(MPU) NMOS 8-BIT MICROPROCESSOR UNIT

DESCRIPTION

The EF 6800 is a monolithic 8-bit microprocessor forming the central control function for THOMSON COMPOSANTS MILITAIRES ET SPATIAUX 6800 family. Compatible with TTL, the EF 6800, as with all 6800 system parts, requires only one +5.0 V power supply, and no external TTL devices for bus interface.

The EF 6800 is capable of addressing 64 K bytes of memory with its 16-bits address lines. The 8-bit data bus is bidirectional as well as threestate, making direct memory addressing and multiprocessing applications realizable.

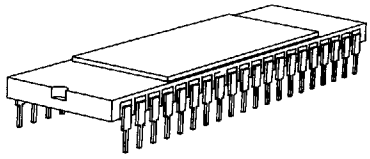
MAIN FEATURES

- 8-bit parallel processing.
- Bidirectional data bus.
- 16-bit address bus - 64 K bytes of addressing.
- 72 instructions - Variable length.
- Seven addressing modes - Direct, relative, immediate, indexed, extended, implied and accumulator.
- Variable length stack.
- Vectored restart.
- Maskable interrupt vector.
- Separate non-maskable interrupt - Internal registers saved in stack.
- Six internal registers - Two accumulators, index register, program counter, stack pointer and condition code register.
- Direct Memory Addressing (DMA) and multiple processor capability.
- Simplified clocking characteristics.
- Clock rates as high as 2.0 MHz.
- Simple bus interface without TTL.
- Halt and single instruction execution capability.
- Three available versions of speed :
  - EF 6800 (1.0 MHz),
  - EF 68A00 (1.5 MHz),
  - EF 68B00 (2.0 MHz at 0-70°C only).
- Temperature range :
  - 0°C to +70°C — for 1, 1.5 and 2 MHz
  - −40°C to +85°C — for 1 MHz and 1.5 MHz only
  - −55°C to +125°C

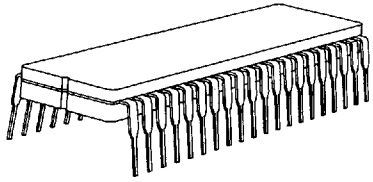
SCREENING / QUALITY

This product is manufactured in full compliance with :

- MIL-STD-883 (class B).
- NFC 96883 (class G).
- TMS standard.



J suffix
DIL 40
Ceramic Side Brazed



C suffix
DIL 40
Cerdip

E suffix
LCCC 44
Leadless Ceramic Chip Carrier

PIN CONNECTION (see chapter 10)

See the ordering information at the end of the data sheet.

5

SUMMARY

A - GENERAL DESCRIPTION

1 - BLOCK DIAGRAM

Figure 1 is a block diagram of the EF 6800.

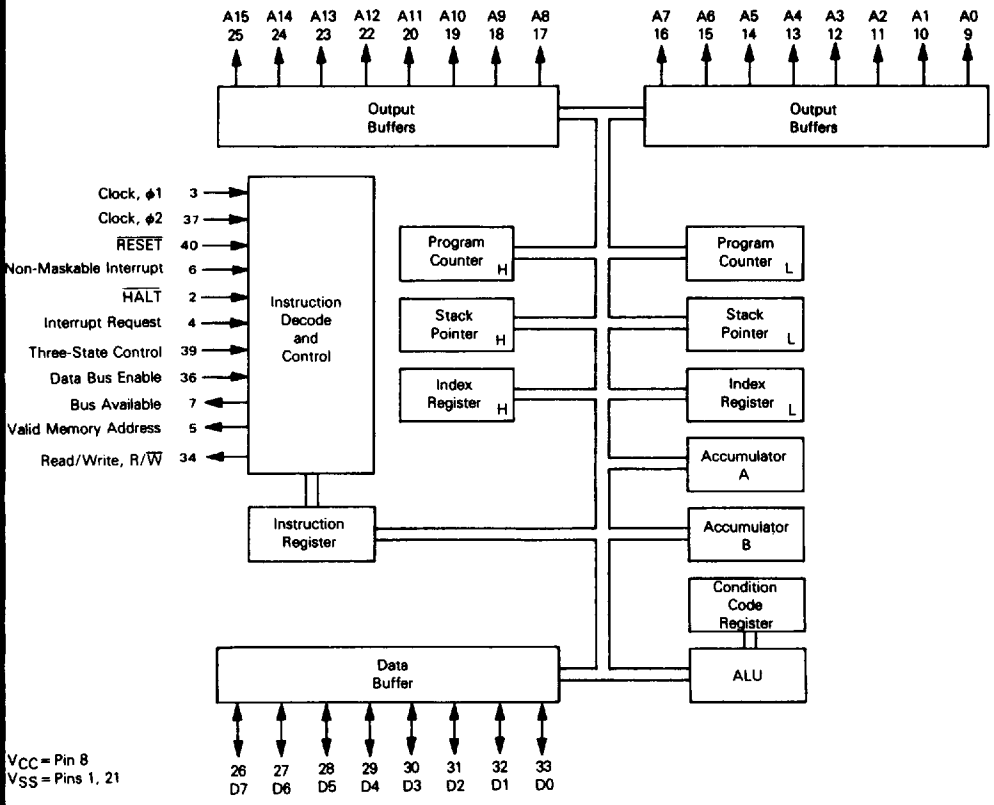


Figure 1 : Expanded block diagram.

B - DETAILED SPECIFICATIONS

1 - SCOPE

This drawing describes the specific requirements for the microprocessor EF 6800 1, 1.5 and 2 MHz, in compliance with MIL-STD-883 class B and TMS standards.

2. APPLICABLE DOCUMENTS

2.1 - MIL-STD-883

1) MIL-STD-883 : test methods and procedures for electronics.
2) MIL-M-38510 : general specifications for microcircuits.

3 - REQUIREMENTS

3.1 - General

The microcircuits are in accordance with the applicable document and as specified herein.

3.2 - Design and construction

3.2.1 - Terminal connections

Depending on the package, the terminal connections shall be is shown in § 10.1and § 10.2.

3.2.2 - Lead material and finish

Lead material and finish shall be any option of MIL-M-38510 except finish C (as described in 3.5.6.1 of 38510).

3.2.3 - Package

The macrocircuits are packaged in a hermetically sealed ceramic packages which are conform to case outlines of MIL-M-38510 appendix C (when defined) :

- 40 leads DIP Ceramic Side Brazed,
- 40 leads DIP Ceramic Cerdip,
- 44 terminals SQ LCCC.

The precise case outlines are described on Figures § 9.1, 9.2 and 9.3.

3.3 - Electrical characteristics

3.3.1 - Absolute maximum ratings (See Table 1)

Table 1

Symbol	Parameter		Test conditions	Min	Max	Unit
V_{CC}	Supply voltage			−0.3	+7.0	V
V_I	Input voltage			−0.3	+7.0	V
P_{dmax}	Max power dissipation		T_{case} = −55°C / +125°C		1.5	W
T_{case}	Operating temperature	M suffix : EF 6800/EF 68A00	f = 1 and 1.5 MHz	−55	+125	°C
		V suffix : EF 6800/EF 68A00	f = 1 and 1.5 MHz	−40	+85	°C
		No suffix : EF 6800/EF 68A00/ EF 68B00	f = 1, 1.5 and 2 MHz	0	+70	°C
T_{stg}	Storage temperature			−55	+150	°C
T_j	Junction temperature				+170	°C
T_{leads}	Lead temperature		Max 5 sec. soldering		+270	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electrical fields ; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{CC}$. Input protection is enhanced by connecting unused inputs to either V_{DD} or V_{SS}.

3.4 - Thermal characteristics (at 25°C)

Table 2

Package	Symbol	Parameter	Value	Unit
DIL 40	θ JA	Thermal resistance - Ceramic junction to ambient	45	°C/W
J and C suffix	θ JC	Thermal resistance - Ceramic junction to case	15	°C/W
LCCC 44	θ JA	Thermal resistance - Ceramic junction to ambient	40	°C/W
E suffix	θ JC	Thermal resistance - Ceramic junction to case	15	°C/W

Power considerations

The average chip-junction temperature, T_J, in °C can be obtained from :

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

T_A = Ambient Temperature, °C.

θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W.

$P_D = P_{INT} + P_{I/O}$.

$P_{INT} = I_{CC} \times V_{CC}$, Watts – Chip Internal Power.

$P_{I/O}$ = Power Dissipation on Input and Output Pins – User Determined.

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is :

$$P_D = K : (T_J + 273) \quad (2)$$

Solving equations (1) and (2) for K gives :

$$K = P_D \bullet (T_A + 273) + \theta_{JA} \bullet P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA}, representing the barrier to heat flow from the semiconductor junction to the package (case), surface (θ_{JC}) and from the case to the outside ambient (θ_{CA}). These terms are related by the equation :

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling and thermal convection. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in equation (1) will result in a lower semiconductor junction temperature.

3.5 - Mechanical and environment

The microcircuits shall meet all mechanical environmental requirements of MIL-STD-833 for class B devices or TMS standards.

3.6 - Marking

The document where are defined the marking are identified in the related reference documents. Each microcircuit are legible and permanently marked with the following information as minimum :

3.6.1 - Thomson logo

3.6.2 - Manufacturer's part number

3.6.3 - Class B identification

3.6.4 - Date-code of inspection lot

3.6.5 - ESD identifier if available

3.6.6 - Country of manufacturing

4 · QUALITY CONFORMANCE INSPECTION

4.1 · MIL-STD-883

Is in accordance with MIL-M-38510 and method 5005 of MIL-STD-883. Group A and B inspections are performed on each production lot. Group C and D inspection are performed on a periodical basis.

5 · ELECTRICAL CHARACTERISTICS

5.1 · General requirements

All static and dynamic electrical characteristics specified for inspection purpose, and the relevant measurements :

– Conditions are given below.

Table 3 : Static electrical characteristics for all electrical variants. See § 5.2.

Table 4 : Dynamic electrical characteristics. See § 5.3.

5.2 · Static characteristics

Table 3 - DC electrical characteristics

V_{CC} = 5.0 V_{dc} ±5 % ; V_{SS} = 0 V_{dc} ; T_c = −55°C / +125°C or −40°C / +85°C or 0°C / +70°C
(unless otherwise specified)

Symbol	Characteristics		Min	Typ	Max	Unit
V_{IH}	Input high voltage	Logic	V_{SS} +2.0		V_{CC}	V
V_{IHC}		$\phi1$, $\phi2$	V_{CC}−0.6		V_{CC}+0.3	V
V_{IL}	Input low voltage	Logic	V_{SS} −0.3		V_{SS} +0.8	V
V_{ILC}		$\phi1$, $\phi2$	V_{SS} −0.3		V_{SS} +0.4	V
I_{in}	Input leakage current					
	(V_{in} = 0 to 5.25 V, V_{CC} = Max)	Logic		1.0	2.5	µA
	(V_{in} = 0 to 5.25 V, V_{CC} = 0 V to 5.25 V)	$\phi1$, $\phi2$			100	µA
I_{IZ}	HI-Z input leakage current	D0-D7		2.0	10	µA
	(V_{in} = 0.4 to 2.4 V, V_{CC} = Max)	A0-A15, R/$\overline{W}$			100	µA
V_{OH}	Output high voltage					
	(I_{Load} = −205 µA, V_{CC} = Min)	D0-D7	V_{SS} +2.4			V
	(I_{Load} = −145 µA, V_{CC} = Min)	A0-A15, R/$\overline{W}$, VMA	V_{SS} +2.4			V
	(I_{Load} = −100 µA, V_{CC} = Min)	BA	V_{SS} +2.4			V
V_{OL}	Output low voltage (I_{Load} = 1.6 mA, V_{CC} = Min)				V_{SS} +0.4	V
P_{INT}	Internal power dissipation (measured at T_A = T_L)			0.5	1.0	W
C_{in}	Capacitance					
	(V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	$\phi1$		25	35	pF
		$\phi2$		45	70	pF
		D0-D7		10	12.5	pF
		Logic inputs		6.5	10	pF
C_{out}		A0-A15, R/$\overline{W}$, VMA			12	pF

5.3 - Dynamic (switching) characteristics

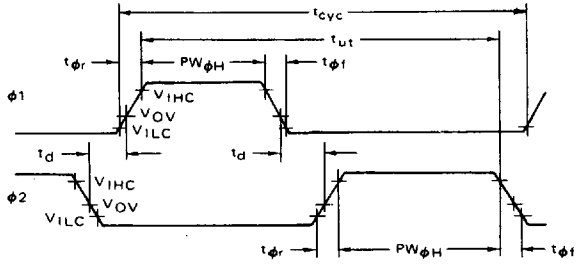
The limits and values given in this section apply over the full case temperature range −55°C to +125°C and V_{CC} in the range 4.75 V to 5.25 V, V_{IL} = 0.8 V and V_{IH} = 2.0 V.

For all tables below parameters are given for T_C from −55°C to +125°C, from −40°C to +85°C and from 0°C to +70°C for 1 MHz (EF 6800) and 1.5 MHz (EF 68A00) devices and for T_C from 0°C to +70°C for 0-70°C (EF 68B00) version.

Table 4 - Clock timing

V_{CC} = 5.0 V ±5 % ; V_{SS} = 0

Symbol	Characteristics		Min	Typ	Max	Unit
f	Frequency of operation	EF 6800	0.1		1.0	MHz
		EF 68A00	0.1		1.5	MHz
		EF 68B00	0.1		2.0	MHz
t_{cyc}	Cycle time (Figure 2)	EF 6800	1.000		10	µs
		EF 68A00	0.666		10	µs
		EF 68B00	0.500		10	µs
$PW_{\phi H}$	Clock pulse width	$\phi1$, $\phi2$ - EF 6800	400		9500	ns
	(Measured at V_{CC} −0.6 V)	$\phi1$, $\phi2$ - EF 68A00	230		9500	ns
		$\phi1$, $\phi2$ - EF 68B00	180		9500	ns
t_{ut}	Total $\phi1$ and $\phi2$ up time	EF 6800	900			ns
		EF 68A00	600			ns
		EF 68B00	440			ns
t_r, t_f	Rise and fall time (Measured between V_{SS} +0.4 V and V_{CC} −0.6 V)				100	ns
t_d	Delay time or clock separation (Figure 2)					
	(Measured at V_{OV} = V_{SS} +0.6 V @ t_r = t_f ⩽ 100 ns)		0		9100	ns
	(Measured at V_{OV} = V_{SS} +1.0 V @ t_r = t_f ⩽ 35 ns)		0		9100	ns



Note 1: Voltage levels shown are $V_L \leqslant 0.4$, $V_H \geqslant 2.4$ V, unless otherwise specified.
Note 2: Measurement points shown are 0.8 V and 2.0 V, unless otherwise noted.

Figure 2: Clock timing waveform.

Table 5 - Read / write timing (Reference Figures 3 through 7, 8, 9, 11, 12 and 13)

Symbol	Characteristic	EF 6800			EF 68A00			EF 68B00			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
t_{AD}	Address Delay C = 90 pF C = 30 pF			 270 250			 180 165			 150 135	 ns ns
t_{acc}	Peripheral read access time $t_{acc} = t_{ut} - (t_{AD} + t_{DSR})$	 605			 400			 290			 ns
t_{DSR}	Data setup time (read)	100			60			40			ns
t_H	Input data hold time	10			10			10			ns
t_H *	Output data hold time	10	25		10	25		10	25		ns
t_{AH}	Address hold time (Address, R/W, VMA) see Note	30	50		30	50		30	50		ns
t_{EH}	Enable high time for DBE input	450			280			220			ns
t_{DDW}	Data delay time (Write)			225			200			160	ns
	Processor controls										
t_{PCS}	Processor control setup time	200			140			110			ns
t_{PCr}, t_{PCf}	Processor control rise and fall time			100			100			100	ns
t_{BA}	Bus available delay			250			165			135	ns
t_{TSE}	Hi-Z Enable	0		40	0		40	0		40	ns
t_{TSD}	Hi-Z Delay			270			270			220	ns
t_{DBE}	Data bus enable down time during $\phi 1$ up time	150			120			75			ns
t_{DBEr}, t_{DBEf}	Data bus enable rise and fall times			25			25			25	ns

Note : For −40°C to + 85°C and −55°C to +125°C, t_{AH} = 10 ns (min) and 30 ns (typ).

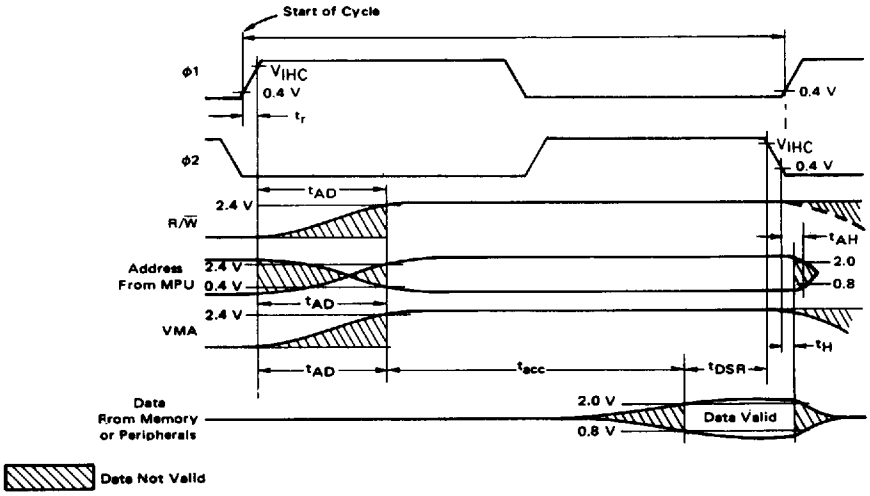
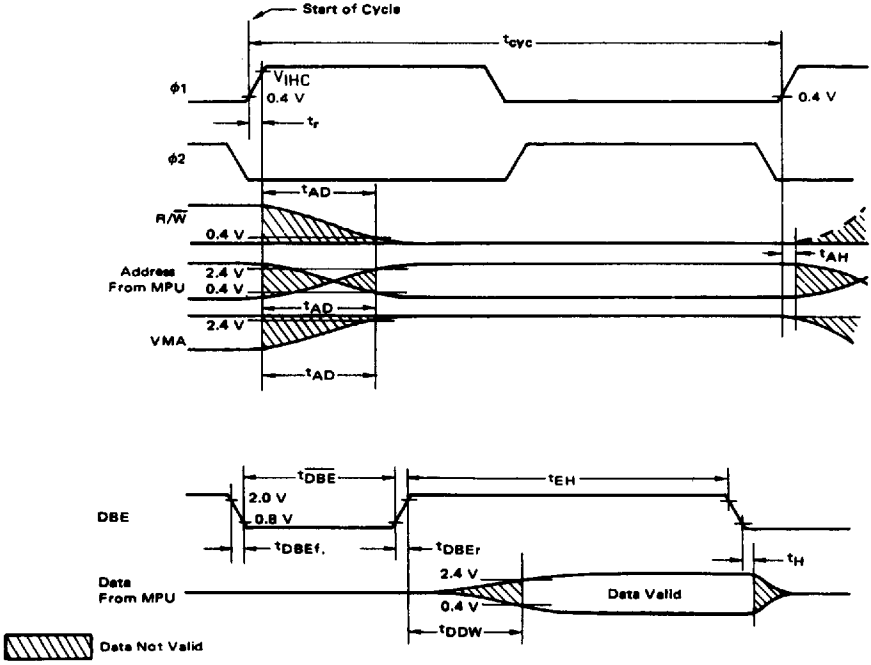


Figure 3 : Read data from memory or peripherals.



Note 1 : Voltage levels shown are $V_L \leq 0.4$, $V_H \geq 2.4$ V, unless otherwise specified.
Note 2 : Measurement points shown are 0.8 V and 2.0 V, unless otherwise noted.

Figure 4 : Write in memory or peripherals.

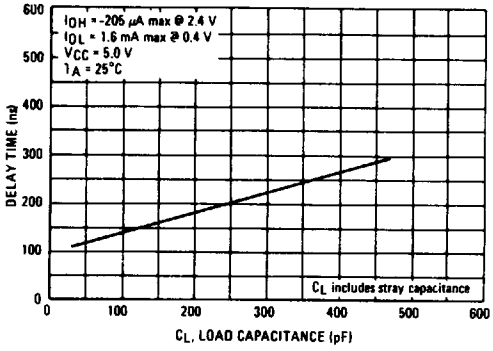


Figure 5 : Typical data bus output delay versus capacitive loading (t_{DDW}).

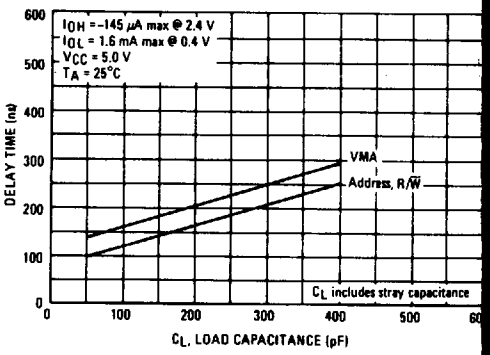
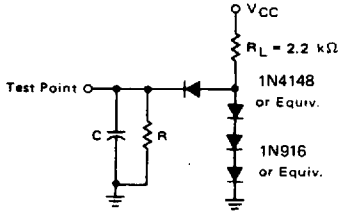


Figure 6 : Typical read/write, VMA, and address output delay versus capacitive loading (t_{AD}).

5.4 · Test conditions specific to the device

5.4.1 · Loading network

The dynamic test load for the data bus is 130 pF and one standard TTL load as shown. The address, R/$\overline{W}$, and VMA outputs are tested under two conditions to allow optimum operation in both buffered and unbuffered systems. The resistor (R) is chosen to insure specified load currents during V_{OH} measurement.
Notice that the data bus lines, the address lines, the interrupt request line, and the DBE line are all specified and tested to guarantee 0.4 V of dynamic noise immunity at both «1» and«0» logic levels.



C = 130 pF for D0-D7, E
= 90 pF for A0-A15, R/$\overline{W}$, and VMA
(Except t_{AD2})
= 30 pF for A0-A15, R/$\overline{W}$, and VMA
(t_{AD2} only)
= 30 pF for BA
R = 11.7 kΩ for D0-D7
= 16.5 kΩ for A0-A15, R/$\overline{W}$, and VMA
= 24 kΩ for BA

Figure 7 : Bus timing test loads.

6 - FUNCTIONAL DESCRIPTION

6.1 - MPU signal description

Proper operation of the MPU requires that certain control and timing signals be provided to accomplish specific funtions and that other signal lines be monitored to determine the state of the processor.

Clocks Phase One and Phase Two (ϕ1, ϕ2) - Two pins are used for a two-phase non-overlapping clock that runs at the V_{CC} voltage level.

Figure 2 shows the microprocessor clocks. The high level is specified at V_{IHC} and the low level is specified at V_{ILC} The allowable clock frequency is specified by f (frequency). The minimum ϕ1 and ϕ2 high level pulse widths are specified by $PW_{\phi H}$ (pulse width high time). To guarantee the required access time for the peripherals, the clock up time, t_{ut}, is specified. Clock separation, t_d, is measured at a maximum voltage of V_{OV} (overlap voltage). This allows for multitude of clock variations at the system frequency rate.

Address Bus (A0-A15) - Sixteen pins are used for the address bus. The outputs are three-state bus drivers capable of driving one standard TTL load an 90 pF. When the output is turned off, it is essentially an open circuit. This permits the MUP to be used in DMA applications. Putting TSC in its high state forces the Address bus to go into the three-state mode.

Data Bus (D0-D7) - Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state outupt buffers capable of driving one standard TTL load and 130 pF. Data Bus is placed in the three-state mode when DBE is low.

Data Bus Enable (DBE) - This level sensivite input is the three-state control signal for the MPU data bus and will enable the bus drivers when in the high state. This input is TTL compatible ; however in normal operation, it would be driven by the phase two clock. During an MPU read cycle, the data bus drivers will be disabled internally. When it is desired that another device control the data bus, such as in Direct Memory Access (DMA) applications, DBE should be held low.

If additional data setup or hold time is required on an MUP write, the DBE down time can be decreased, as show in Figure 4 (DBE $\neq \phi$2). The minimum down time for DBE is t_{DBE} as shown. By skewing DBE with respect to E, data setup or hold time can be increased.

Bus Available (BA) - The Bus Available signal will normally be in the low state ; when activated, it will go to the high state indicating that the microprocessor has stopped and that the address bus is available. This will occur if the HALT line is in the low state or the processor is in the WAIT state as a result of the execution of WAIT a instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level. The processor is removed from the WAIT state by the occurence of a maskable (mask bit I = 0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30 pF. If TSC is in the high state, Bus Available will be low.

Read/Write (R/$\overline{W}$) - This TTL compatible output signals the peripherals and memory devices wether the MUP is in a Read (high) or Write (low) state. The normal standby state of this signal is Read (high). Three-State Control going high will turn Read/Write to the off high impedance) state. Also, when the processor is halted, it will be in the off state. This output is capable of driving one standard TTL load and 90 pF.

$\overline{RESET}$ - The $\overline{RESET}$ input is used to reset and start the MPU from a power down condition resulting from a power failure or initial start-up of the processor. This level sensitive input can also be used to reinitialize the machine at any time after star t-up.

If a high level is detected in this input, this will signal the MPU to begin the reset sequence. During the reset sequence, the contents of the last two locations (FFFE, FFFF) in memory will be loaded into the Program Counter to point to the beginning of the reset routine. During the reset routine, the interrupt mask bit is set and must be cleared under program control before the MPU can be interrupted by $\overline{IRQ}$. While $\overline{RESET}$ is low (assuming a minimum of 8 clock cycles have occured) the MPU output signals will be in the following states : VAM = low, BA = low, Data Bus = high impedance, R/$\overline{W}$ = high (read state), and the Address Bus will contain the reset address FFFE. Figure 8 illustrates a power up sequence using the $\overline{RESET}$ control line. After the power supply reaches 4.75 V, a minimum of eight clock cycles are required for the processor to stabilize in preparation for restarting. During these eight cycles, VMA will be in an indeterminate state so any devices that are enabled by VMA which could accept a false write during this time (such as battery-backed RAM) must be disabled until VMA is forced low after eight cycles. $\overline{RESET}$ can go high asynchronously with the system clock any timeafter the aight cycle.

$\overline{RESET}$ timing is shown in Figure 8. The maximum rise and fall transition times are specified by t_{PCr} and t_{PCt}. If $\overline{RESET}$ is high at t_{PCS} (processor control setup time), as shown in Figure 8, in any given cycle then the restart sequence will begin on the next cycle as shown. The $\overline{RESET}$ control line may also be used to reinitialize the MPU system at any time during its operation. This is accomplished by pulsing $\overline{RESET}$ low for the duration of a minimum of three complete ϕ2 cycles. The $\overline{RESET}$ pulse can be completely asynchronous with the MPU system clock and will recognized during ϕ2 if setup time t_{PCS} is met.

Interrupt Request ($\overline{IRQ}$) - This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next, the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory. Interrupt timing is shown in Figure 9.

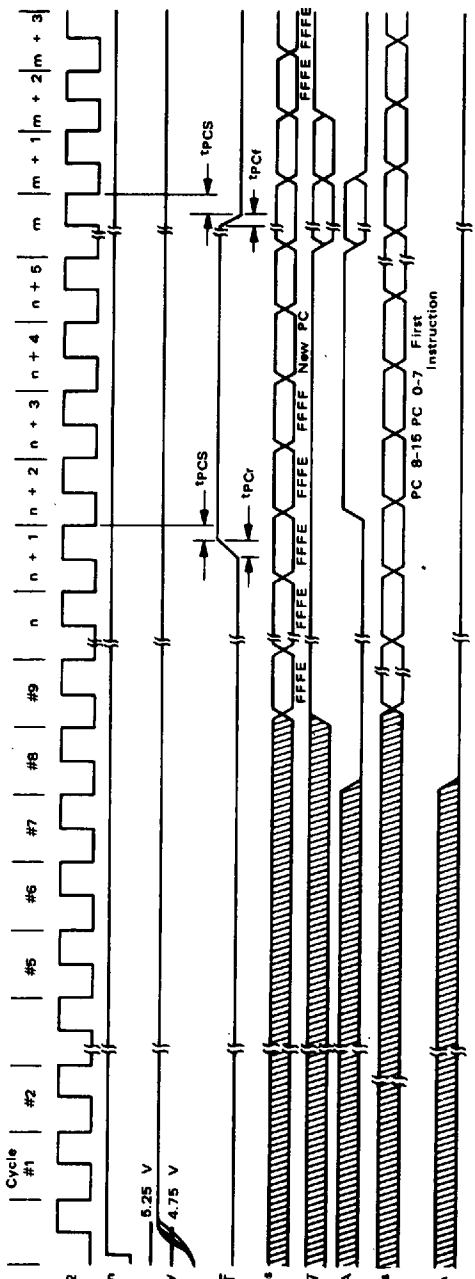


Figure 8 : RESET timing.

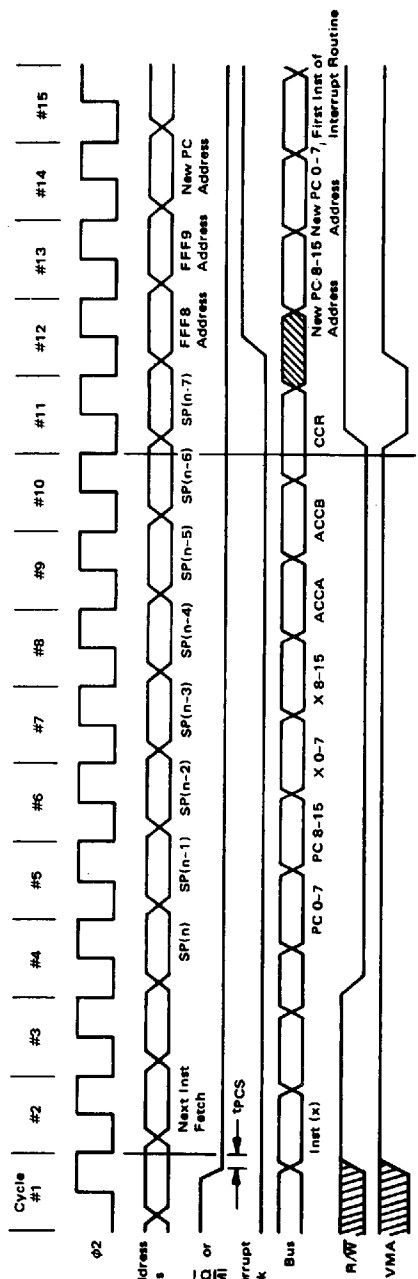


Figure 9 : Interrupt timing.

The $\overline{HALT}$ line must be in the high state for interrupts to be serviced. Interrupts will be latched internally while $\overline{HALT}$ is low.

The $\overline{IRQ}$ has a high-impendance pullup device internal to chip ; however, a 3 kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Non-Maskable Interrupt ($\overline{NMI}$) and Wait for Interrupt (WAI) - The EF 6800 is capable of handling two types of interrupts : maskable ($\overline{IRQ}$) as described earlier, and nonmaskable ($\overline{NMI}$) which is an edge sensitive input. $\overline{IRQ}$ is maskable by the interrupt mask in the condition code register while $\overline{NMI}$ is not maskable. The handling of these interrupts by the MPU is the same except that each has its own vector address. The behavior of the MPU when Interrupted is shown in Figure 9 which details the MPU response to an interrrupt while the MPU is executing the control program. The interrupt shown could be either $\overline{IRQ}$ or $\overline{NMI}$ and can be asynchronous with respect to $\phi 2$. The interrupt is shown going low at time t_{PCS} in cycle = 1 which precedes the first cycle of an instruction (OP code fetch). This instruction is not executed but instead the Program Counter (PC), Index Register (IX), Accumulators (ACCX), and the Condition Code Register (CCR) are pusched onto the stack.

The interrupt mask bit is set to prevent further interrupts. The address of the interrupt service routine is then fetched from FFFC, FFFFD for an NMI interrupt and from FFF8, FFF9 for an IRQ interrupt. Upon completion of the interrupt service routine, the execution of RTI will pull the PC, IX, ACCX, and CCR off the stack ; the interrupt mask bit is restored to its condition prior to interrupts (see Figure 10).

Figure 11 is similar interrupt sequence, except in this case, a WAIT instruction has been executed in preparation for the interrupt. This technique speeds up the MPU's response to the interrupt because the stacking of the PC, IX, ACCX, and the CCR is already done. While the MPU is waiting for the interrupt, bus available will go high indicating the following states of the control lines : VMA is low, and the address bus, R/W and data bus are all in the high impedance state. After the interrupt occurs, it is serviced as previously described.

A 3-10 kΩ external resistor to V_{CC} should be used for wire OR and optimum control of interrupts.

MEMORY MAP FOR INTERRUPT VECTORS

Vector		Description
MS	LS	
FFFE	FFFF	Reset
FFFC	FFFD	Non-Maskable Interrupt
FFFA	FFFB	Software Interrupt
FFF8	FFF9	Interrupt Request

Refer to Figure 10 for program flow for Interrupts.

5

Three-State Control (TSC) - When the level sensitive Three-State Control (TSC) line is a logic «1», the address bus and the R/W line are placed in a high-impedance state. VMA and BA are forced low when TSC = «1» to prevent false reads or writes on any device enabled by VMA. It is necessary to delay program execution while TSC is held high. This is done by insuring that no transitions of $\phi 1$ (or $\phi 2$) occur during this period. (Logic levels of the clocks are irrelevant so long as they do not change). Since the MPU is a dynamic device, the $\phi 1$ clock can be stopped for a maximum time $PW_{\phi H}$ without destroying data within the MPU. TSC then can be used a short Direct Memory Access (DMA) application.

Figure 12 shows the effect of TSC on the MPU. TSC must have its transitions at t_{TSE} (three-state enable) while holding $\phi 1$ high and $\phi 2$ low as shown. The addres bus and R/W line will reach the high-impedance state at t_{TSD} (three-state delay), with VMA being forced low. In this exemple, the data bus is also in the high-impedance state while $\phi 2$ is being held low since DBE # $\phi 2$. At this point in time, a DMA transfer could occur on cycles #3 and #4. When TSC is returned low, the MPU address and R/W lines return to the bus. Because it is too late in cycle #5 to access memory, this cycle is dead and used for synchronization. Program execution resumes in cycle #6.

Valid Memory Address (VMA) - This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling perpheral interfaces such as the PIA and ACIA. This signal is not three-state. On standard TTL load and 90 pF may be directly driven by this active high signal.

$\overline{HALT}$ - When this level sensitive input is in the low state, all activity in the machine will be halted. This input is level sensitive.

The $\overline{HALT}$ line provides an input to the MPU to allow control of program execution by an outside source. If $\overline{HALT}$ is high, the MPU will execute the instructions ; if it is low, the MPU will go to a halted or idle mode. A response signal, Bus Available (BA) provides an indication of the current MPU status. When BA is low, the MPU is in the process of executing the control program ; if BA is high, the MPU has halted and all internal activity has stopped.

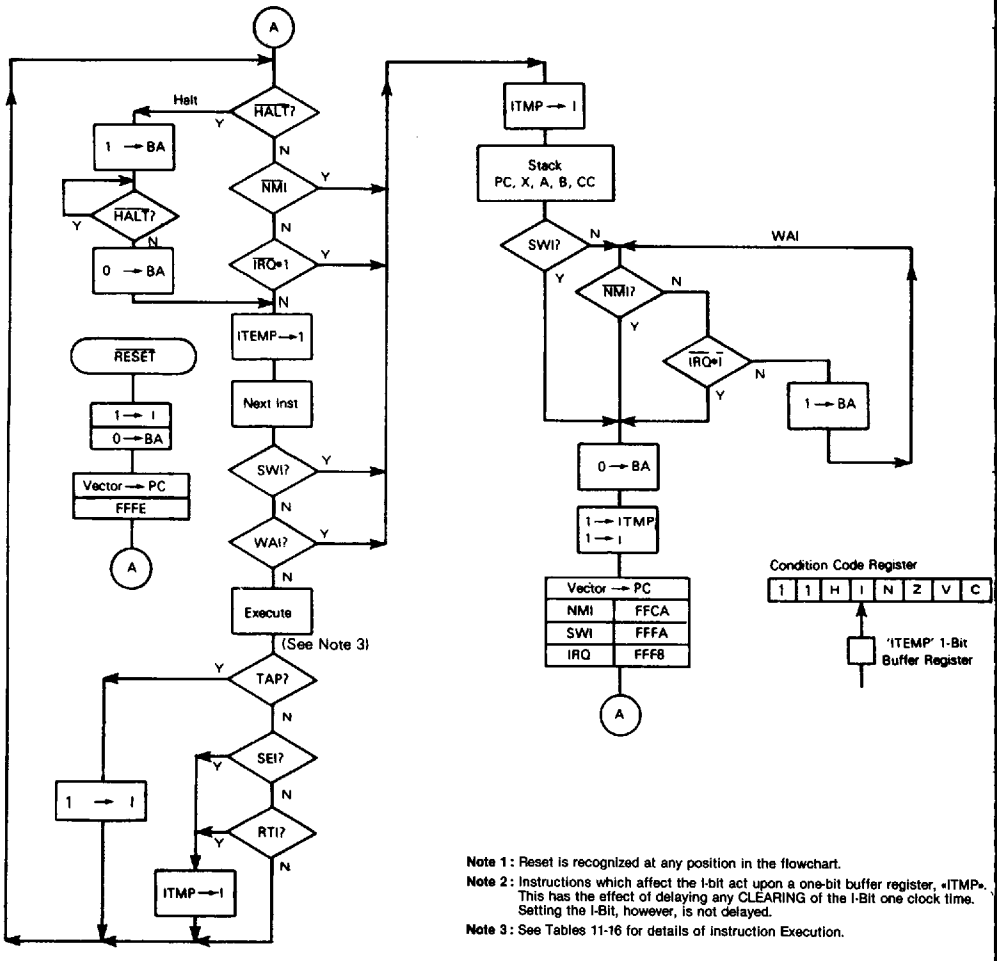
When BA is high, the address bus, data bus, and R/W line will be in high-impedance state effectively removing the MPU from the systems bus. VMA is forced low so that the floating system bus will not activate any device on the bus that is enabled by VMA.

While the MPU is halted, all program activity is stopped, and if either an $\overline{NMI}$ or $\overline{IRQ}$ interrupt occurs, it will be latched into the MPU and acted on as soon as the MPU is taken out of the halted mode. If a $\overline{RESET}$ command occurs while the MPU is halted, the following states occur : VMA = low, BA = low, data bus = high impedance, R/W = high (read state), and the address bus will contain address FFFE as long as RESET is low. As soon as the $\overline{RESET}$ line goes high, the MPU will go to locations FFFE = FFFF for the address of the reset routine.

Figure 13 shows the timing relationships involved when halting the MPU. The instruction illustrated is a one byte, 2 cycle instruction such as CLRA. When $\overline{HALT}$ goes low, the MPU will halt after completing execution of the current instruction. The transition of $\overline{HALT}$ must occur t_{PCS} before the trailing edge of $\phi 1$ of the last cycle of an instruction (point A of Figure 13). $\overline{HALT}$ must not go low any time later than minmum t_{PCS} specified.

The fetch of the OP code by the MPU is the first cycle of the instruction. If $\overline{HALT}$ had not been low at Point A but went low during $\phi 2$ of that cycle, the MPU would have halted after completion of the following instruction. BA will go high by time t_{BA} (bus available delay time) after the last instruction cycle. At this point in time, VMA is low and R/$\overline{W}$, address bus, and the data bus are in the high-impedance state.

To debug programs it is advantageous to step through programs instruction by instruction. To do this, $\overline{HALT}$ must be brought high for one MPU cycle and then returned low as shown at point B of Figure 13. Again, the transitions of $\overline{HALT}$ must occur t_{PCS} before the trailing edge of $\phi 1$. BA will go low at t_{BA} after the leading edge of the next $\phi 1$, indicating that the address bus, data bus, VMA and R/$\overline{W}$ lines are back on the bus. A single byte, 2 cycle instruction such as LSR is used for this example also. During the first cycle, the instruction Y is fetched from address M + 1. BA returns high at t_{BA} on the last cycle of the instruction indicating the MPU is off the bus. If instruction Y had been three cycles, the width of the BA low time would have been increased by one cycle.

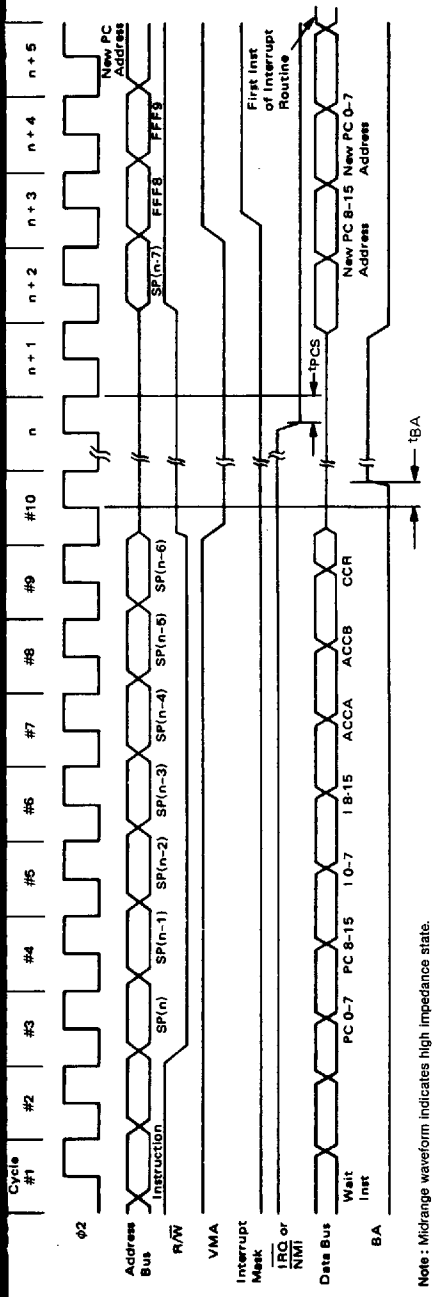


Note 1: Reset is recognized at any position in the flowchart.

Note 2: Instructions which affect the I-bit act upon a one-bit buffer register, «ITMP». This has the effect of delaying any CLEARING of the I-Bit one clock time. Setting the I-Bit, however, is not delayed.

Note 3: See Tables 11-16 for details of instruction Execution.

Figure 10: MPU flowchart.



Note : Midrange waveform indicates high impedance state.

Figure 11 : Wait instruction timing.

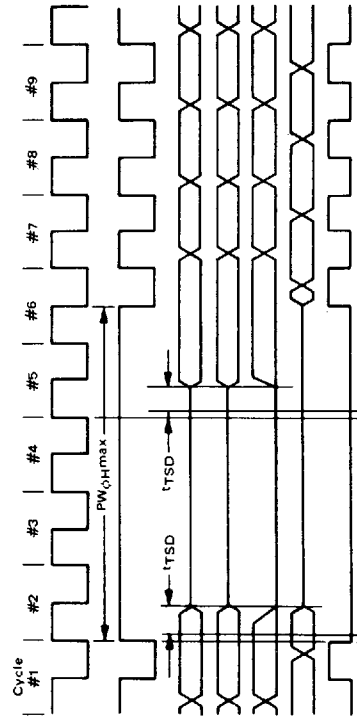
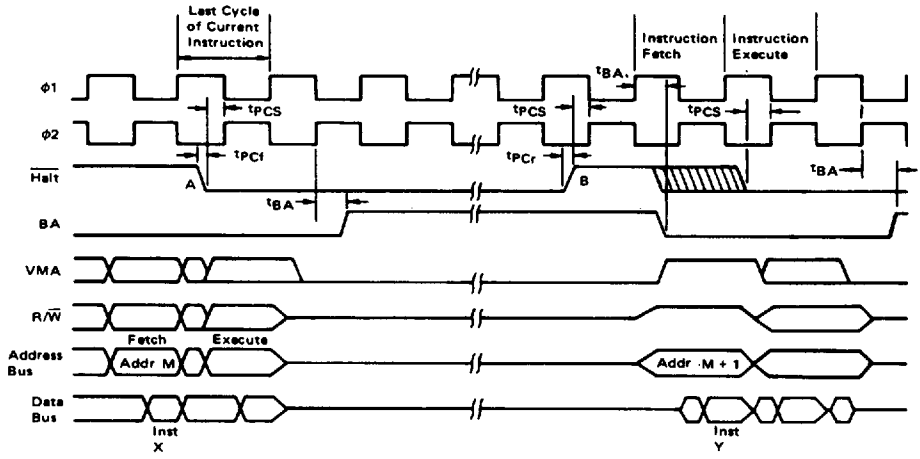


Figure 12 : Three-state control timing.

5



Note : Midrange waveform indicates high impedance state.

Figure 13 : $\overline{\text{HALT}}$ and single instruction execution for system debug.

6.2 - MPU registers

The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer (Figure 14).

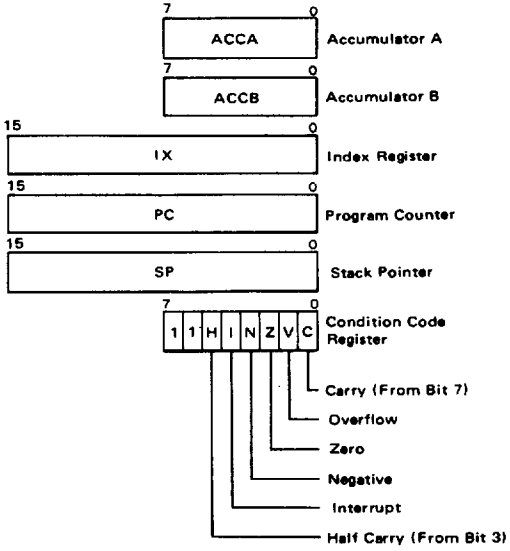


Figure 14: Programming model of the microprocessing unit.

Program Counter - The program counter is a two byte (16 bits) register that points to the current program address.

Stack Pointer - The stack-pointer is a two byte register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be nonvolatile.

Index Register - The index register is a two register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

Accumulators - The MPU contains two 8-bit accumulators that are used to hold operands and resulte from an arithmetic unit (ALU).

Condition Code Register - The condition code register indicates the results of an Arithmetic Logic Unit operation : Negative (N), Zero (Z), Overflow (V), Carry from bit 7 (C), and half carry from bit 3 (H). These bits of the Condition Code Register are used as testable conditions for the condition branch instructions. Bit 4 is the interrupt mask bit (I). The unused bits of the Condition Code Register (b6 and b7) are ones.

6.3 - MPU instruction set

The EF 6800 instructions are described in detail in the EF 6800 programming manual. This section will provide a brief introduction and discuss their use in de velopping EF 6800 control programs. The EF 6800 has a set of 72 different executable source instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

Each of the 72 executable instructions of the source language assembles into 1 to 3 bytes of machine code. The number of bytes depends on the particular instruction and on the addressing mode. (The addressing modes which are available for use with the various executive instructions are discussed later).

The coding of the first (or only) byte corresponding to an executable instruction is sufficient to identify the instruction and the addressing mode. The hexadecimal equivalents of the binary codes, which result from the translation of the 72 instructions in all valid modes of addressing, are shown in Table 6. There are 197 valid machine codes, 59 of the 256 possible codes being unassigned.

When an instruction translates into two or three bytes of code, the second byte, or the second and third bytes contain(s) an operand, an address, or information from which an address is obtained during execution.

Microprocessor instructions are often divided into three general classifications : (1) memory reference, so called because they operate on specific memory locations ; (2) operating instructions that function without needing a memory reference ; (3) I/O instructions for transferring data between the microprocessor and peripheral devices.

In many instances, the EF 6800 performs the same operation on both its internal accumulators and the external memory locations. In addition, the EF 6800 interface adapters (PIA and ACIA) allow the MPU to treat peripheral device exactly like other memory locations, hence, no I/O instructions as such are required. Because of these features, other classifications are more suitable for introducing the EF 6800's instruction set : (1) Accumulator and memory operations ; (2) Program control operations ; (3) Condition Code Register operations.

Table 6 - Hexadecimal values of machine codes

00	·			40	NEG	A		80	SUB	A	IMM	C0	SUB	B	IMM
01	NOP			41	·			81	CMP	A	IMM	C1	CMP	B	IMM
02	·			42	·			82	SBC	A	IMM	C2	SBC	B	IMM
03	·			43	COM	A		83	·			C3	·		
04	·			44	LSR	A		84	AND	A	IMM	C4	AND	B	IMM
05	·			45	·			85	BIT	A	IMM	C5	BIT	B	IMM
06	TAP			46	ROR	A		86	LDA	A	IMM	C6	LDA	B	IMM
07	TPA			47	ASR	A		87	·			C7	·		
08	INX			48	ASL	A		88	EOR	A	IMM	C8	EOR	B	IMM
09	DEX			49	ROL	A		89	ADC	A	IMM	C9	ADC	B	IMM
0A	CLV			4A	DEC	A		8A	ORA	A	IMM	CA	ORA	B	IMM
0B	SEV			4B	·			8B	ADD	A	IMM	CB	ADD	B	IMM
0C	CLC			4C	INC	A		8C	CPX	A	IMM	CC	·		
0D	SEC			4D	TST	A		8D	BSR		REL	CD	·		
0E	CLI			4E	·			8E	LDS		IMM	CE	LDX		IMM
0F	SEI			4F	CLR	A		8F	·			CF	·		
10	SBA			50	NEG	B		90	SUB	A	DIR	D0	SUB	B	DIR
11	CBA			51	·			91	CMP	A	DIR	D1	CMP	B	DIR
12	·			52	·			92	SBC	A	DIR	D2	SBC	B	DIR
13	·			53	COM	B		93	·			D3	·		
14	·			54	LSR	B		94	AND	A	DIR	D4	AND	B	DIR
15	·			55	·			95	BIT	A	DIR	D5	BIT	B	DIR
16	TAB			56	ROR	B		96	LDA	A	DIR	D6	LDA	B	DIR
17	TBA			57	ASR	B		97	STA	A	DIR	D7	STA	B	DIR
18	·			58	ASL	B		98	EOR	A	DIR	D8	EOR	B	DIR
19	DAA			59	ROL	B		99	ADC	A	DIR	D9	ADC	B	DIR
1A	·			5A	DEC	B		9A	ORA	A	DIR	DA	ORA	B	DIR
1B	ABA			5B	·			9B	ADD	A	DIR	DB	ADD	B	DIR
1C	·			5C	INC	B		9C	CPX		DIR	DC	·		
1D	·			5D	TST	B		9D	·			DD	·		
1E	·			5E	·			9E	LDS		DIR	DE	LDX		DIR
1F	·			5F	CLR	B		9F	STS		DIR	DF	STX		DIR
20	BRA		REL	60	NEG		IND	A0	SUB	A	IND	E0	SUB	B	IND
21	·			61	·			A1	CMP	A	IND	E1	CMP	B	IND
22	BHI		REL	62	·			A2	SBC	A	IND	E2	SBC	B	IND
23	BLS		REL	63	COM		IND	A3	·			E3	·		
24	BCC		REL	64	LSR		IND	A4	AND	A	IND	E4	AND	B	IND
25	BCS		REL	65	·			A5	BIT	A	IND	E5	BIT	B	IND
26	BNE		REL	66	ROR		IND	A6	LDA	A	IND	E6	LDA	B	IND
27	BEQ		REL	67	ASR		IND	A7	STA	A	IND	E7	STA	B	IND
28	BVC		REL	68	ASL		IND	A8	EOR	A	IND	E8	EOR	B	IND
29	BVS		REL	69	ROL		IND	A9	ADC	A	IND	E9	ADC	B	IND
2A	BPL		REL	6A	DEC		IND	AA	ORA	A	IND	EA	ORA	B	IND
2B	BMI		REL	6B	·			AB	ADD	A	IND	EB	ADD	B	IND
2C	BGE		REL	6C	INC		IND	AC	CPX		IND	EC	·		
2D	BLT		REL	6D	TST		IND	AD	JSR		IND	ED	·		
2E	BGT		REL	6E	JMP		IND	AE	LDS		IND	EE	LDX		IND
2F	BLE		REL	6F	CLR		IND	AF	STS		IND	EF	STX		IND
30	TSX			70	NEG		EXT	B0	SUB	A	EXT	F0	SUB	B	EXT
31	INS			71	·			B1	CMP	A	EXT	F1	CMP	B	EXT
32	PUL	A		72	·			B2	SBC	A	EXT	F2	SBC	B	EXT
33	PUL	B		73	COM		EXT	B3	·			F3	·		
34	DES			74	LSR		EXT	B4	AND	A	EXT	F4	AND	B	EXT
35	TXS			75	·			B5	BIT	A	EXT	F5	BIT	B	EXT
36	PSH	A		76	ROR		EXT	B6	LDA	A	EXT	F6	LDA	B	EXT
37	PSH	B		77	ASR		EXT	B7	STA	A	EXT	F7	STA	B	EXT
38	·			78	ASL		EXT	B8	EOR	A	EXT	F8	EOR	B	EXT
39	RTS			79	ROL		EXT	B9	ADC	A	EXT	F9	ADC	B	EXT
3A	·			7A	DEC		EXT	BA	ORA	A	EXT	FA	ORA	B	EXT
3B	RTI			7B	·			BB	ADD	A	EXT	FB	ADD	B	EXT
3C	·			7C	INC		EXT	BC	CPX		EXT	FC	·		
3D	·			7D	TST		EXT	BD	JSR		EXT	FD	·		
3E	WAI			7E	JMP		EXT	BE	LDS		EXT	FE	LOX		EXT
3F	SWI			7F	CLR		EXT	BF	STS		EXT	FF	STX		EXT

Note 1 : Addressing Modes.
A = Accumulator A
B = Accumulator B
REL = Relative
IND = Indexed
IMM = Immediate
DIR = Direct

Note 2 : Unassigned code indicated "·".

Table 7 · Accumulator and memory operations

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A — 0	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B — b7 b0 C	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B — 0 b7 b0 C	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

ADDRESSING MODES — BOOLEAN/ARITHMETIC OPERATION — COND. CODE REG.

LEGEND:

- OP Operation Code (Hexadecimal);
- ~ Number of MPU Cycles;
- # Number of Program Bytes;
- + Arithmetic Plus;
- − Arithmetic Minus;
- · Boolean AND;
- M_{SP} Contents of memory location pointed to be Stack Pointer;
- + Boolean Inclusive OR;
- ⊕ Boolean Exclusive OR;
- M Complement of M;
- → Transfer Into;
- 0 Bit = Zero;
- 00 Byte = Zero;

CONDITION CODE SYMBOLS:

- H Half-carry from bit 3;
- I Interrupt mask
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry from bit 7
- R Reset Always
- S Set Always
- ↕ Test and set if true, cleared otherwise
- • Not Affected

CONDITION CODE REGISTER NOTES:
(Bit set if test is true and cleared otherwise)

1. (Bit V) Test: Result = 10000000?
2. (Bit C) Test: Result = 00000000?
3. (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4. (Bit V) Test: Operand = 10000000 prior to execution?
5. (Bit V) Test: Operand = 01111111 prior to execution?
6. (Bit V) Test: Set equal to result of N⊕C after shift has occurred.

Note : Accumulator addressing mode instructions are included in the column for IMPLIED addressing.

6.4 - Program control operations

Program Control operation can be subdivided into two categories : (1) Index Register/Stack Pointer instructions ; (2) Jump and Branch operations.

Index Register/Stack Pointer Operations

The instruction for direct operation on the MPU's index register and stack pointer are summarized in Table 8. Decrement (DEX, DES), increment (INX, INS) load (LDX, LDS), and store (STX, STS) instruction are provided for both. The compare instruction, CPX, can be used to compare the index register to a 16-bit value and update the condition code register accordingly.

The TSX instruction causes the index register to be loaded with the address of the last data byte put onto the «stack».The TXS instruction loads the stack pointer with a value equal to one less than the current contents of the index register. This causes the next byte to be pulled from the «stack» to come from the location indicated by the index register. The utility of these two instructions can be clarified by describing the «stack» concept relative to the EF 6800 systems.

The «stack» can be throught of as a sequential list of data stored in the MPU's read/write memory. The stack pointer contains a 16-bit memory address that is used to access the list from one end on a last-in-first-out (LIFO) basis in contrast to the random access mode used by the MPU's other addressing modes.

The EF 6800 instruction set and interrupt structure allow extensive use of the stack concept for efficient handling of data movement, subroutines and interrupts. The instructions can be used to etablish one or more «stacks» anywhere in read/write memory. Stack lenght is limited only by the amount of memory that is made available.

Operation of the stack pointer with the Push and Pull instruction is illustrated in Figures 15 and 16. The Push instruction (PSHA) causes the contents of the indicated accumulator (A in this example) to be stored in memory at the location indicated by the stack pointer. The stack pointer is automatically decremented by one following the storage operation and is «pointing» to the next empty stack location. The Pull instruction (PULA or PULB) causes the last byte stacked to be loaded into the appropriate accumulator. The stack pointer is automaticcaly incremended by one just prior to the data transfer so that it will point to the last byte stacked rather than the next empty location. NOte that the PULL instruction does not «remove» the data from memory ; in the example, 1A is still in location (m + 1) following execution of PULA. A subsequent PUSH instruction would overwrite that location with the new «pushed» data.

Execution of the Branch to Subroutine (BSR) and Jump to Subroutine (JSR) instructions cause a return address to be saved on the stack as shown in Figure 18 through 20. The stack is decremented after each byte of the return address is pushed onto the stack. For both of these instructions, the return address is the memory location following the bytes of code that correspond to the BSR or JSR instruction. The code required for BSR or JSR may be either two or three bytes, depending on whether the JSR is in the indexed (two bytes) or the exented (three bytes) addressing mode. Before it is stacked, the Program Counter is automatically incremented the correct number of times to be pointing at the location of the next instruction. The return for subroutine instruction, RTS, causes the return address to be retrieved and loaded into the program counter as shown in Figure 21.

There are several operations that cause the status of the MPU to be saved on the stack. The Software Interrupt (SWI) and Wait for Interrupt (WAI) instructions as well as the maskable (IRQ) and non-maskable (NMI) hardware interrupts all cause the MPU's internal registers (except for the stack pointer itself) to be stacked as shown in Figure 23. MPU status restored by the return from interrupt, RTI, as shown in Figure 22.

Jump and Branch Operation

The jump and branch instructions are summarized in Table 9. These instructions are used to control the transfer or operation from one point another in the control program.

The No Operation instruction, NOP, while included here, is a jump operation in a very limited sense. Its only effect is to increment the Program Counter by one. It is useful during program development as a «stand-in» for some other instruction that is to be determined during debug. It is also used for equalizing the execution time through alternate paths in a control program.

Table 8 - Index register and stack pointer instructions

		IMMED			DIRECT			INDEX			EXTND			IMPLIED				COND. CODE REG.						
																		5	4	3	2	1	0
POINTER OPERATIONS	MNEMONIC	OP	~	=	OP	~	=	OP	~	=	OP	~	=	OP	~	=	BOOLEAN/ARITHMETIC OPERATION	H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M + 1)$	•	•	①	↕	②	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	•	•	③	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	•	•	③	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	•	•	③	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	•	•	③	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

① (Bit N) Test : Sign bit of most significant (MS) byte of result = 1 ?

② (Bit V) Test : 2's complement overflow from subtraction of ms bytes ?

③ (Bit N) Test : Result less than zero ? (Bit 15 = 1).

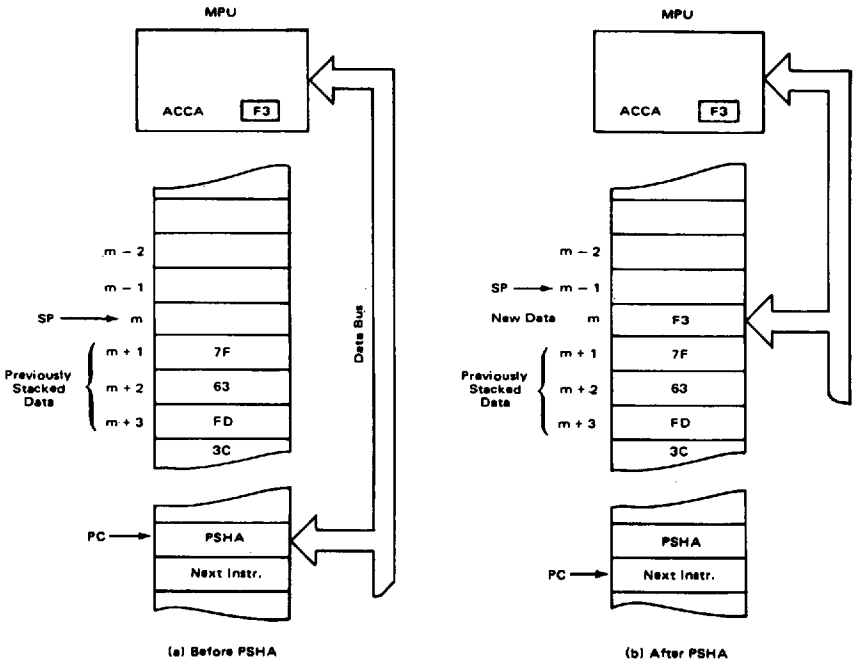


Figure 15 : Stack operation, push instruction.

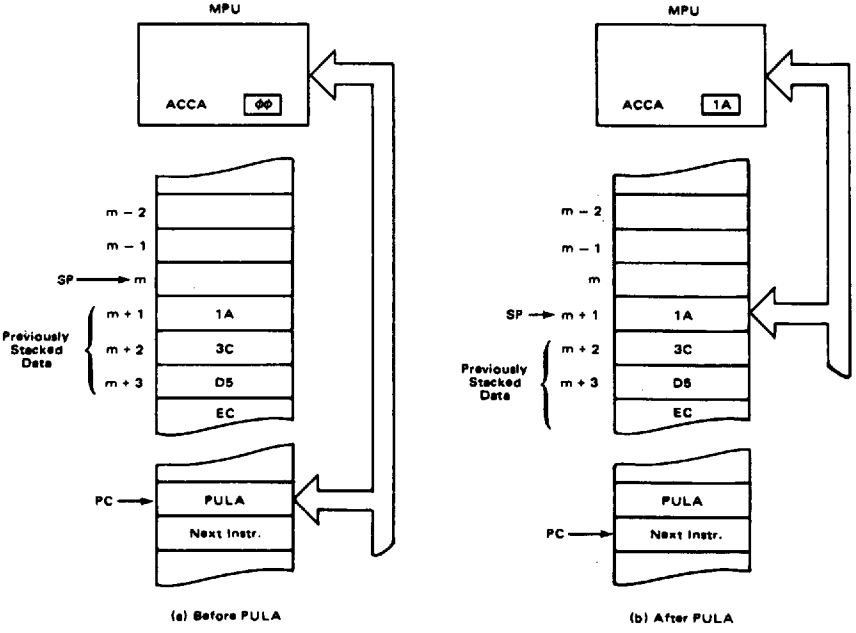


Figure 16 : Stack operation, pull instruction.

Table 9 · Jump and branch instructions

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		———			①	———	
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	②	•	•	•	•

*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

① (All) Load condition code register from stack. (See special operations).

② (Bit 1) Set when interrupt occurs. If previously set, a non-maskable interrupt is required to exit the wait state.

③ (Bit N) Test : Result less than zero ? (Bit 15 = 1).

Execution of the Jump Instruction, JMP, and Branch Always, BRA, affects program flow as shown in Figure 17. When the MPU encounters the jump (indexed) instruction, it adds the offset to the value in the index register and uses the result as the address of the next instruction to be executed. In the extended addressing mode, the address of the next instruction to be executed is fetched from the two locations immediately following the JMP instruction. The Branch Always (BRA) instruction is similar to the JMP (extended) instruction except that the relative addressing mode applies and the branch is limited to the range within - 125 or + 127 bytes of the branch instruction itself. The opcode for the BRA instruction requires one less byte than JMP (exended) but takes one more cycle to execute.

The effect on program flow for the Jump Subroutine (JSR) and Branch to Subroutine (BSR) is shown in Figures 18 through 20. Note that the program counter is properly incremented to be pointing at the correct return address before it is stacked. Operation of the branch to subroutine and jump to subroutine (extended) instruction is similar except for the range. The BSR instruction requires less opcode than JSR (2 bytes versus 3 bytes) and also executes one cycle faster than JSR. The return from Subroutine, RTS, is used as the end of a subroutine to return to the main program as indicated in Figure 21.

The effect of executing the Software Interrupt, SWI, and the Wait for Interrupt, WAI, and their relationship to the hardware interrupts is shown in Figure 22. SWI causes the MPU contents to be stacked and then fetches the starting address of the interrupt routine from the memory locations that respond to the addresses FFFA and FFFB. Note that as in the case of the subroutine instructions, the program counter is incremented to point at the correct return address before being stacked. The Return from Interrupt instruction, RTI, (Figure 22) is used at the end of an interrupt routine to restore control to the main program. The SWI instruction is useful for inserting break points in the control program, that is, it can be used to stop operation and put the MPU registers in memory where they can be examined. The WAI instruction is used to decrease the time required to service a hardware interrupt ; it stacks the MPU contents and then waits for the interrupt to occur, effectively removing the stacking time from a hardware interrupt sequence.

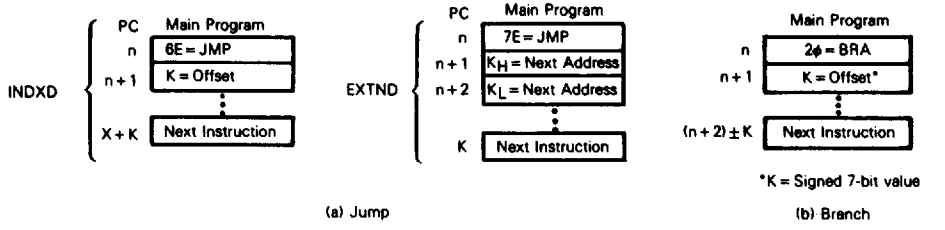


Figure 17 : Program flow for jump and branch instructions.

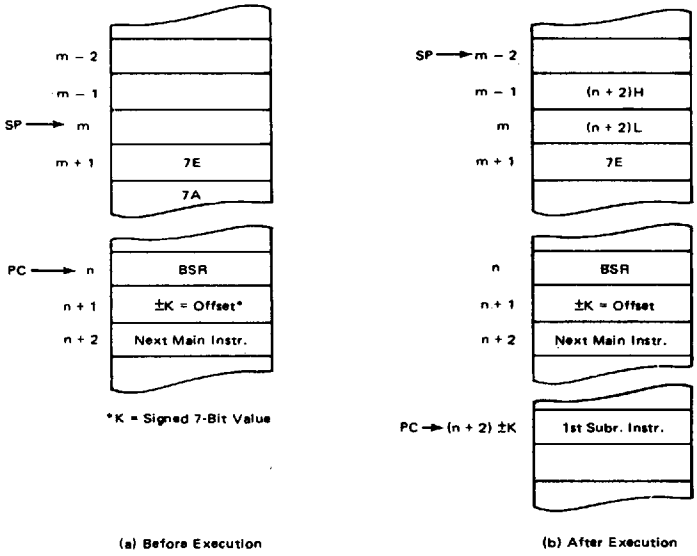


Figure 18: Program flow for BSR.

5

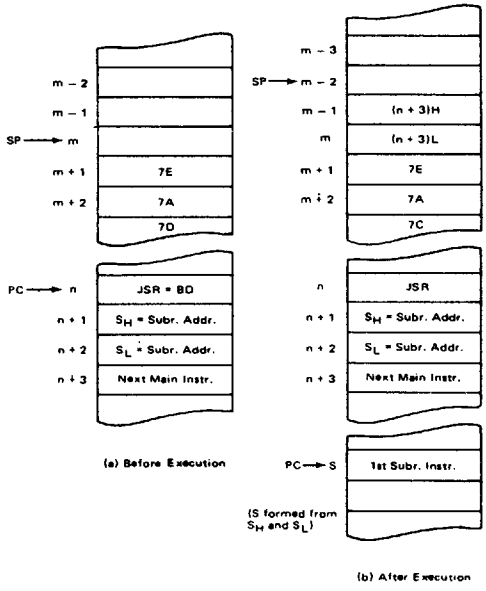


Figure 19: Program flow for JSR (extended).

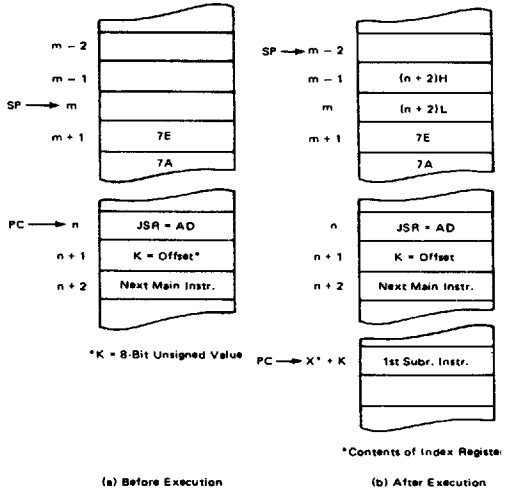


Figure 20: Program flow for JSR (indexed).

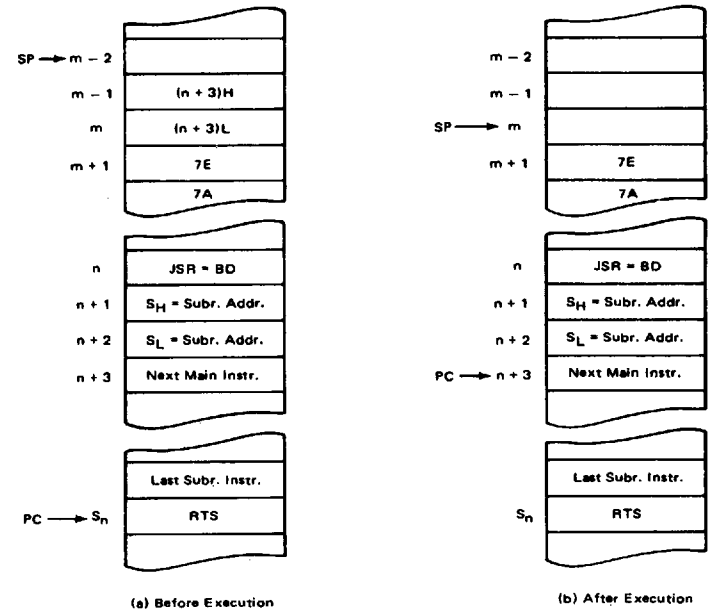


Figure 21 : Program flow for RTS.

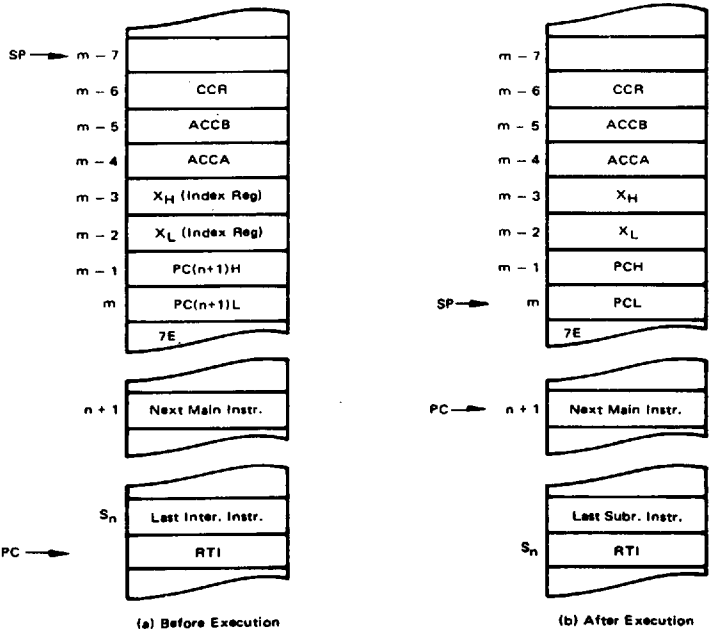


Figure 22 : Program flow for RTI.

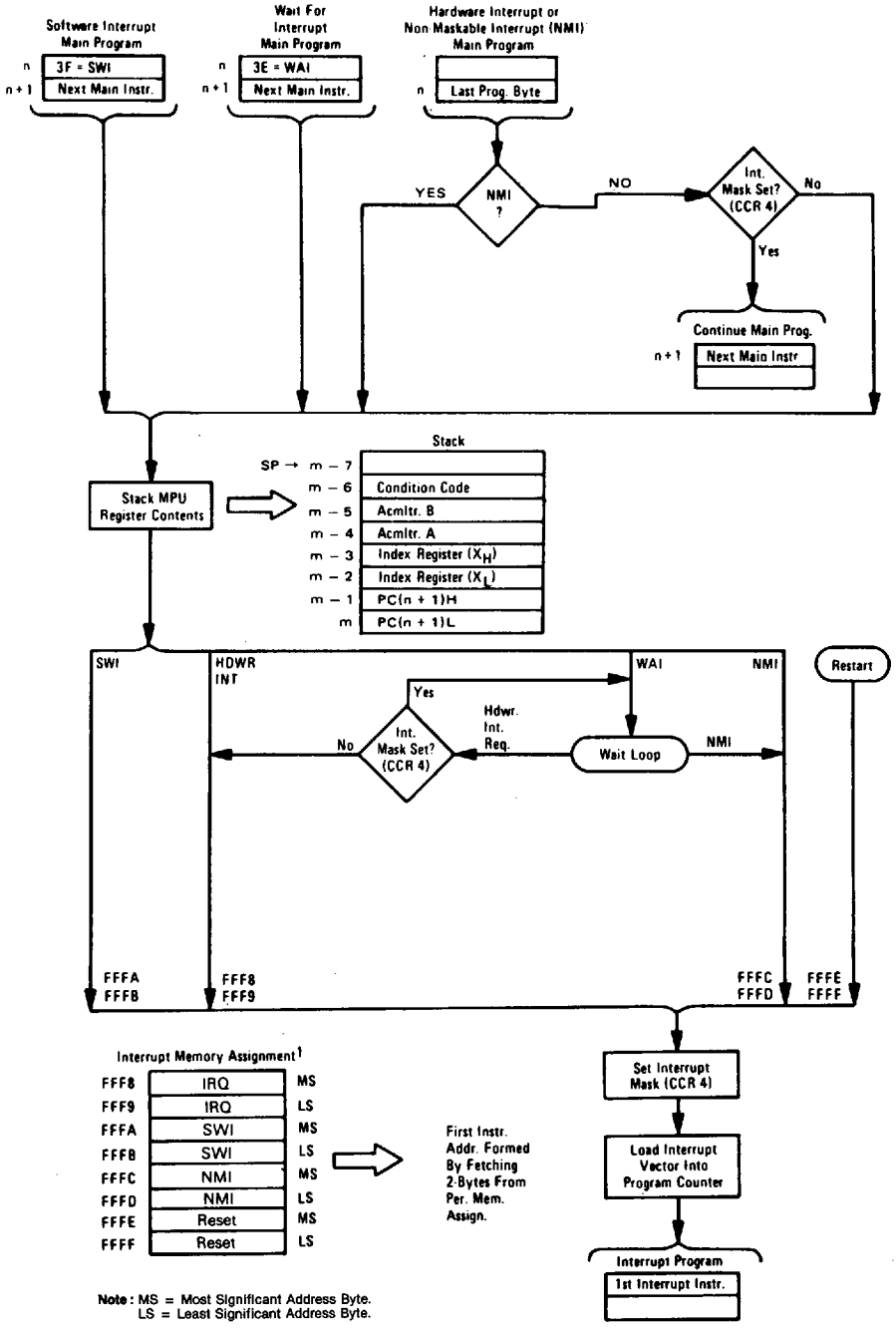


Figure 23: Program flow for interrupts.

BMI :	N = 1 ;	BEQ :	Z = 1 ;
BPL :	N = ϕ ;	BNE :	Z = ϕ ;
BVC :	V = ϕ ;	BCC :	C = ϕ ;
BVS :	V = 1 ;	BCS :	C = 1 ;
BHI :	C + Z = ϕ ;	BLT :	N⊕V = 1 ;
BLS :	C + Z = 1 ;	BGE :	N⊕V = ϕ ;
BLE :	Z + (N⊕V) = 1 ;		
BGT :	Z + (N⊕V) = ϕ ;		

Figure 24 : Conditional branch instructions.

The conditional branch instructions, Figure 24, consists of seven pairs of complementary instructions. They are used to test the results of the preceding operations and either continue with the next instruction in sequence (test fails) or cause a branch to another point in the program (test succeeds).

Four of the pairs are used for simple tests of statuts bits N, Z, V, and C :

1. Branch on Minus (BMI) and Branch On Plus (BPL) tests the sign bit, N, to determine if the previous result was negative or positive, respectively.

2. Branch On Equal (BEQ) and Branch On Not Equal (BNE) are used to test zero statuts bit, Z, to determine whether or not the result of the previous operation was equal to zero. These two instructions are useful following a Compare (CMP) instruction to test for equality between an accumulator and the operand. They are also used following the Bit Test (BIT) to determine whether or not whe same bit positions are in an accumulator and operand.

3. Branch On Overflow Clear (BVC) and Branch On Overflow Set (BVS) tests the state of the V bit to determine if the previous operation caused an arithmetic overflow.

4. Branch On Carry Clear (BCC) and Branch On Carry Set (BCS) tests the state of the C bit to determine if the previous operation caused a carry to occur. BCC and BCS are useful for testing relative magnitude when the values being tested are regarded as unsigned binary numbers, that is, the values are in the range 00 (lowest) to FF (highest). BCC following a comparison (CMP) will cause a branch if the (unsigned) value in the accumulator is higher than or the same as the value of the operand. Conversely, BCS will cause a branch if the accumulator value is lower than the operand.

The fifth complementary pair, Branch On Higher (BHI) and Branch On Lower or Same (BLS) are, in a sense, complements to BCC and BCS. BHI tests for both C and Z = 0 ; if used following a CMP, it will cause a branch if the value in the accumulator is higher than the operand. Conversely, BLS will cause a branch if the unsigned binary value in the accumulator is lower than or the same as the operand.

The remaining two pairs are useful in testing results of operations in which the values are regarded as signed two's complement numbers. This differs from the unsigned binary case in the following sense : in unsigned, the orientation is higher or lower ; in signed two's complement, the comparison is between Jarger or smaller where the range of values is between - 128 and + 127.

Branch On Less Than Zero (BLT) and Branch On Greater Than Or Equal Zero (BGE) test the status bits for N + V = 1 and N + V = 0, respectively. BTL will always cause a branch following an operation in which two negative numbers were added. In addition, it will cause a branch following a CMP in which the value in the accumulator was negative and the operand was positive. BLT will never cause a branch following a CMP in which the accumulator value was positive and the operand negative. BGE, the complement to BLT, will cause a branch following operations in which two positive values were added or in the result was zero.

The last pair, Branch On Less Than Or Equal Zero (BLE) and Branch On Greater Than Zero (BGT) test the status bits for Z + (N + V) = 1 and Z + (N + V) = 0, respectively. The action of BLE is identical to that for BLT except that a branch will also occur if the result of the previous result was zero. Conversely, BGT is similar to BGE except that no branch will occur following a zero result.

6.5 - Condition code register operations

The Condition Code Register (CCR) is a 6-bit register within the MPU that is useful in controlling program flow during system operation. The bits are defined in Figure 25.

The instructions shown in Table 10 are available to the user for direct manipulation of the CCR.

A CLI-WAI instruction sequence operated properly, with early EF 68000 processors, only if the preceding instruction was odd (Least Significant Bit = 1). Similary it was advisabler to precede any SEI instruction with an odd opcode - such as NOP. These precautions are not necessary for EF 6800 processor indicating manufacture in November 1977 or later.

Systems which require an interrupt window to be opened under program control should use a CLI-NOP-SEI sequence rather than CLI-SEI.

b_5	b_4	b_3	b_2	b_1	b_0
H	I	N	Z	V	C

H = Half-carry; set whenever a carry from b_3 to b_4 of the result is generated by ADD, ABA, ADC; cleared if no b_3 to b_4 carry; not affected by other instructions.

I = Interrupt Mask; set by hardware or software interrupt or SEI instruction; cleared by CLI instruction. (Normally not used in arithmetic operations.) Restored to a zero as a result of an RTI instruction if I_m stored on the stacked is low.

N = Negative; set if high order bit (b_7) of result is set; cleared otherwise.

Z = Zero; set if result = 0; cleared otherwise.

V = Overlow; set if there was arithmetic overflow as a result of the operation; cleared otherwise.

C = Carry; set if there was a carry from the most significant bit (b_7) of the result; cleared otherwise.

Figure 25 : Condition code register bit definition.

5

Table 10 - Condition code register instructions.

		IMPLIED				COND. CODE REG.					
						5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	=	BOOLEAN OPERATION	H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0→C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0→I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0→V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1→C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1→I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1→V	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	A → CCR	———	———	———	①	———	———
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

R = Reset
S = Set
• = Not affected
① (ALL) Set according to the contents of Accumulator A.

6.6 - Addressing modes

The MPU operates on 8-bit binary numbers presented to it via the Data Bus. A given number (byte) may represent either data or an instruction to be executed, depending on where it is encountered in the control program. The 6800 has 72 unique instructions, however, it recognizes and takes action on 197 of the 256 possibilitis that can occur using an 8-bit word length. This larger number of instructions results from the fact that many of the executive instructions have more than one addressing mode.

These addressing modes refer to the manner in which the program causes the MPU to obtain its instructions and data. Program causes the MPU to obtain its instructions and data. The programmer must have a method for addressing the MPU's internal register and all of the external memory locations.

Selection of the desired addressing mode is made by the user as the source statements are written. Translation into appropriate opcode then depends on the method used. If manual translation is used, the addressing mode is inherent in the opcode. For example, the Immediate, Direct, Indexed, and Extended modes may all be used with the ADD instruction. The proper mode is determined by selecting (hexadecimal notation) 8B, 9B, AB, or BB, respectively.

The source statement format includes adequate information for the selection if an assembler program is used to generate the opcode. For instance, the Immediate mode is selected by the assembler whenever it encounters the« = » symbol in the operand field. Similarly, an «X» in the operand field causes the indexed mode to be selected. Only the relative mode applies to the branch instructions, therefore, the mnemonic instruction itself is enough for the assembler to determine addressing mode.

For the instructions that use both direct and extended modes, the assembler selects the direct mode if the operand value is in the range 0-255 and extended otherwise. There are a number of instructions for which the extended mode is valid but the direct is not. For these instructions, the assembler automatically selects the extended mode even if the operand is in the 0-255 range. The addressing modes are summarized in Figure 26.

Inherent (Includes «Accumulator Addressing» Mode)

The successive fields in a statement are normally separated by one or more spaces. An exception to this rule occurs for instructions that use dual addressing in the operand field and for instructions that must distinguish between the two accumulators. In these cases, A and B are «operands» but the space between them and the operator may be omitted. This is commonly done, resulting in apparent four character mnemonics for those instructions.

The addition instruction, ADD, provides an example of dual addressing in the operand field:

Operator	Operand	Comment
ADDA	MEM12	ADD CONTENTS OF MEM12 TO ACCA
or		
ADDB	MEM12	ADD CONTENTS OF MEM12 TO ACCB

The example used earlier for the text instruction, TST, also applies to the accumulators and uses the «accumulator addressing mode» to designate which of the two accumulators is being tested:

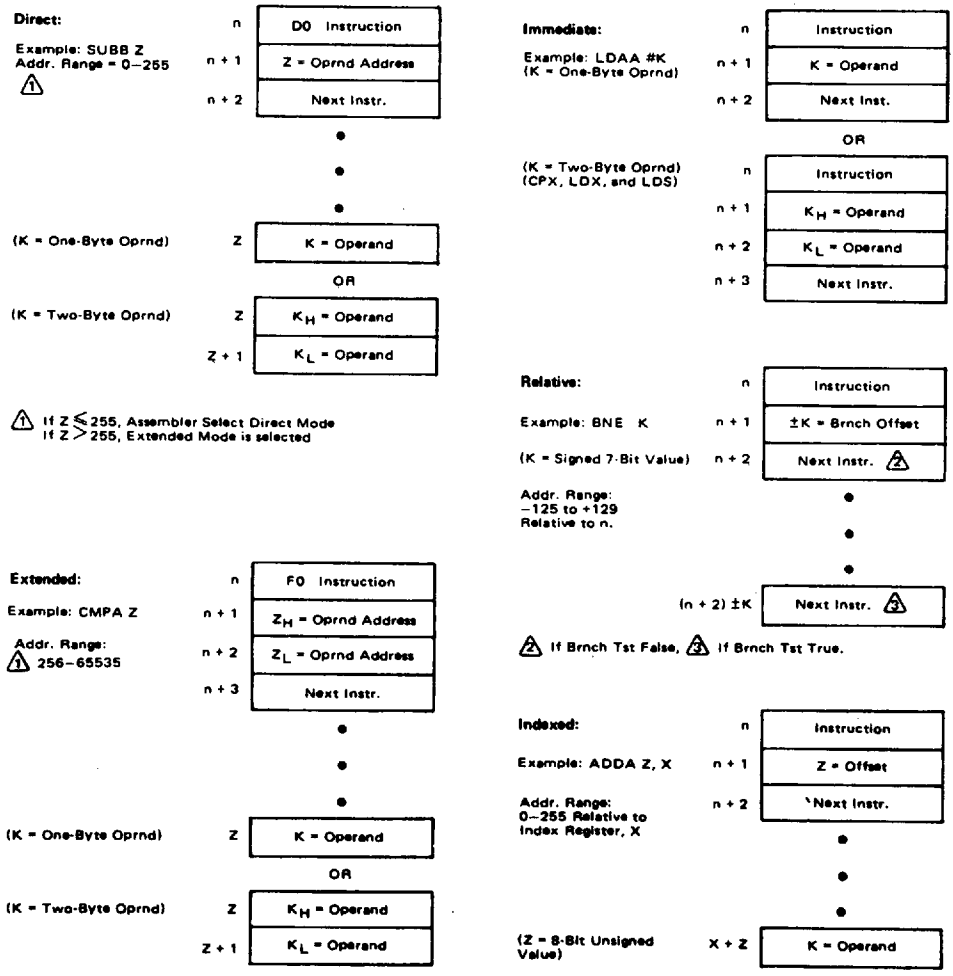


Figure 26: Addressing mode summary.

Operator	Comment
TSTB	TEST CONTENTS OF ACCB
TSTA	TEST CONTENTS OF ACCA

A number of the instructions either alone or together with an accumulator operand contain all of the address information that is required, that is «inherent» in the instruction itself. For instance, the instruction ABA causes the MPU to add the contents of accumulators A and B together and place the result in accumulator A. The instruction INCB, another example of «accumulator addressing», causes the contents of accumulator B to be increased by one. Similary, INX, increment the Index Register, causes the contents of the Index Register to by increased be one.

Program flow for instructions of this type is illustrated in Figures 27 and 28. In these figures, the general case is shown on the left and a specific example is shown on the right. Numerical examples are in decimal notation. Instructions of this type require only one byte of opcode. Cycle-by-cycle operation of the inherent mode is shown in Table 11.

Operator	Operand	Comment
STAA	X	PUT A IN INDEXED LOCATION

Immediate Addressing Mode - In the Immediate addressing mode, the operand is the value that is to be operated on. For instance, the instruction

causes the MPU to «immediately load accumulator A with the value 25» ; no further address reference is required. The Immediate mode is selected by preceding the operand value with the « = » symbol. Program flow for this addressing mode is illustrated in Figure 29.

The operand format allows either properly defined symbols or numerical values. Except for the instructions CPX, LDX, and LDS, the operand may be any value in the range 0 to 255. Since Compare Index Register (CPX), Load Index Register (LDX), and Load Stack Pointer (LDS), require 16-bit values, the immediate mode for these three instructions require two-byte operands. In the immediate addressing mode, the «address» of the operand is effectively the memory location immediately following the instruction itself. Table 12 shows the cycle-by-cycle operation for the immediate addressing mode.

Direct and Extended Addressing Modes - In the direct and extended modes of addressing, the operand field of the source statement is the address of the value that is to be operated on. The direct and extended modes differ only in the range of memory locations to which they can direct the MPU. Direct addressing generates a single 8-bit operand and, hence, can address only memory locations 0 through 255 ; a two byte operand is generated for extended addressing, enabling the MPU to reach the remaining memory locations, 256 through 65535. An example of direct addressing and its effect on program flow is illustrated in Figure 30.

The MPU, after encountering the opcode for the instruction LDAA (Direct) at memory location 5004 (Program Counter = 5004), looks in the next location, 5005, for the address of the operand. It then sets the program counter equal to the value found there (100 in the example) and fetches the operand, in this case a value to be loaded into accumulator A from that location. For instruction requiring a two-byte operand such as LDX (Load the Index Register), the operand bytes would be retrieved from locations 100 and 101. Table 13 shows the cycle-by-cycle operation for the direct mode of addressing.

Extended addressing, Figure 31, is similar except that a two-byte address is obtained from locations 5007 and 5008 after the LDAB (Extended) opcode shows up in location 5006. Extended addressing can be throught of as the «standart» addressing mode, that is, it is a method of reaching any place in memory. Direct addressing, since only one address byte is required, provides a faster method of processing data and generates fewer bytes of control code. In most applications, the direct addressing range, memory locations 0-255, are reserved for RAM. They are used for data buffering and temporary storage of system variables, the area in which faster addressing is of most value. Cycle-by-cycle operation is shown in Table 14 for Extended Addressing.

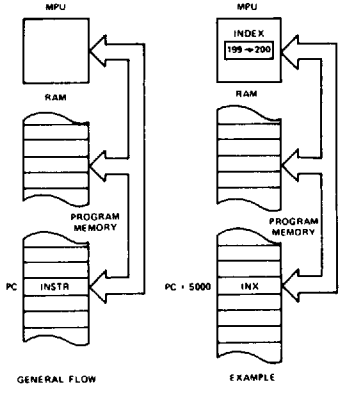


Figure 27 : Inherent addressing.

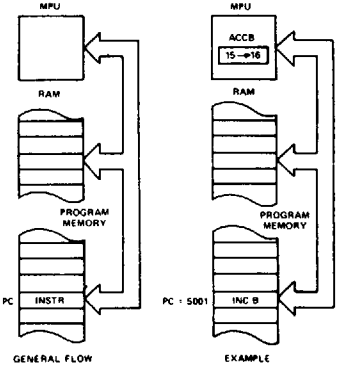


Figure 28 : Accumulator addressing.

Relative Address Mode - In both the direct and extended modes, the address obtained by the MPU is an absolute numerical address. The Relative addressing mode, implemented for the MPU's branch instructions, specifies a memory location relative to the Program Counter's current location. Branch instructions generate two bytes of machine code, one for the instruction opcode and one for the «relative» address (see Figure 32). Since it is desirable to be able to branch in either direction, the 8-bit address byte is interpreted as a signed 7-bit value ; the 8th bit of the operand is treated as a sign bit, «0» = plus and «1» = minus. The remaining seven bits represent the numerical value. This results in a relative addressing range of ± 127 with respect to the location of the branch instruction itself. However, the branch range is computed with respect to the next instruction that would be executed if the branch conditions are not satisfied. Since two bytes are generated, the next instruction is located at PC + 2. If D is defined as the address of the branch destination, the range is then :

$$(PC + 2) - 127 \leqslant D \Delta (PC + 2) + 127$$

or

$$PC - 125 \leqslant D \leqslant PC + 129$$

that is, the destination of the branch instruction must be within − 125 to + 129 memory locations of the branch instruction itself. For transferring control beyond this range, the unconditional jump (JMP), jump to subroutine (JSR), and return from subroutine (RTS) are used.

In Figure 32, when the MPU encounters the opcode for BEQ (Branch if result of last instruction was zero), it tests the Zero bit in the Condition Code Register. If that bit is «0», indicating a non-zero result, the MPU continues execution with the next instruction (in location 5010 in Figure 32). If the previous result was zero, the branch condition is satisfied and the MPU adds the offset, 15 in this case, to PC + 2 and branches to location 5025 for the next instruction.

The branch instructions allow the programmer to efficiently direct the MPU to one point or another in the control program depending on the outcome of test results. Since the control program is normally in read-only memory and cannot be changed, the relative address used in execution of branch instructions is a constant numerial value. Cycle-by-cycle operation is shown in Table 15 for relative addressing.

Indexed Addressing Mode - With Indexed addressing, the numerical address is variable and depends on the current contents of the Index Register. A source statement such as

Operator	Operand	Comment
STAA	X	PUT A IN INDEXED LOCATION

causes the MPU to store the contents of accumulator A in the memory location specified by the contents of the index register

Table 11 - Inherent mode cycle-by-cycle operation (continued)

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
DES DEX INS INX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Previous Register Contents	1	Irrelevant Data (Note 1)
		4	0	New Register Contents	1	Irrelevant Data (Note 1)
PSH	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Accumulator Data
		4	0	Stack Pointer − 1	1	Accumulator Data
PUL	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Operand Data from Stack
TSX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	0	New Index Register	1	Irrelevant Data (Note 1)
TXS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Index Register	1	Irrelevant Data
		4	0	New Stack Pointer	1	Irrelevant Data
RTS	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)
		5	1	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)

Table 11 · Inherent mode cycle-by-cycle operation

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
WAI	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6 (Note 3)	1	Contents of Cond. Code Register
RTI	10	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 2)
		3	0	Stack Pointer	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer + 1	1	Contents of Cond. Code Register from Stack
		5	1	Stack Pointer + 2	1	Contents of Accumulator B from Stack
		6	1	Stack Pointer + 3	1	Contents of Accumulator A from Stack
		7	1	Stack Pointer + 4	1	Index Register from Stack (High Order Byte)
		8	1	Stack Pointer + 5	1	Index Register from Stack (Low Order Byte)
		9	1	Stack Pointer + 6	1	Next Instruction Address from Stack (High Order Byte)
		10	1	Stack Pointer + 7	1	Next Instruction Address from Stack (Low Order Byte)
SWI	12	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (Note 1)
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6	0	Contents of Cond. Code Register
		10	0	Stack Pointer – 7	1	Irrelevant Data (Note 1)
		11	1	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
		12	1	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)

Note 1 : If device which is addressed during this cycle uses VMA, then the data bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the data bus.

Note 2 : Data is ignored by the MPU.

Note 3 : While the MPU is waiting for the interrupt, bus available will go high indicating the following states of the control lines : VMA is low ; address bus ; R/$\overline{W}$, and data bus are all in the high impedance state.

(recall that the label «X» is reserved to designate the index register). Since there are instructions for manipulating X during program execution (LDX, INX, DEC, etc.), the indexed addressing mode provides a dynamic «on the fly» way to modify program activity.

The operand field can also contain a numerical value that will be automatically added to X during execution. This format is illustrated in Figure 33.

When the MPU encounters the LDAB (indexed) opcode in location 5006, it looks in the next memory location for the value to be added to X (5 in the example) and calculates the required address by adding 5 to the present index register value of 400. In the operand format, the offset may be represented by a label or an numerical value in the range 0-255 as in the example. In the earlier example, STAA X, the operand is equivalent to 0, X, that is, the 0 may be omitted when the desired address is equal to X. Table 16 shows the cycle-by-cycle operation for the indexed mode of addressing.

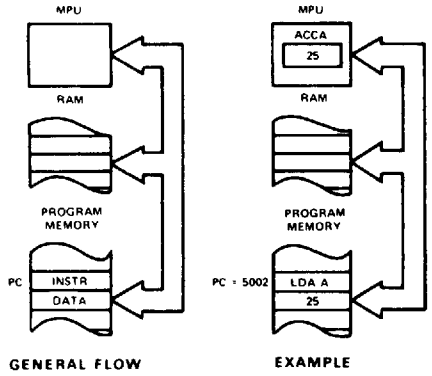


Figure 29 : Immediate addressing mode.

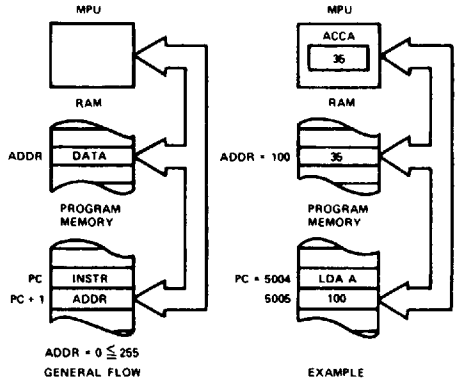


Figure 30 : Direct addressing mode.

Table 12 · Immediate mode cycle-by cycle operation

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data
CPX LDS LDX	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data (High Order Byte)
		3	1	Op Code Address + 2	1	Operand Data (Low Order Byte)

Table 13 · Direct mode cycle-by-cycle operation

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data
CPX LDS LDX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data (High Order Byte)
		4	1	Operand Address + 1	1	Operand Data (Low Order Byte)
STA	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address
		3	0	Destination Address	1	Irrelevant Data (Note 1)
		4	1	Destination Address	0	Data from Accumulator
STS STX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	0	Address of Operand	1	Irrelevant Data (Note 1)
		4	1	Address of Operand	0	Register Data (High Order Byte)
		5	1	Address of Operand + 1	0	Register Data (Low Order Byte)

Note 1 : If device which is address during this cycle uses VMA, then the data bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the data bus.

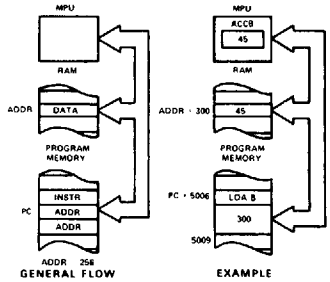


Figure 31 : Extended addressing mode.

Table 14 · Extended mode cycle-by-cycle

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
STS STX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	0	Address of Operand	1	Irrelevant Data (Note 1)
		5	1	Address of Operand	0	Operand Data (High Order Byte)
		6	1	Address of Operand + 1	0	Operand Data (Low Order Byte)
JSR	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	1	Subroutine Starting Address	1	Op Code of Next Instruction
		5	1	Stack Pointer	0	Return Address (Low Order Byte)
		6	1	Stack Pointer − 1	0	Return Address (High Order Byte)
		7	0	Stack Pointer − 2	1	Irrelevant Data (Note 1)
		8	0	Op Code Address + 2	1	Irrelevant Data (Note 1)
		9	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
JMP	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	1	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data
CPX LDS LDX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data (High Order Byte)
		5	1	Address of Operand + 1	1	Operand Data (Low Order Byte)
STA A STA B	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	1	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	0	Operand Destination Address	1	Irrelevant Data (Note 1)
		5	1	Operand Destination Address	0	Data from Accumulator
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Current Operand Data
		5	0	Address of Operand	1	Irrelevant Data (Note 1)
		6	1/0 (Note 2)	Address of Operand	0	New Operand Data (Note 2)

Note 1 : If device which is address during this cycle uses VMA, then the data bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the data bus.

Note 2 : For TST, VMA = 0 and operand data does not change.

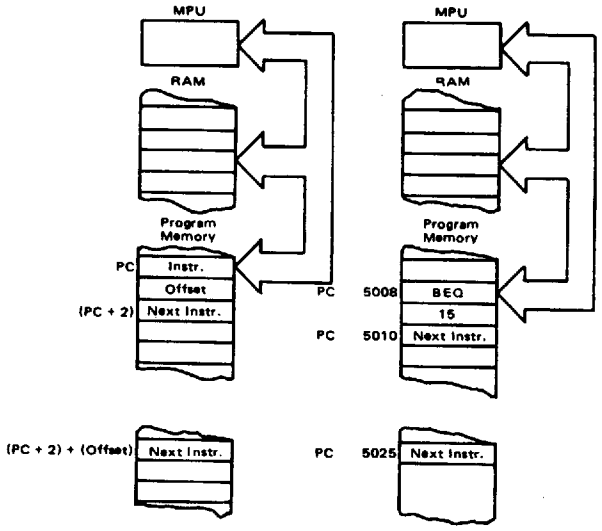


Figure 32: Relative addressing mode.

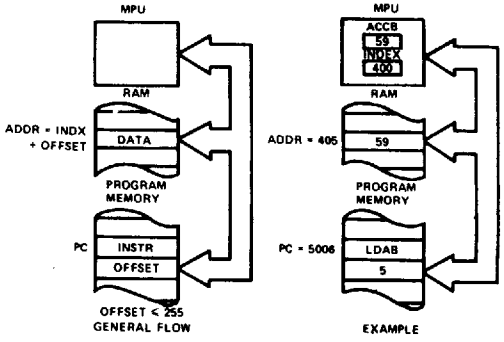


Figure 33: Indexed addressing mode.

Table 15 · Relative mode cycle-by-cycle operation

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
BCC BHI BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMI BVS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Op Code Address + 2	1	Irrelevant Data (Note 1)
		4	0	Branch Address	1	Irrelevant Data (Note 1)
BSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		7	0	Return Address of Main Program	1	Irrelevant Data (Note 1)
		8	0	Subroutine Address	1	Irrelevant Data (Note 1)

Note 1: If device which is address during this cycle uses VMA, then the data bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the data bus.

Table 16 - Indexed mode cycle-by-cycle

Address Mode and Instructions	Cycles	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
INDEXED						
JMP	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Operand Data
CPX LDS LDX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Operand Data (High Order Byte)
		6	1	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STA	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	1	Index Register Plus Offset	1	Current Operand Data
		6	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		7	1/0 (Note 2)	Index Register Plus Offset	0	New Operand Data (Note 2)
STS STX	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (Note 1)
		6	1	Index Register Plus Offset	0	Operand Data (High Order Byte)
		7	1	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
JSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (Note 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer – 2	1	Irrelevant Data (Note 1)
		7	0	Index Register	1	Irrelevant Data (Note 1)
		8	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (Note 1)

Note 1 : If device which is address during this cycle uses VMA, then the data bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the data bus.

Note 2 : For TST, VMA = 0 and operand data does not change.

7 - PREPARATION FOR DELIVERY

7.1 - Packaging

Microcircuit are prepared for delivery in accordance with MIL-M-38510 or CECC 90000.

7.2 - Certificate of compliance

TMS offers a certificate of compliance with each shipment of parts, affirming the products are in compliance either with MIL-STD-883 or TMS standards and guarantying the parameters are tested at extreme temperatures for the entire temperature range.

8 - HANDLING

MOS device must be handled with certain precautions to avoid damage due to accumulation of static charge. Input protection devices have been designed in the chip to minimize the effect of this static buildup. However, the following handling practices are recommended :

a) Device should be handled on benches with conductive and grounded surface.

b) Ground test equipement, tools and operator.

c) Do not handle devices by the leads.

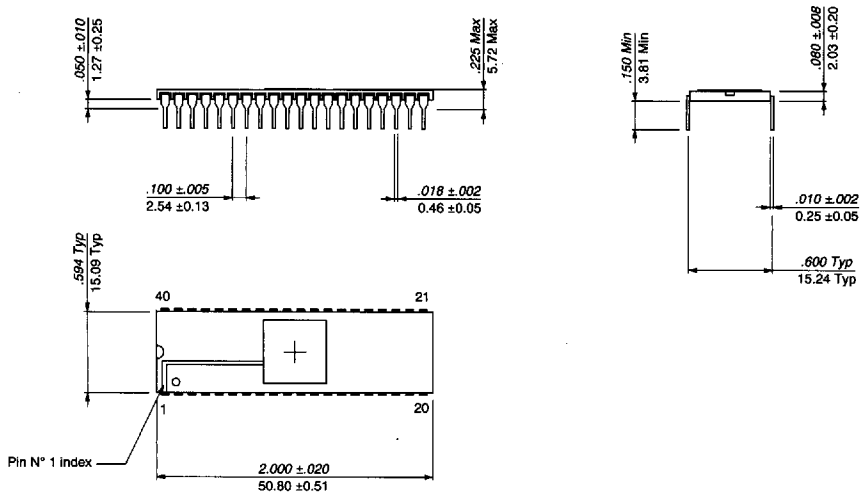
d) Store devices in conductive foam or carriers.

e) Avoid use of plastic, rubber, or silk in MOS areas.

f) Maintain relative humidity above 50 %, if pratical.

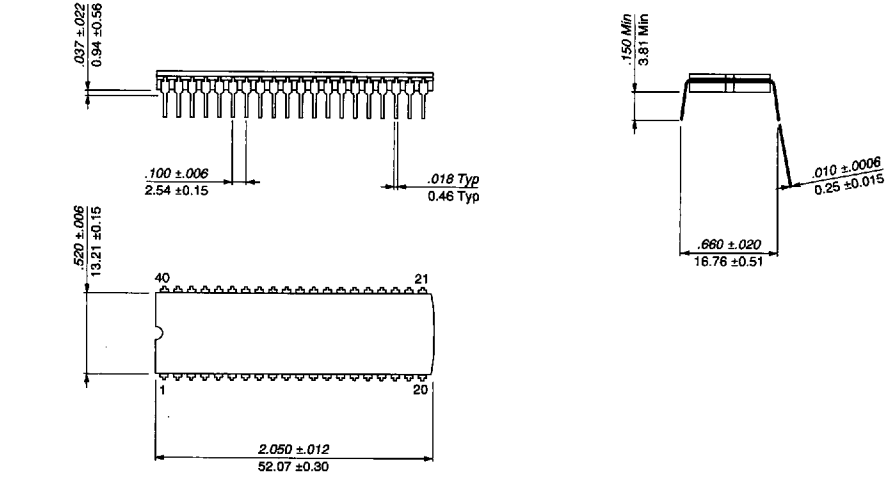
9 - PACKAGE MECHANICAL DATA

9.1 - 40 pins - DIL Ceramic Side Brazed package

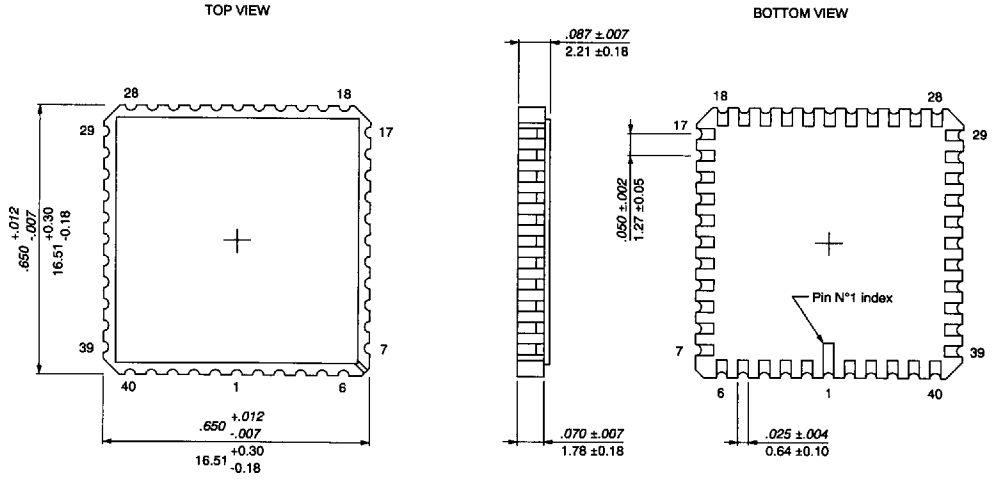


5

9.2 - 40 pins - DIL Cerdip package

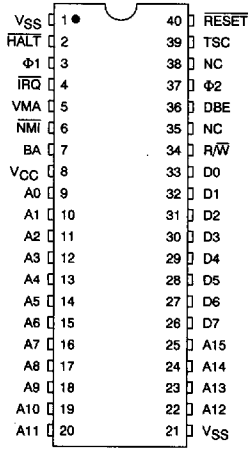


9.3 - 44 pins - Leadless Ceramic Chip carrier

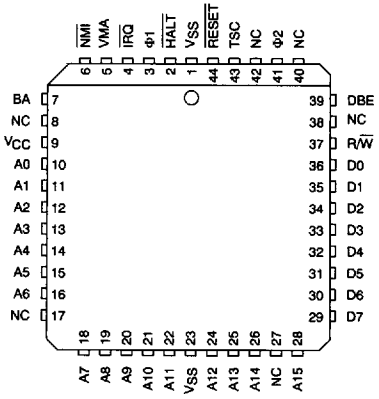


10 - TERMINAL CONNECTIONS

10.1 - DIL 40 - Pin assignment



10.2 - LCCC 44 - Pin assignment



11 - ORDERING INFORMATION

11.1 - Hi-REL product

Commercial TMS Part-Number (see Note)	Norms	Package	Temperature range T_C (°C)	Frequency (MHz)	Drawing number
EF6000JMG/B	NFC 96883 - Class G	DIL Cerdip	−55 / +125	1	Data-sheet
EF6000CMG/B*	NFC 96883 - Class G	DIL Side Brazed	−55 / +125	1	Data-sheet
EF6000EMG/B	NFC 96883 - Class G	LCCC	−55 / +125	1	Data-sheet
EF68A00JMG/B	NFC 96883 - Class G	DIL Cerdip	−55 / +125	1.5	Data-sheet
EF68A00CMG/B*	NFC 96883 - Class G	DIL Side Brazed	−55 / +125	1.5	Data-sheet
EF68A00EMG/B	NFC 96883 - Class G	LCCC	−55 / +125	1.5	Data-sheet
EF6000JMB/B	MIL STD 883 - Class B	DIL Cerdip	−55 / +125	1	Data-sheet
EF6000C1MB/B*	MIL STD 883 - Class B	DIL Side Brazed	−55 / +125	1	Data-sheet
EF68A00JMB/B	MIL STD 883 - Class B	DIL Cerdip	−55 / +125	1.5	Data-sheet
EF68A00C1MB/B*	MIL STD 883 - Class B	DIL Side Brazed	−55 / +125	1.5	Data-sheet

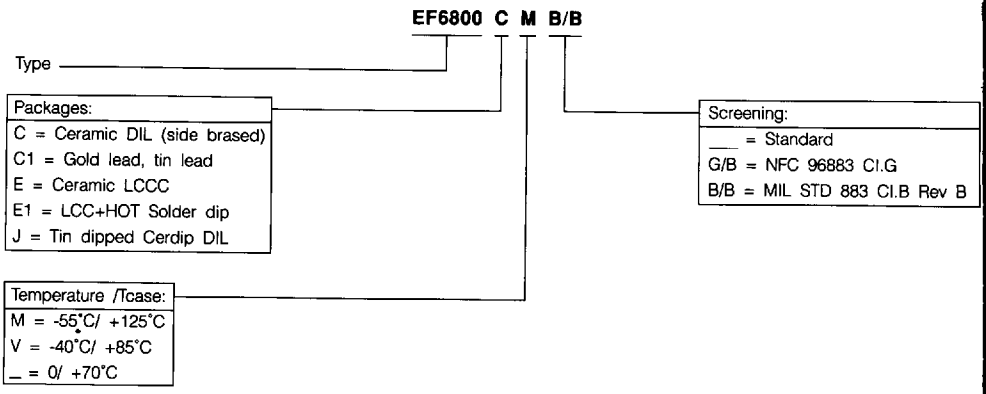
* DIL side brazed is not considered any more as standard package, Cerdip is preferred.
Note : THOMSON COMPOSANTS MILITAIRES ET SPATIAUX.

11.2 - Standard product

Commercial TMS Part-Number (see Note)	Norms	Package	Temperature range T_C (°C)	Frequency (MHz)	Drawing number
EF6800CV*	TMS standard	DIL Side Brazed	−40 / +85	1	Data sheet
EF6800JV	TMS standard	Cerdip DIL	−40 / +85	1	Data sheet
EF68A00CV*	TMS standard	DIL Side Brazed	−40 / +85	1.5	Data sheet
EF68A00JV	TMS standard	Cerdip DIL	−40 / +85	1.5	Data sheet
EF6800JM	TMS standard	Cerdip DIL	−55 / +125	1	Data sheet
EF6800CM*	TMS standard	DIL Side Brazed	−55 / +125	1	Data sheet
EF6800EM	TMS standard	LCCC	−55 / +125	1	Data sheet
EF68A00JM	TMS standard	Cerdip DIL	−55 / +125	1.5	Data sheet
EF68A00CM*	TMS standard	DIL Side Brazed	−55 / +125	1.5	Data sheet
EF68A00EM	TMS standard	LCCC	−55 / +125	1.5	Data sheet
EF6800C*	TMS standard	DIL Side Brazed	0 / +70	1	Data sheet
EF6800J	TMS standard	Cerdip DIL	0 / +70	1	Data sheet
EF68A00C*	TMS standard	DIL Side Brazed	0 / +70	1.5	Data sheet
EF68A00J	TMS standard	Cerdip DIL	0 / +70	1.5	Data sheet
EF68B00J	TMS standard	Cerdip DIL	0 / +70	2	Data sheet

* DIL side brazed is not considered any more as standard package, Cerdip is preferred.
Note : THOMSON COMPOSANTS MILITAIRES ET SPATIAUX.

5


EF6800 C M B/B
Type
Packages:
C = Ceramic DIL (side brased)
C1 = Gold lead, tin lead
E = Ceramic LCCC
E1 = LCC+HOT Solder dip
J = Tin dipped Cerdip DIL
Temperature /Tcase:
M = -55°C/ +125°C
V = -40°C/ +85°C
_ = 0/ +70°C
Screening:
___ = Standard
G/B = NFC 96883 Cl.G
B/B = MIL STD 883 Cl.B Rev B