oscillator. When ordering, request MC908JB8Axxx.	MC68HC908JB8A
MC68HC908JB8A		2K	60 Flash															1- to 8-MHz high-frequency oscillator. When ordering, request MC908JB8Axxx.	
MC68HC908JK3	n/a	0 Flash			Dual 2-CH 16-bit IC, OC, or PWM	36	SCI SPI		8-CH 8-bit					C	48-pin QFP (FB), 42-pin SDIP (B)			Internal clock generator, Wireless module, Low-voltage inhibit with selectable trip points	MC68HC908JK3
MC68HC908JL3		512	16 Flash																





Package & Product Numbering