HMOS 8-BIT MICROPROCESSOR UNIT (MPU)*

DESCRIPTION

The EF 6809 is a revolutionary high-performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the EF 6800 Family has major architectural improvements which include additional registers, instructions, and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The EF 6809 has the most complete set of addressing modes availables on any 8-bit microprocessor today.

The EF 6809 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

MAIN FEATURES

EF 6800 compatible

- Hardware - interfaces with all EF 6800 peripherals.
- Software - upward source code compatible instruction set and addressing modes.

Architectural features

- Two 16-bit index registers.
- Two 16-bit indexable stack pointers.
- Two 8-bit accumulators can be concatenated to form one 16-bit accu mulator.
- Direct page register allows direct addressing throughout memory.

Hardware features

- On-chip oscillator (crystal frequency = 4 x E).
- DMA/BREQ allows DMA operation on memory refresh.
- Fast interrupt request input stacks only condition code register and program counter.
- MRDY input extends data access times for use with slow memory.
- Interrupt acknowledge output allows vectoring by devices.
- Sync acknowledge output allows for syncronization to external event.
- Single bus-cycle RESET.
- Single 5-volt supply operation.
- NMI inhibited after RESET until after first load of stack pointer.
- Early address valid allows use with slower memories.
- Early write data for dynamic memories.

Software features

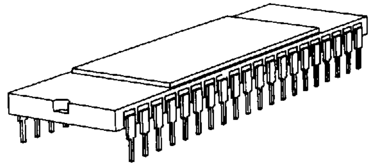
- 10 addressing modes :
  - EF 6800 upward compatible addressing modes,
  - direct addressing anywhere in memory map,
  - long relative branches,
  - program counter relative,
  - expended indexed addressing :
    - 0-, 5-, 8- or 16-bit constant offsets,
    - 8- or 16-bit accumulator offsets,
    - auto increment/decrement by 1 or 2.
- Improved stack manipulation.
- 1464 instructions with unique addressing modes.
- 8 x 8 unsigned multiply.
- 16-bit arithmetic.
- Transfer/exchange all registers.
- Push/pull any registers or any set of registers.
- Load effective address.
- Frequency of operation over full military temperature range : 1 & 1.5 MHz.
- EF68B09J (2 MHz in 0 – 70°C).

SCREENING / QUALITY

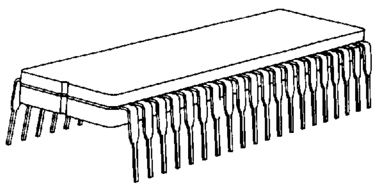
This product could be manufactured in full compliances with either :

- CECC 90000 (class B, assessment level Y) 90110-008.
- MIL-STD-883 (class B).
- or according to TMS standards.

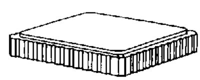
* High density, N channel silicon gate.



C suffix
DIL 40
Ceramic Side Brazed package



J suffix
DIL 40
Ceramic Cerdip package



E suffix
LCCC 44
Ceramic Leadless Chip Carrier

See the ordering information page 50.

Pin connection : see page 49.

SUMMARY

A - GENERAL DESCRIPTION

1 - EF 6809 EXPANDED BLOCK DIAGRAM

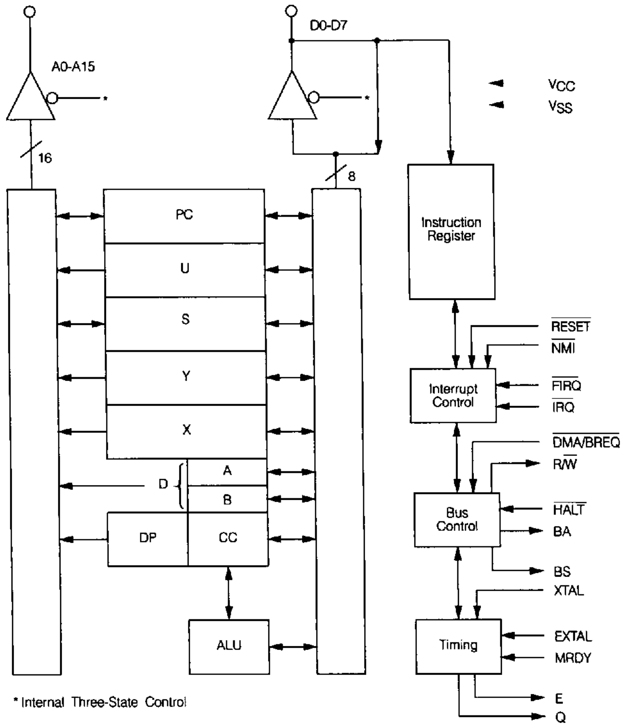


Figure 1 : EF 6809 expanded block diagram.

2 - SIGNAL DESCRIPTION

POWER (V_{SS}, V_{CC})

Two pins are used to supply power to the part : V_{SS} is ground or 0 volts while V_{CC} is +5.0 V ±5%.

ADDRESS BUS (A0 A15)

Sixteen pins are used to output address information from the MPU onto the address bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{W}$ = 1, and BS = 0 ; this is a «dummy access» or $\overline{VMA}$ cycle. Addresses are valid on the rising edge of Q. All address bus drivers are made high impedance when output bus available (BA) is high. Each pin will drive one Schottky TTL load or four LSTTL loads, and 90 pF.

DATA BUS (D0-D7)

These eight pins provide communication with the system bidirectional data bus. Each pin will drive one Schottky TTL load or four LSTTL loads, and 130 pF.

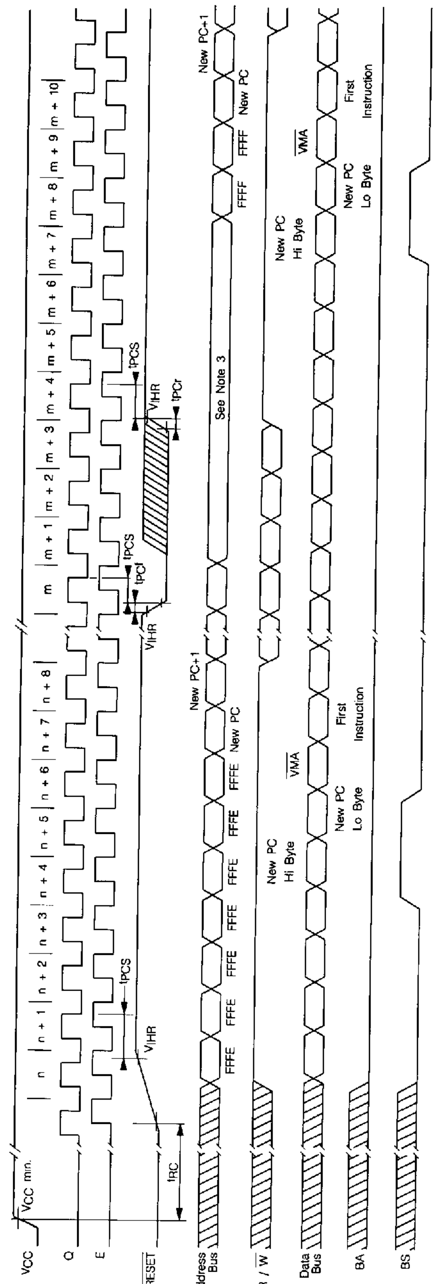
READ/WRITE (R/$\overline{W}$)

This signal indicates the direction of data transfer on the data bus. A low indicates that the MPU is writing data onto the data bus. R/$\overline{W}$ is made high impedance when BA is high. R/$\overline{W}$ is valid on the rising edge of Q.

$\overline{RESET}$

A low level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 2. The reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (Table 1) when interrupt acknowledge is true, ($\overline{BA}$ • BS = 1). During initial power on, the $\overline{RESET}$ line should be held low until the clock oscillator is fully operatinal.

Because the EF 6809 $\overline{RESET}$ pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the processor.

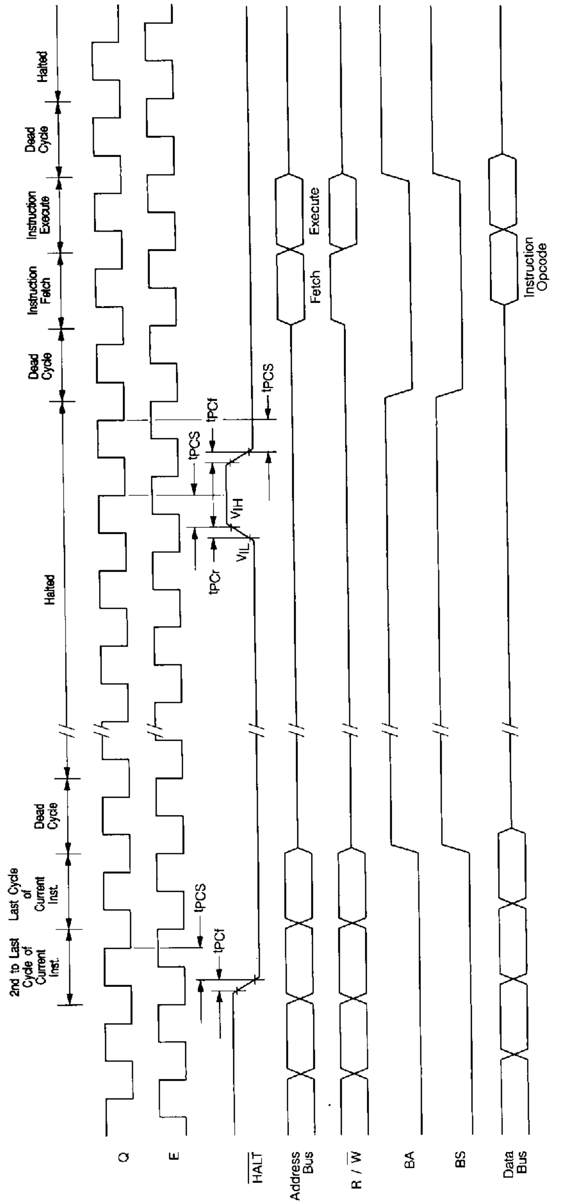


Note 1 : Parts with date codes prefixed by 7F or 5A will come out of $\overline{RESET}$ one cycle sooner than shown.

Note 2 : Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

Note 3 : FFFE appears on the bus during $\overline{RESET}$ low time. Following the active transition of the $\overline{RESET}$ line, three more FFFE cycles will appear followed by the vector fetch.

Figure 2 : $\overline{RESET}$ timing



Note : Waveform measurements for all inputs and outputs are specified at logic high 2.0 V and logic low 0.8 V unless otherwise specified.

Figure 3 : $\overline{HALT}$ and single instruction execution for system debug.

5

HALT

A low level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven high indicating the buses are high impedance. BS is alos high which indicates the processor is in the halt or bus grant state. While halted, the MPU will not respond to external real-time requests (FIRQ, IRQ) although DMA/BREQ will always be accepted, and NMI or RESET will be latched for later response. During the halt state, Q and E continue to run normally. If the MPU is not running (RESET, DMA/BREQ), a halted state (BA • BS = 1) can be achieved by pulling HALT low while RESET is still low. If DMA/BREQ and HALT are both pulled low, the processor will reach the last cycle of the instruction (by reverse cycle stealing) where the machine will the become halted. See Figure 3.

BUS AVAILABLE, BUS STATUS (BA, BS)

The bus available output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. This signal does not imply that the bus will be available for more than one cycle. When BA goes low, a dead cycle will elapse before the MPU acquires the bus.

The bus status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

MPU State		MPU State Definition
BA	BS	
0	0	Normal (running)
0	1	Interrupt or reset acknowledge
1	0	Sync acknowledge
1	1	Halt or bus grant acknowledge

INTERRUPT ACKNOWLEDGE is indicated during both cycles of a hardware-vector-fetch (RESET, NMI, FIRQ, IRQ, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

Table 1 - Memory map for interrupt vectors

Memory map for vector locations		Interrupt vector description
MS	LS	
FFFE	FFFF	RESET
FFFC	FFFD	NMI
FFFA	FFFB	SWI
FFF8	FFF9	IRQ
FFF6	FFF7	FIRQ
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

SYNC ACKNOWLEDGE is indicated while the MPU is waiting for external synchronization on an interrupt line.

HALT/BUS GRANT is true when the MC 6809 is in a halt or bus grant condition.

NON MASKABLE INTERRUPT (NMI)*

A negative transition on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than FIRQ, IRQ, or software interrupts. During recognition of an NMI, the entire machine state is saved on the hardware stack. After reset, an NMI will not be recognized until the first program load of the hardware stack pointer (S). The pulse width of NMI low must be at least one E cycle. If the NMI input does not meet the minimum set up with respect to Q, the interrupt will not the recognized until the next cycle. See Figure 4.

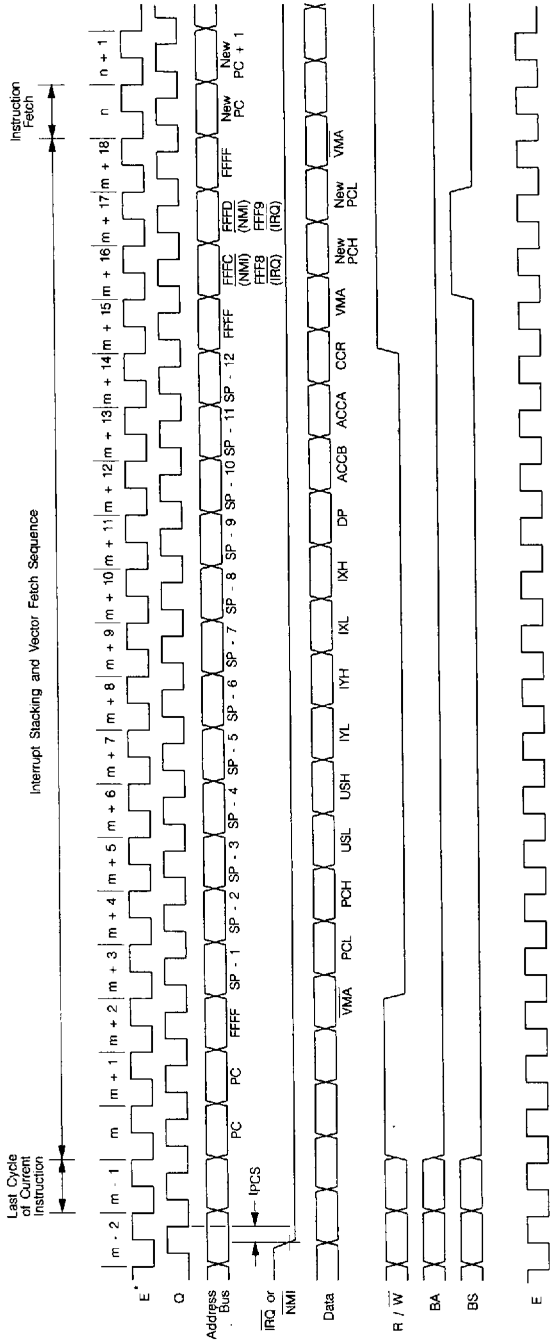
FAST-INTERRUPT REQUEST (FIRQ)*

A low level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard interrupt request (IRQ), and is fast in the sense that it stacks onlu the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 5.

INTERRUPT REQUEST (IRQ)*

A low level input on this pin will initiate an interrupt request sequence provided the mask bit (I) in the CC is clear. Since IRQ stacks the entire machine stae it provides a slower response to interrupts than FIRQ. IRQ also has a lower priority than FIRQ. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 4.

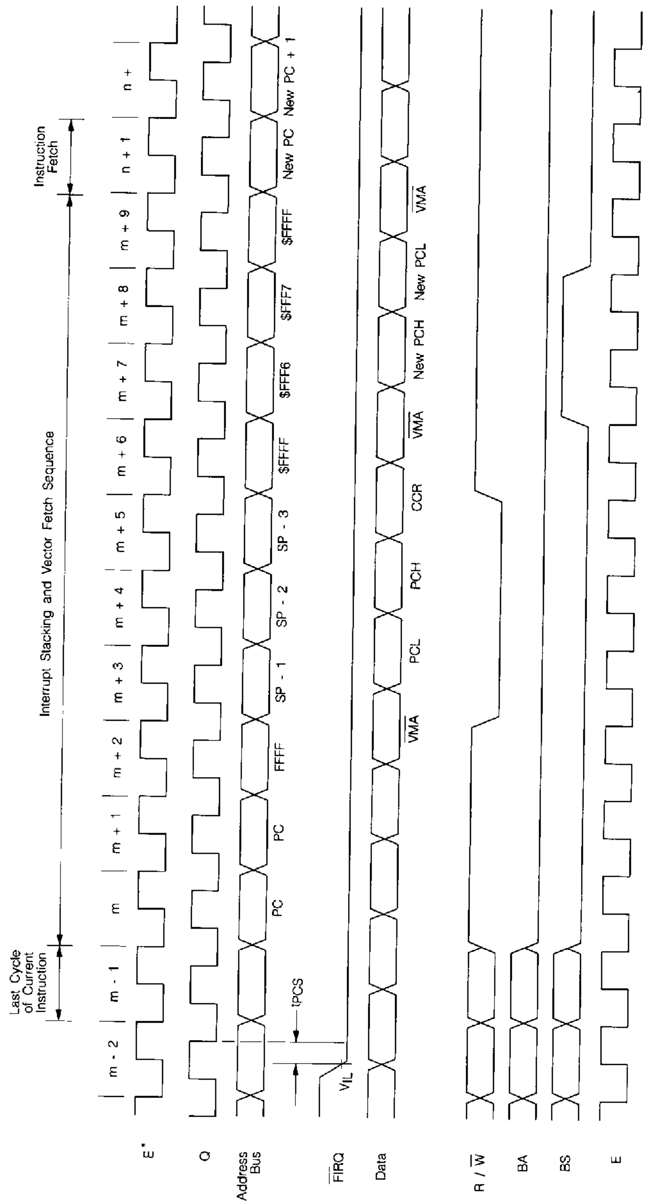
* NMI, FIRQ, and IRQ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion to the current instruction unless a SYNC or CWAI condition is present. If IRQ and FIRQ do not remain low until completion of the current instruction they may not be recognized. Howerver, NMI is latched and need only remain low for one cycle. No interrupts are recognized or latched between the falling edge of RESET and the rising edge of BS indicating RESET acknowledge.



Note : Waveform measurements for all inputs and outputs are specified at logic high = 2.0 V and logic low = 0.8 V unless otherwise specified.

* E clock shown for reference only.

Figure 4 : $\overline{IRQ}$ and $\overline{NMI}$ interrupt timing.

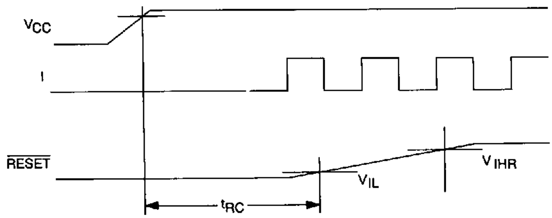


Note : Waveform measurements for all inputs and outputs are specified at logic high = 2.0 V and logic low = 0.8 V unless otherwise specified.
* E clock shown for reference only.

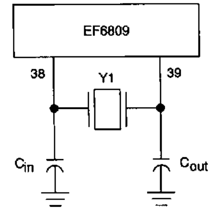
Figure 5 : $\overline{FIRQ}$ interrupt timing.

XTAL, EXTAL

These inputs are used to connect the on-chip oscillator to an external parallel-resonant crystal. Alternately, the pin EXTAL may be used as a TTL level input for external timing by grounding XTAL. The crystal or external frequency is four times the bus frequency. See Figure 6. Proper RF layout techniques should be observed in the layout of printed circuit boards.



Note : Waveform measurements for all inputs and outputs are specified at logic high = 2.0 V and logic low = 0.8 V unless otherwise specified.



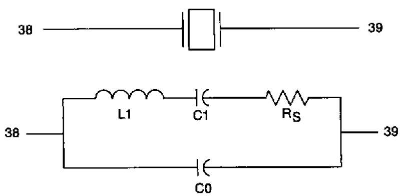
Y1	C_{in}	C_{out}
8 MHz	18 pF	18 pF
6 MHz	20 pF	20 pF
4 MHz	24 pF	24 pF

Nominal crystal parameters

	3.58 MHz	4.00 MHz	6.0 MHz	8.0 MHz
R_S	60 Ω	50 Ω	30-50 Ω	20-40 Ω
C0	3.5 pF	6.5 pF	4-6 pF	4-6 pF
C1	0.015 pF	0.025 pF	0.01-0.02 pF	0.01-0.02 pF
Q	> 40 k	> 30 k	> 20 k	> 20 k

All parameters are 10 %

Note : These are representative AT-cut crystal parameters only. Crystals of other types of cut may also be used.



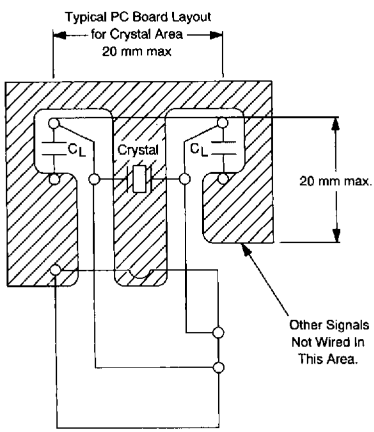
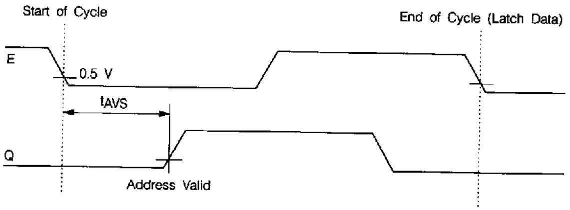


Figure 6 : Crystal connections and oscillator start up.

E, Q

E is similar to the EF 6800 bus timing signal phase 2 ; Q is a quadrature clock signal which leads E. Q has no parrallel on the EF 6800. Addresses from the MPU will be valid with the leading edge of Q. Data is latched on the falling edge of E. Timing for E and Q is shown in Figure 7.



Note : Waveform measurements for all inputs and outputs are specified at logic high = 2.0 V and logic low = 0.8 V unless otherwise specified.

Figure 7 : E/Q Relationship.

MRDY

The on-board generator furnishes E and Q to both the system and the MPU. When MRDY is pulled low, both the system clocks and the internal MPU clocks are stretched. Assetion of $\overline{\text{DMA/BREQ}}$ input stops the internal MPU clocks while allowing the external system clocks to RUN (i.e., release the bus to a DMA controller). The internal MPU clocks resume operation after $\overline{\text{DMA/BREQ}}$ is released or after 16 bus cycles (14 DMA, two dead), whichever occurs first. While $\overline{\text{DMA/BREQ}}$ is asserted it is sometimes necessary to pull MRDY low to allow DMA to/from slow memory/peripherals. As both MRDY and $\overline{\text{DMA/BREQ}}$ control the internal MPU clocks, care must be exercised not to violate the maximum t_{cyc} specification for MRDY or $\overline{\text{DMA/BREQ}}$. (Maximum t_{cyc} during MRDT or $\overline{\text{DMA/BREQ}}$ is 16 μs).

This input control signal allows stretching of E and Q to extend data-access time. E and Q operate normally while MDRY is high. When MRDY is low, E and Q may be stretched in integral multiples of quarter (1/4) bus cycles, thus allowing interface to slow memories, as shown in Figure 8 (a). During non-valid memory access ($\overline{\text{VMA}}$ cycles), MRDY has no effect on stretching E and Q ; this inhibits slowing the processor during «don't care» bus accesses. MRDY may also be used to stretch clocks (for slow memory) when bus control has been transferred to an external device (through the use of $\overline{\text{HALT}}$ and $\overline{\text{DMA/BREQ}}$).

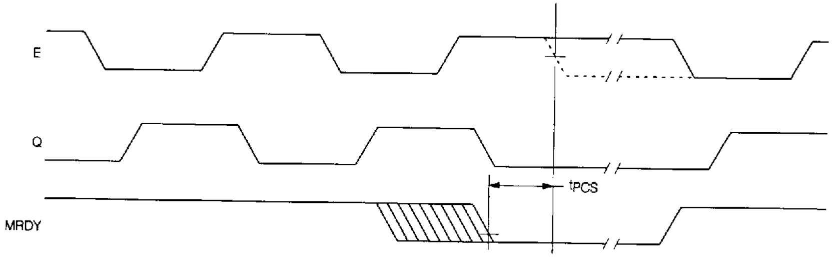


Figure 8a : MRDY Timing.

$\overline{\text{DMA/BREQ}}$

The $\overline{\text{DMA/BREQ}}$ input provides a method of suspending execution and acquiring the MPU bus for another use, as shown in Figure 9. Typical uses include DMA and dynamic memory refresh.

A low level on this pin will stop instruction execution at the end of the current cycle unless pre-empted bu self-refresh. The MPU will acknowledge $\overline{\text{DMA/BREQ}}$ by setting BA and BS to a one. The requesting device will now have up to 15 bus cycles before the MPU retrieves the bus for self-refresh. Self-refresh requires one bus cycles with a leading and trailing dead cycle. See Figure 10. The self-refresh counter is only cleared if $\overline{\text{DMA/BREQ}}$ is inactive for two or more MPU cycles.

Typically, the DMA controller will request to use the bus by asserting $\overline{\text{DMA/BREQ}}$ pin low on the leading edge of E. When the MPU replies by setting BA and BS to a one, that cycle will be a dead cycle used to transfer bus mastership to the DMA controller.

False memory accesses may be prevented during any dead cycles by developing a system $\overline{\text{DMAVMA}}$ signal which is LOW in any cycle when BA has changed.

When BA goes low (either as a result of $\overline{\text{DMA/BREQ}}$ = HIGH or MPU self-refresh), the DMA device should be taken off the bus. Another dead cycle will elapse before the MPU accesses memory to allow transfer of bus mastership without contention.

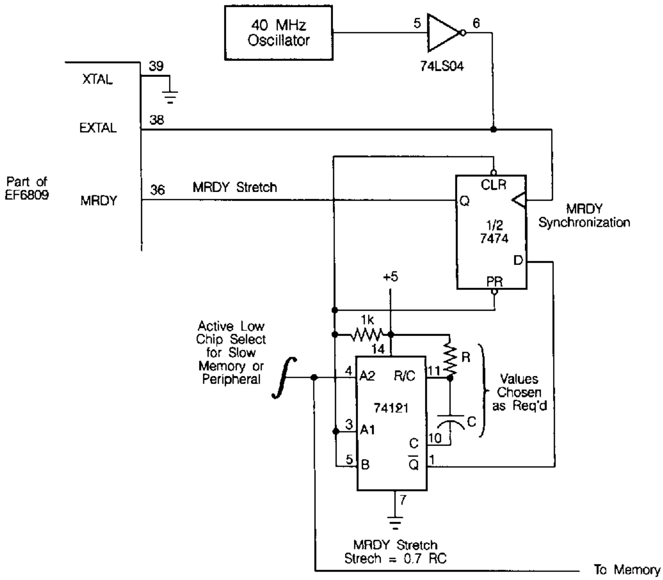


Figure 8b : Synchronization.

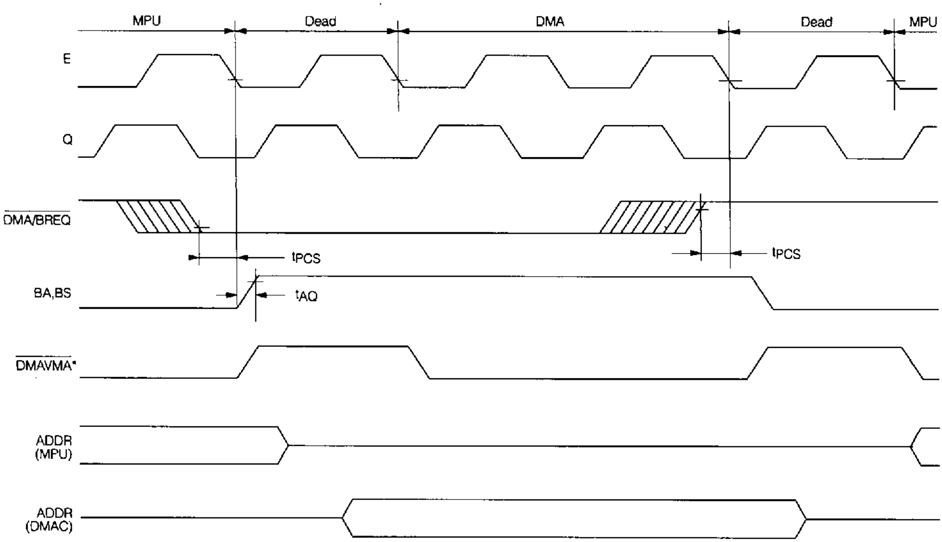
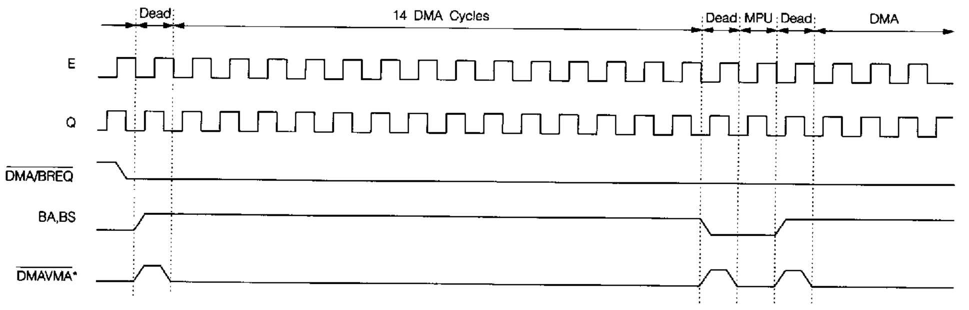


Figure 9 : Typical DMA timing (< 14 cycles).



* DMAVMA is a signal which is developed externally, but is a system requirement for DMA.

Note : Waveform measurements for all inputs and outputs are specified at logic high = 2.0 V and logic low = 0.8 V unless otherwise specified.

Figure 10 : Auto-refresh DMA timing (> 14 cycles) (reverse cycle stealing).

B · DETAILED SPECIFICATIONS

1 · SCOPE

This drawing describes the specific requirements for the microprocessor EF 6809, 1 and 1.5 MHz, in compliance either with MIL-STD-883 class B or CECC 90000, the 2 MHz version is available in 0-70°C range only.

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 as shown on § 10.1 and § 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 hermetically sealed ceramic packages which are conform to case outlines of MIL-M-38510 appendix C (when defined) :

- 40 leads DIP (for ceramic and cerdip packages)
- 44 terminals SQ. LCC (for leadless chip carrier package)

The precise case outlines are described on § 9.

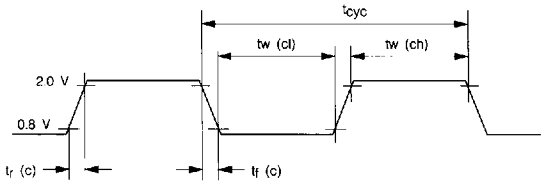
3.3 - Electrical characteristics

*3.3.1 - **Absolute maximum ratings*** *(see Table 2)*

Table 2

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		1.1	W
			T_{case} = +25°C		0.8	W
T_{case}	Operating temperature	M suffix EF 6809/EF 68A09	f = 1 and 1.5 MHz	−55	+125	°C
		V suffix EF 6809/EF 68A09	f = 1 and 1.5 MHz	−40	+85	°C
		No suffix EF 6809/EF 68A09 EF 68B09	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

Note : Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range start outside, and pass through, the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.



This device contains protective circuitry against damage due to high static voltages or electrical fields ; however, it is advises that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

3.4 - Thermal characteristics (at 25°C)

Table 3

Package	Symbol	Parameter	Value	Unit
40 ceramic DIL side brazed C suffix	θ JA	Thermal resistance - Ceramic junction to ambient	50	°C/W
	θ JC	Thermal resistance - Ceramic junction to case	10	°C/W
Cerdip 40 J suffix	θ JA	Thermal resistance - Ceramic junction to ambient	60	°C/W
	θ JC	Thermal resistance - Ceramic junction to case	10	°C/W
LCCC 44 E suffix	θ JA	Thermal resistance - Ceramic junction to ambient	50	°C/W
	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 :

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

TA = 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 reliationship 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 either MIL-STD-883 for class B devices or CECC 90000 devices.

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 - DESC / 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.

4.2 - CECC

Is in accordance with CECC 90000. Group A and B inspection are performed on each production lot as specified in CECC 9011 0-008. Group C inspection is performed on a periodic basis in accordance with CECC 90110-008.

5 - ELECTRICAL CHARACTERISTICS

5.1 - General requirements

All static and dynamic electrical characteristics are specified for inspection purpose, refer to relevant specification.

Table 4 : Static electrical characteristics for all electrical variants. See § 5.2.

Table 5 : Dynamic electrical characteristics. See § 5.3.

For static characteristics, test methods refer to IEC 748-2 method number, where existing.

For dynamic characteristics (Table 5), test methods refer to clause 5.4 hereafter of this specification.

5.2 · Static characteristics

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

Table 4

Symbol	Characteristic		Min	Typ	Max	Unit
V_{IH} V_{IHR}	Input high voltage	Logic, EXTAL $\overline{RESET}$	V_{SS} +2.0 V_{SS} +4.0		V_{CC} V_{CC}	V V
V_{IL}	Input low voltage	Logic, EXTAL, $\overline{RESET}$	V_{SS} −0.3		V_{SS} +0.8	V
I_{in}	Input leakage current (V_{in} = 0 to 5.25 V, V_{CC} = max)	Logic			2.5	µA
V_{OH}	DC output high voltage (I_{Load} = −205 µA, V_{CC} = min) (I_{Load} = −145 µA, V_{CC} = min) (I_{Load} = −100 µA, V_{CC} = min)	 D0-D7 A0-A15, R/$\overline{W}$, Q, E BA, BS	 V_{SS} +2.4 V_{SS} +2.4 V_{SS} +2.4			 V V V
V_{OL}	DC output low voltage (I_{Load} = 2.0 mA, V_{CC} = min)				V_{SS} +0.5	V
P_{INT}	Internal power dissipation (measured at T_C = −55°C in steaby state operation)				1.1	W
C_{in}	Capacitance* (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	 D0-D7, $\overline{RESET}$ Logic inputs, EXTAL, XTAL		 10 10	 15 15	 pF pF
C_{out}		A0-A15, R/$\overline{W}$, BA, BS			15	pF
f_{XTAL}	Frequency of operation (crystal or external input)	EF 6809 EF 68A09 EF 68B09	0.4 0.4 0.4		4 6 8	MHz MHz MHz
I_{TSI}	Hi-Z (off state) input current (V_{in} = 0.4 to 2.4 V, V_{CC} = max)	 D0-D7 A0-A15, R/W		 2.0	 10 10	 µA µA

* Capacitances are periodically tested rather than 100 % tested.

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 for 1 and 1.5 MHz and V_{CC} in the range 4.75 V to 5.25 V V_{IL} = 0.8 V and V_{IH} = 2 V (See also Note 1).

Table 5 · Bus timing characteristics (See Note 1)

Symbol	Ident number	Characteristic	EF 6809		EF 68A09		EF 68B09*		Unit
			Min	Max	Min	Max	Min	Max	
t_{cyc}	1	Cycle time (See Note 2)	1.0	10	0.667	10	0.5	10	µs
PW_{EL}	2	Pulse width, E low	430	5000	280	5000	210	5000	ns
PW_{EH}	3	Pulse width, E high	450	15500	280	15700	220	15700	ns
t_r, t_f	4	Clock rise and fall time		25		25		20	ns
PW_{QH}	5	Pulse width, Q high	430	5000	280	5000	210	5000	ns
PW_{QL}	6	Pulse width, Q low	450	15500	280	15700	220	15700	ns
t_{AVS}	7	Delay time, E to Q rise	200	250	130	165	80	125	ns
t_{AH}	9	Address hold time** (See Note 3)	20		20		20		ns

Table 5 - Bus timing characteristics (Continued) (See Note 1)

Symbol	Ident number	Characteristic	EF 6809		EF 68A09		EF 68B09*		Unit
			Min	Max	Min	Max	Min	Max	
t_{AQ}	10	BA, BS, R/$\overline{W}$, and address valid time to Q rise	50		25		15		ns
t_{DSR}	17	Read data setup time	80		60		40		ns
t_{DHR}	18	Read data hold time**	10		10		10		ns
t_{DDQ}	20	Data delay time from Q		200		140		110	ns
t_{DHW}	21	Write data hold time*	30		30		30		ns
t_{ACC}	29	Usable access time (See Note 4)	695		440		330		ns
t_{PCS}		Processor control setup time (MRDY, interrupts, $\overline{DMA/BREQ}$, $\overline{HALT}$, $\overline{RESET}$) (Figures 2, 3, 4, 5, 8 and 9)	200		140		110		ns
t_{RC}		Crystal oscillator start time (Figures 2 and 6)	100		100		100		ms
t_{PCr}, t_{PCf}		Processor control rise and fall Time (Figures 2 and 3)		100		100		100	ns

* For T_C : from 0 to 70°C only.
** Address and data hold times are periodically tested rather than 100% tested.
Note 1 : Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.
Note 2 : Maximum t_{cyc} during MRDY or $\overline{DMA/BREQ}$ is 16 µs.
Note 3 : Hold time (9) for BA and BS is not specified.
Note 4 : Usable access time is computed by : 1-4-7 max + 10-17.

5.4 - Test conditions specific to the device

5.4.1 - Time definitions

The times specified in Table 5 as dynamic characteristics are defined in Figure 11 below, by a reference number given the column «method» of the tables together with the relevant figure number.

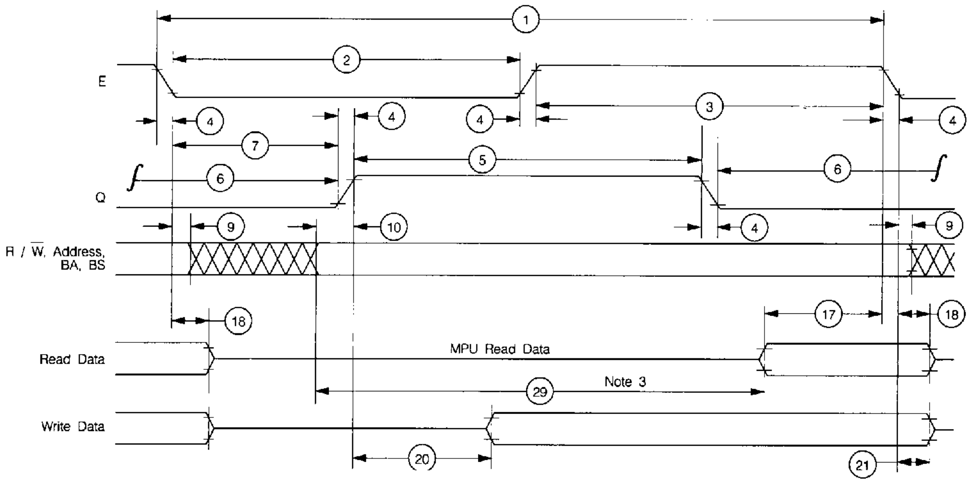
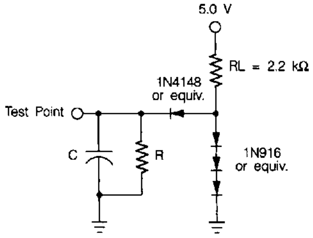


Figure 11 : Bus timing.

5.4.2 - Loading network

Figure 12 : here below shows the loading network applicable to the timing table.



C = 30 pF for BA, BS, LIC, AVMA, BUSY
130 pF for D0-D7
90 pF for A0-A15, R / $\overline{W}$

R = 11.7 kΩ for D0-D7
16.5 kΩ for A0-A15, R / $\overline{W}$
24 kΩ for BA, BS, LIC, AVMA, BUSY

Figure 12 : Bus timing test load.

5.5 - Additional information

Additional information shall not be for any inspection purposes.

***5.5.1 - Power considerations** (See § 3.4)*

***5.5.2 - Capacitance (Not for inspection purposes)** see § 5.2 static characteristic table*

6 - FUNCTIONNAL DESCRIPTION

6.1 - Programming model

As shown in Figure 13, the EF 6809 adds three registers to the set available in the EF 6800. The added registers include a direct page register, the user stack pointer, and a second index register.

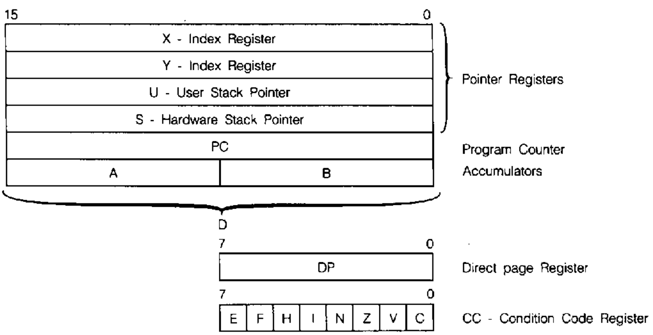


Figure 13 : Programming model of the microprocessing unit.

ACCUMULATORS (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register, and is formed with the A register as the most significant byte.

DIRECT PAGE REGISTER (DP)

The direct page register of the EF 6809 serves to enhance the direct addressing mode. The content fo this register appears at the higher address outputs (A8-A15) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure EF 6800 compatibility, all bits of this register are cleared during processor reset.

INDEX REGISTERS (X, Y)

The index registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

STACK POINTER (U, S)

The hardware stack pointer (S) is used automatically by the processor during subroutine calls and interrupts. The stack pointers of the EF 6809 point to the top of the stack, in contrast ot he EF 6800 stack pointer, which pointed to the next free location on the stack. Ther user stack pointer (U) is controlled exclusively by the programmer. This allows arguments to be passed to and from subroutines with ease Both stack pointers have the same indexed mode addressing capabilities as the X and Y register, but also support Push and Pull instructions. This allows the EF 6809 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

PROGRAM COUNTER

The program counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative addressing is provided allowing the program counter to be used like an index register in some situations.

CONDITION CODE REGISTER

The condition code register defines the state of the processor at any given time. See Figure 14

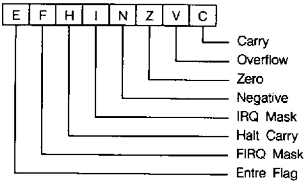


Figure 14 : Condition code register format.

CONDITION CODE REGISTER DESCRIPTION

BIT 0 (C)

Bit 0 is the carry flag, and is usually the carry from the binary ALU C is also used to represent a «borrow» from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

BIT 1 (V)

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not math the carry from the MSB 1.

BIT 2 (Z)

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

BIT 3 (N)

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

BIT 4 (I)

Bit 4 is the IRQ mask bit. The processor will not recognize interrupts from the IRQ line if this bit is set to a one. NMI, FIRQ, IRQ, RESET, and SWI all set I to a one, SWI2 and SWI3 do not affect I.

BIT 5 (H)

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-likė instruction.

BIT 6 (F)

Bit 6 is the FIRQ mask bit. The processor will not recognize interrupts from the FIRQ line if this bit is a one. NMI, FIRQ, SWI and RESET all set F to a one. RIQ, SWI2, and SWI3 do not affect F.

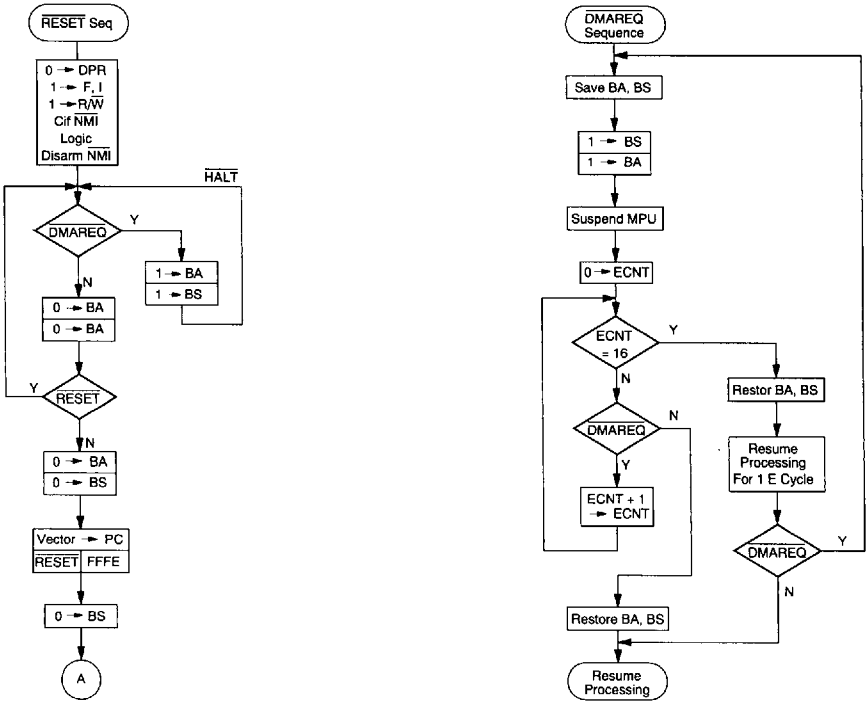
BIT 7 (E)

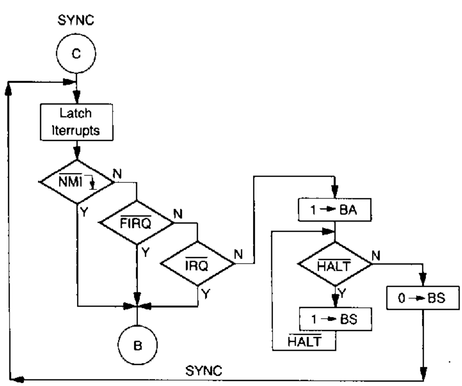
Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the condition code register represents past action.

6.2 - MPU operation

During normal operation, the MPU fetches an instruction from memory and then executes the requested function.

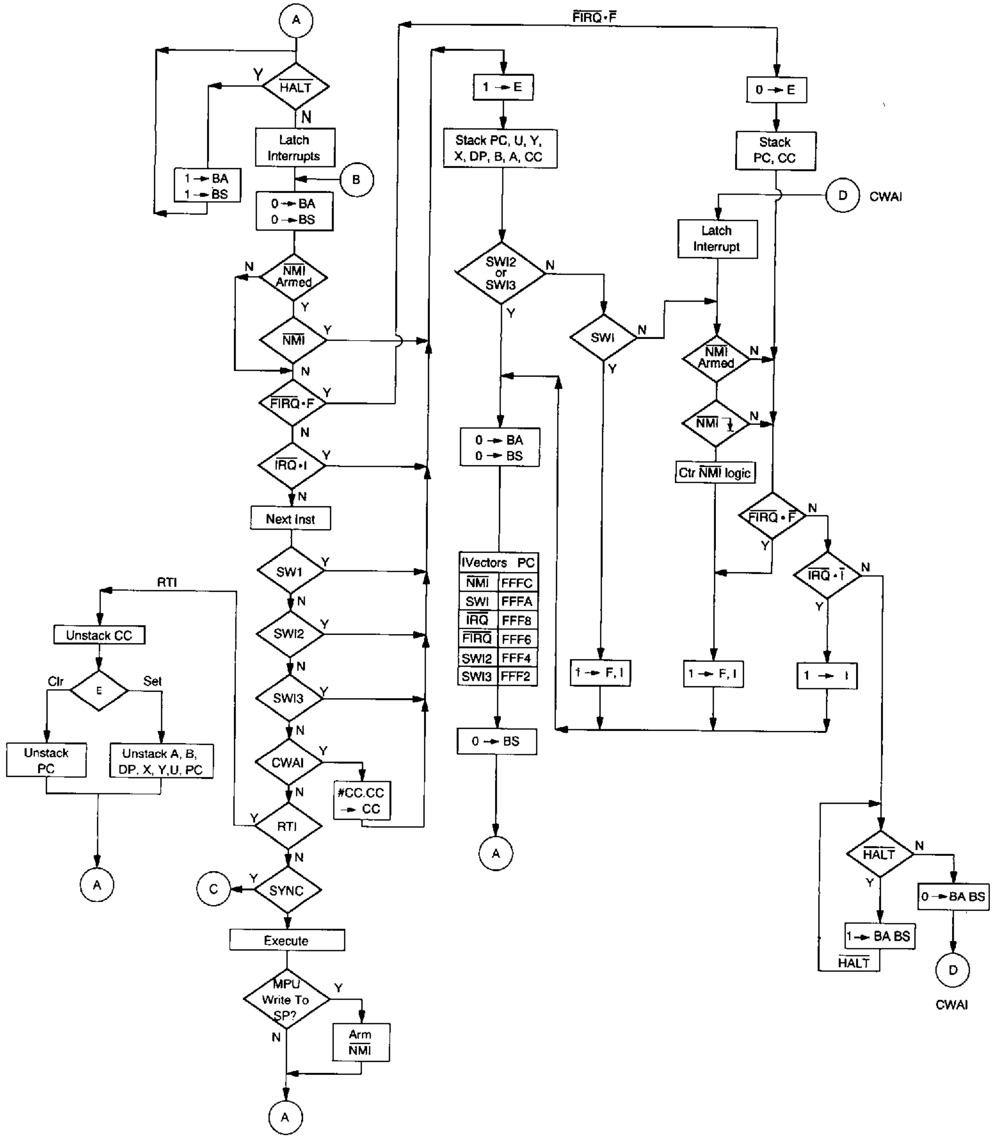
This sequence begins after $\overline{\text{RESET}}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI, and SYNC. An interrupt, $\overline{\text{HALT}}$, or $\overline{\text{DMA/BREQ}}$ can also alter the normal execution of instructions. Figure 15 illustrates the flowchart for the EF 6809.





Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync Acknowledge	1	0
Halt or Bus Grant Acknowledge	1	1

Figure 15: Flowchart for EF 6809 instructions.



Note : Asserting $\overline{\text{RESET}}$ will result in entering the reset sequence from any point in the flowchart.

Figure 15 : Flowchart for EF 6809 instructions (continued).

6.3 - Addressing modes

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The EF 6809 has the most complete set of addressing modes available on any microcomputer today. For example, the EF 6809 has 59 basic instructions ; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the EF 6809 :

- Inherent (includes accumulator)
- Immediate
- Extended
    - Extended direct
- Direct
- Register
- Indexed
    - Zero-Offset
    - Constant Offset
    - Accumulator Offset
    - Auto Increment/Decrement
    - Indexed Indirect
- Relative
    - Short/Long Relative Branching
    - Program Counter Relative Addressing.

INHERENT (INCLUDES ACCUMULATOR)

in this addressing mode the opcode of the instruction contains all the address information necessary. Examples of inherent addressing are : ABX, DAA, SWI, ASRA, and CLRB.

IMMEDIATE ADDRESSING

In immediate addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately following the opcode of the instruction). The EF 6809 uses both 8- and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with immediate addressing are :

```
LDA # $20
LDX # $F000
LDY # CAT
```

Note : signifies Immediate addressing ; $ signifies hexadecimal value.

EXTENDED ADDRESSING

In extended addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position idenpendent. Examples of extended addressing include :

```
LDA   CAT
STW   MOUSE
LDD   $2000
```

EXTENDED INDIRECT

As in the special case of indexed addresing (discussed below), one level of indirection may be added to extended addressing. In extended indirect, the two bytes following the postbyte of an indexed instruction contain the address of the data.

```
LDA   [CAT]
LDX   [$FFFE]
STU   [DOG]
```

DIRECT ADDRESSING

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower eight bits of the address to be used. The upper eight bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accesed without redefining the contents of the DP register. Since the DP register is set to $00 on reset, direct addressing on the EF 6809 is comptible with direct addressing on the EF 6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are :

```
LDA    $30
SETDP  $10 (assembler directive)
LDB    $1030
LDD    < CAT
```

Note : < is an assembler directive which forces direct addressing.

REGISTER ADDRESING

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addresing are :

```
TFR    X, Y,        Transfers X into Y
EXG    A, B         Exchanges A with B
PSHS   A, B, X, Y   Push Y, X, B and A onto S
PULU   X, Y, D      Pull D, X, and Y from U
```

INDEXED ADDRESSING

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 16 lists the legal formats for the postbyte. Table 6 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Postbyte Register Bit								Indexed addressing mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R + 5 bit offset
1	R	R	0	0	0	0	0	,R +
1	R	R	i	0	0	0	1	,R + +
1	R	R	0	0	0	1	0	,– R
1	R	R	i	0	0	1	1	,– – R
1	R	R	i	0	1	0	0	EA = ,R + 0 offset
1	R	R	i	0	1	0	1	EA = ,R + ACCB offset
1	R	R	i	0	1	1	0	EA = ,R + ACCA offset
1	R	R	i	1	0	0	0	EA = ,R + 8 bit offset
1	R	R	i	1	0	0	1	EA = ,R + 16 bit offset
1	R	R	i	1	0	1	1	EA = ,R + D offset
1	x	x	i	1	1	0	0	EA = ,PC + 8 bit offset
1	x	x	i	1	1	0	1	EA = ,PC + 16 bit offset
1	R	R	i	1	1	1	1	EA = [,Address]

Bits 3–0: Addressing Mode Field
Bit 4: Indirect Field (Sign bit when b_7 = 0)
Bits 6–5: Register Field : RR
00 = X
01 = Y
10 = U
11 = S

x = Don't Care
d = Offset Bit
i = 0 = Not Indirect
    1 = Indirect

Figure 16 : Indexed addressing postbyte register bit assignments.

ZERO-OFFSET INDEXED

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are :

```
LDD    O, X
LDA    S
```

CONSTANT OFFSET INDEXED

In this mode, a two's complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of ofssets are available :

5 bit (– 16 to + 15)
8 bit (– 128 to + 127)
16 bit (– 32768 to + 32767)

The two's complement 5-bit offset is inculded in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are :

```
LDA    23, X
LDX    –2, S
LDY    300, X
LDU    CAT, Y
```

Table 6 - Indexed addressing mode

Type	Forms	Non indirect				Indirect			
		Assembler form	Postbyte opcode	+~	+#	Assembler form	Postbyte opcode	+~	+#
Constant offset from R	No offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
(2s complement offsets)	5-bit offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8-bit offset	n, R	1RR0100000	1	1	[n, R]	1RR11000	4	1
	16-bit offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator offset from R (2s complement offsets)	A register offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B register offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D register offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto increment/decrement R	Increment by 1	,R+	1RR00000	2	0	not allowed			
	Increment by 2	,R+	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement by 1	,−R	1RR00010	2	0	not allowed			
	Decrement by 2	,−−R	1RR00011	3	0	[n, −−R]	1RR10011	6	0
Constant offset from PC (2s complemet ofsets)	8-bit offset	n, PCR	1xx01100	1	1	[n, PCR]	1xx11100	4	1
	16-bit offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended indirect	16-bit address					[n]	10011111	5	2

R = X, Y, U or S
x = don't care

RR:
00 = X
01 = Y
10 = U
11 = S

+~ and +# indicate the number of additionnal cycles and bytes for the particular variation.

5

ACCUMULATOR-OFFSET INDEXED

This mode is similar to constant offset indexed except that the two's complement value in one of the accumulators (A, B, or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulatro to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA    B, Y
LDX    D, Y
LEAX   B, X
```

AUTO INCREMENT/DRECREMENT INDEXED

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is drecremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc, are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8- or 16-bit data to be accessed and is selectable by the programmer. The pre-decrement, post-increment nature of these modes allows them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples, of the auto increment/decrement addressing modes are:

```
LDA    ,X +
STD    ,Y + +
LDB    ,− Y
LDX    ,− − S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction: STX 0, X + + (X initialized to 0)

The desired result is to store zero in locations \$0000 and \$0001 then increment X to point to \$0002. In reality, the following occurs:

```
0 → temp      calculate the EA ; temp is a holding register
X - 2 → X     perform auto increment
X → (temp)    do store operation
```

INDEXED INDIRECT

All of the indexing modes, with the exception of auto increment/decrement by one or a ±4-bit offset, may have an additional level of indirection specidied. In indirect addressing, the effective address is contained at the location specified by the contents of the index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated form the index register and an offset.

```
        Before Execution
        A = XX (don't care)
        X = $F000
$0100   LDA ($10, X)   EA is now $F010
$F010   $F1            $F150 is now the new EA
$F011   $50
$F150   $AA
        After Execution
        A = $AA Actual Datat Loaded
        X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by one indirect). Some examples of indexed indirect are:

```
LDA   [,X]
LDD   [10, S]
LDA   [B, Y]
LDD   [,X + +]
```

RELATIVE ADDRESSING

The byte(s) following the branch opcode is (are) treated as a signed offset which may be adged to the program counter. If the branch condition is true, then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (one byte offset) and long (two bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
        BEQ     CAT      (short)
        BGT     DOG      (short)
CAT     LBEQ    RAT      (long)
DOG     LBGT    RABBIT   (long)
        •
        •
        •
RAT     NOP
RABBIT  NOP
```

PROGRAM COUNTER RELATIVE

The PC can be used as the pointer register with 8- or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program counter relative addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the program counter. Examples are:

```
LDA    CAT, PCR
LEAX   TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA    [CAT, PCR]
LDU    [DOG, PCR]
```

6.4 - Instruction set

The instruction set of the EF 6809E is similar to that of the EF 6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

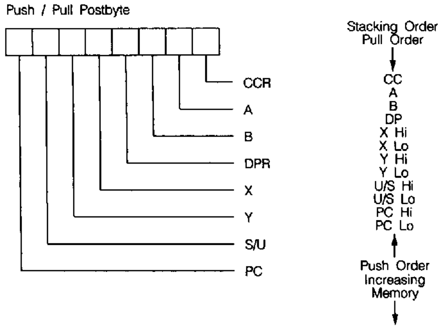
Some of the new instructions are described in detail below.

PSHU/PSHS

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register or set of registers with a single instruction

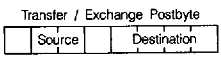
PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual push/pull sequence is fixed, each bit defines a unique register to push or pull, as shown below.



TFR/EXG

Within the EF 6809E, any register may be transferred to or exchanged with another of like size, i.e., 8 bit to 8 bit or 16 bit to 16 bit. Bits 4-7 of postby te define the source register, while bits 0-3 represent the destination regsiter. These are denoted as follows :



Register Field

0000 = D (A.B)	1000 = A
0001 = X	1001 = B
0010 = Y	1010 = CCR
0011 = U	1011 = DPR
0100 = S	
0101 = PC	

Note : All other combinations are undefined and INVALID.

LEAX/LEAY/LEAU/LEAS

The LEA (load effective address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 3.

The LEA instruction also allows the user to access data and tables in a position independent manner. For example :

```
        LEAX    MSG1, PCR
        LBSR    PDATA (print message routine)
        •
        •
MSG1    FCC     «MESSAGE»
```

This sample program prints : «MESSAGE». By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the auto increment and auto decrement addressing modes due to the sequence of internal operatins. The LEA internal sequence of internal operations. The LEA internal sequence is outlined as follows :

LEAa, b+ (any of the 16-bit pointer registers X, Y, U, or S may be substituted for a and b)
1. b → temp (calculate the EA)
2. b + 1 → b (modify b, postincrement)
3. temp → a (load a)

LEAa, – b
1. b – 1 → temp (calculate EA with predecrement)
2. b – 1 → b (modify b, predecrement)
3. temp → a (load a)

Auto increment-by-two and auto decrement-by-two instruction work similary. Note that LEAX ; X + does not change X ; however, LEAX, – X does decrement ; LEAX 1, X should be used to increment X by one.

Table 7 - LEA examples

Instruction		Operation		Comment
LEAX	10, X	X + 10	X	Adds 5-bit constant 10 to X
LEAX	500, X	X + 500	X	Adds 16-bit constant 500 to X
LEAY	A, Y	Y + A	Y	Adds 8-bit A accumulator to Y
LEAY	D, Y	Y + D	Y	Adds 16-bit D accumulator to Y
LEAU	– 10, U	U – 10	U	Substracts 10 from U
LEAS	– 10, S	S – 10	S	Used to reserve area on stack
LEAS	10, S	S + 10	S	Used to clean up'stack
LEAX	5, S	S + 10	X	Transfers as well as adds

MUL

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumlator. The unsigned multiply also allows multipleprecision multiplications.

LONG AND SHORT RELATIVE BRANCHES

The EF 6809 has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8- or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64K memory map. Position-independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

SYNC

After encountering a sync instruction, the MPU enters a sync state, stops processing instructions, and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{\text{NMI}}$) or maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) clear, the processor will clear the sync state and perform the normal interrupt stacking and service routine. Since $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) set, the processor will clear the sync state and continue processing by executing the next in-line instruction. Figure 17 depicts sync timing.

SOFTWARE INTERRUPTS

A software interrupt is an instruction which will cause an interrupt and its associated vector fetch. These software interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software developement systems. Three levels of SWI are available on the EF 6809, and are prioritized in the following order : SWI, SWI1, SWI3.

16-BIT OPERATION

The EF 6809, has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes, and pulls.

CYCLE-BY-CYCLE OPERATION

The address bus cycle-by-cycle performance chart (Figure 18) illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the EF 6809. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds through-put). Next, the operation of each opcode will follow the flowchart. $\overline{\text{VMA}}$ is an indication of $FFFF_{16}$ on the address bus, $R/\overline{W}$ = 1 and BS = 0. The following examples illustrate the use of the chart.

Example 1 : LBSR (Branch Taken). Before Execution SP = F000

```
                    •
                    •
                    •
$ 8000              LBSR     CAT
                    •
                    •
                    •
$ A000     CAT      •
```

Cycle-by-cycle flow

Cycle #	Address	Data	R/W	Description
1	8000	17	1	Opcode fetch
2	8001	20	1	Offset high byte
3	8002	00	1	Offset low byte
4	FFFF	*	1	VMA cycle
5	FFFF	*	1	VMA cycle
6	A000	*	1	Computed branch address
7	FFFF	*	1	VMA cycle
8	EFFF	80	0	Stack high order byte of return address
9	EFFE	03	0	Stack low order byte of return address

Example 2 : DEC (Extended)

$ 8000 DEC $ A000
•
•
•
$ A8000 $ 80

Cycle-by-cycle flow

Cycle #	Address	Data	R/W	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	VMA Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	VMA Cycle
7	A000	7F	0	Store the Decremented Data
* The data bus has the data at that particular address.				

5

INSTRUCTION SET TABLES

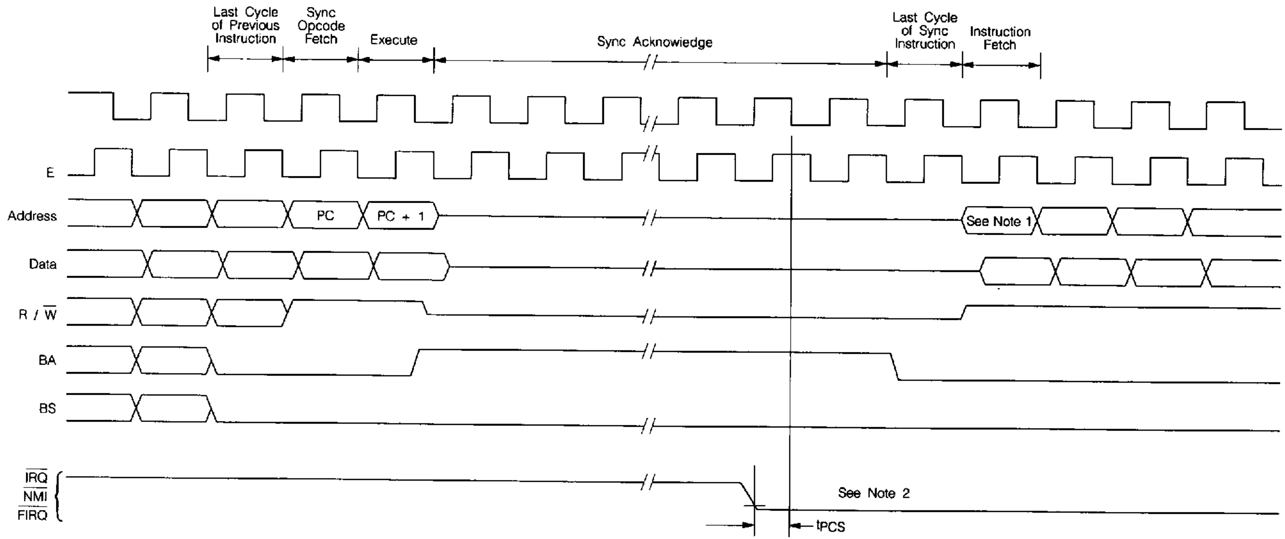
The instructions of the EF 6809 have been broken down into five different categories. They are as follows :

8-bit operation (Table 8)
16-bit operation (Table 9)
Index register/stack pointer instructions (Table 10)
Relative branches (long or short) (Table 11)
Miscellaneous instructions (Table 12)

Hexadecimal value for the instructions are given in Table 13.

PROGRAMMING AID

Figure 19 contains a compilation of data that will assist in programming the EF 6809.

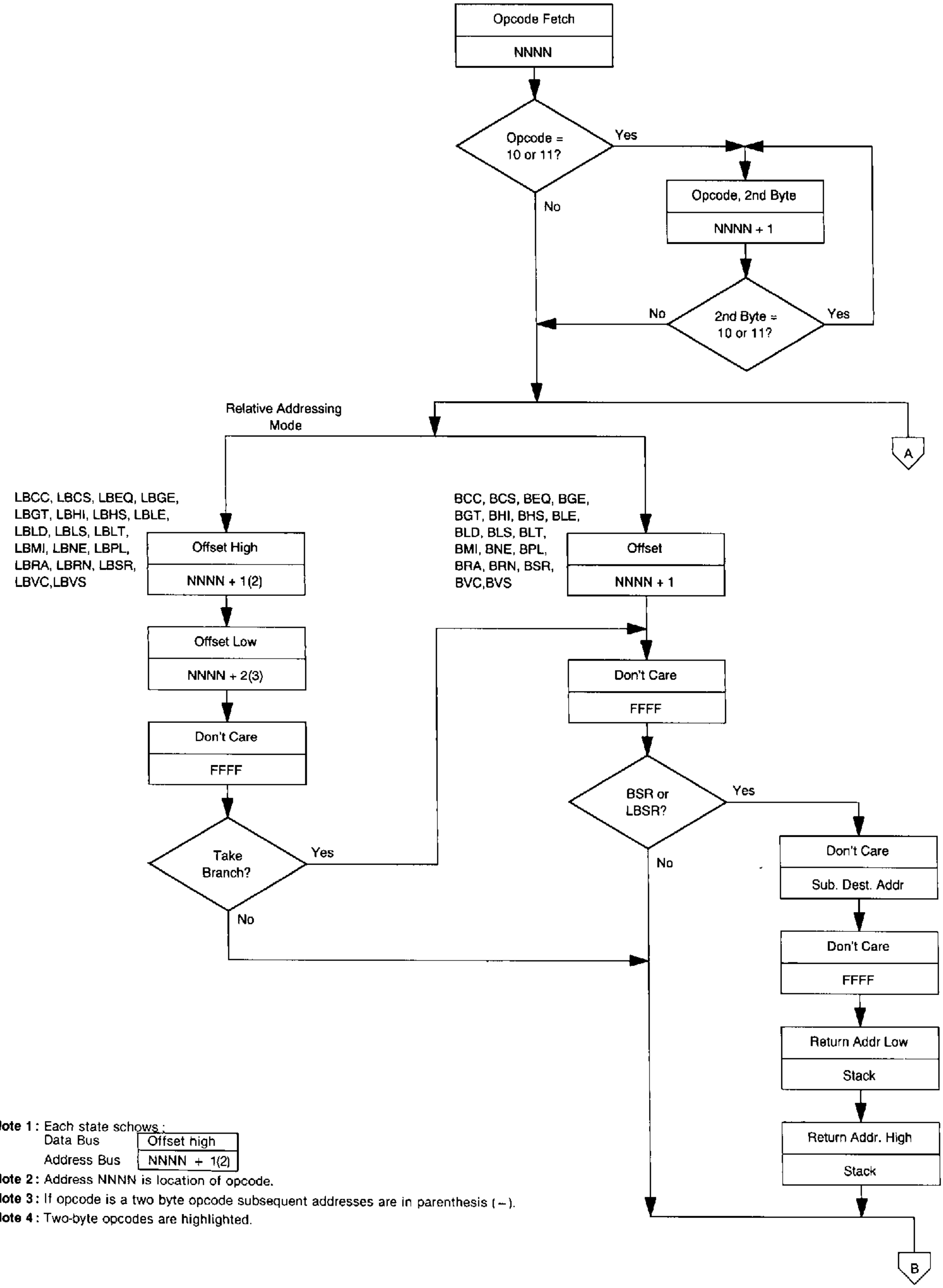


Note 1 : If the associated mask bit is set when the interrupt is requested, this cycle will be an instruction fetch from address location PC + 1. Howerver, if the interrupt is accepted ($\overline{NMI}$ or an unmasked $\overline{FIRQ}$ or $\overline{IRQ}$) interrupt processing continues with this cycle as m on Figures 9 and 10 (Interrupt Timing).

Note 2 : If mask bits are clear, $\overline{IRQ}$ and $\overline{FIRQ}$ must be held low for three cycles to guarantee interrupt to be taken, although only one cycle is necessary to bring the processor out of SYNC.

Note 3 : Waveform measurements for all inputs and outputs are specified at logic high 2.0 V and logic low 0.8 V unless otherwise specified.

Figure 17 : Sync timing.



Note 1 : Each state schows :

Data Bus	Offset high
Address Bus	NNNN + 1(2)

Note 2 : Address NNNN is location of opcode.

Note 3 : If opcode is a two byte opcode subsequent addresses are in parenthesis (–).

Note 4 : Two-byte opcodes are highlighted.

Figure 18 : Cycle-by-cycle performance (Sheet 1 of 9).

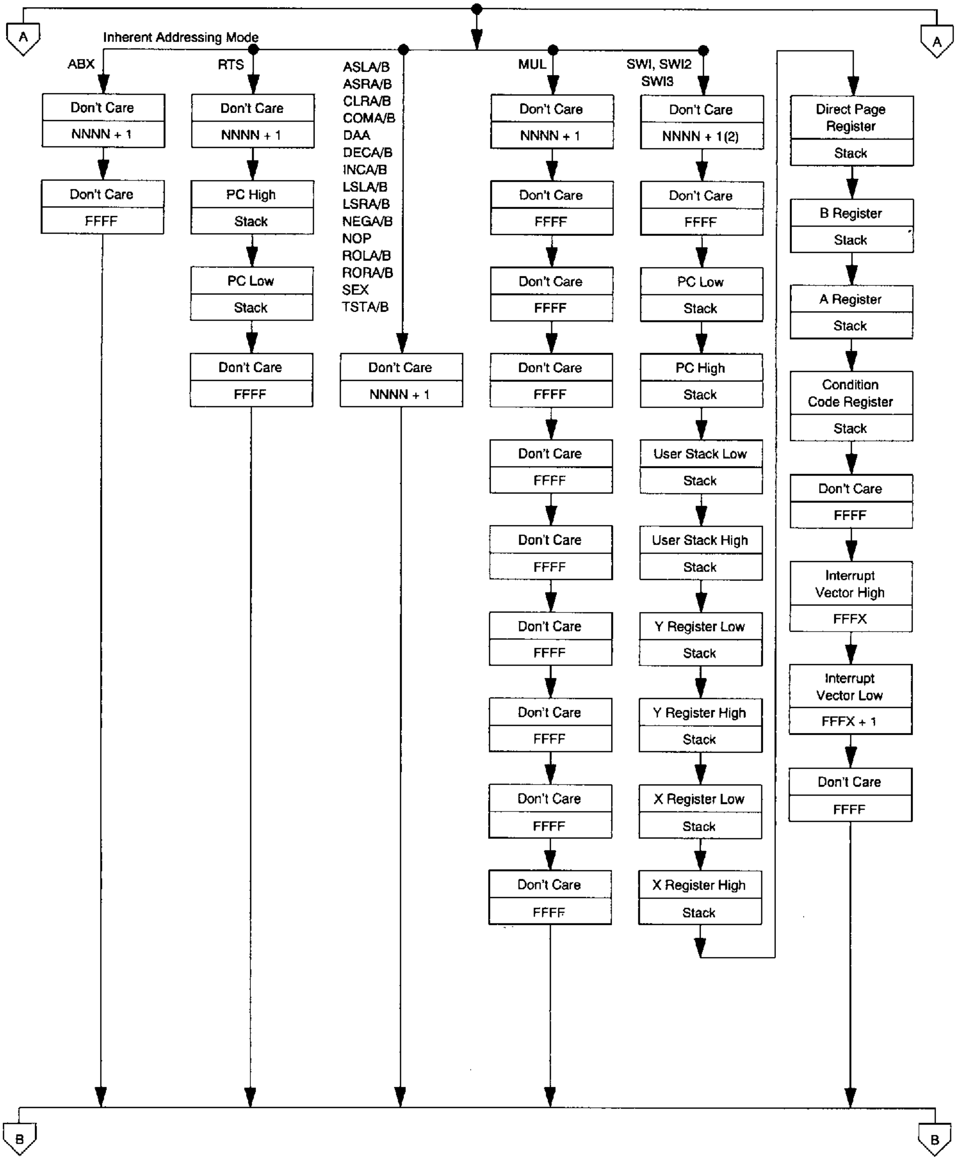


Figure 18 : Cycle-by-cycle performance (Sheet 2 of 9).

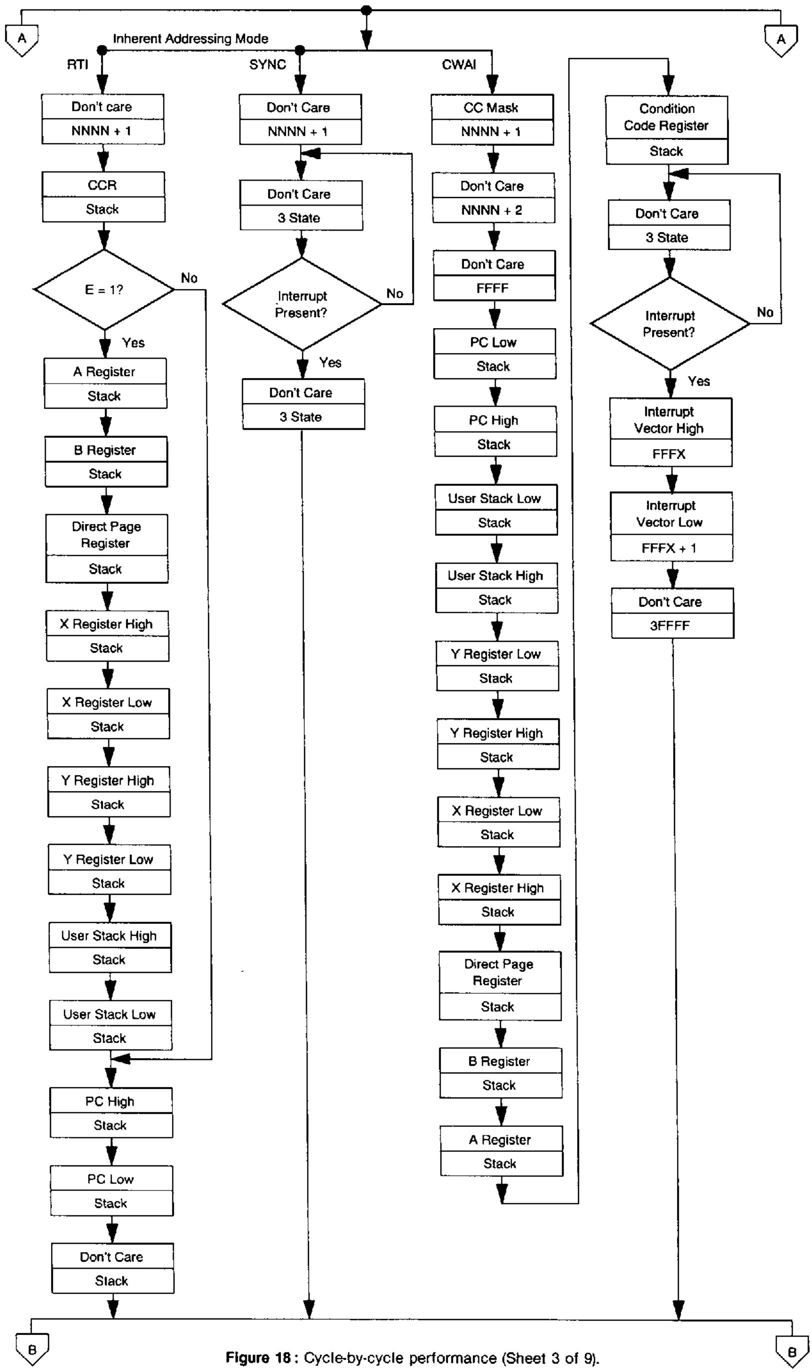


Figure 18 : Cycle-by-cycle performance (Sheet 3 of 9).

5

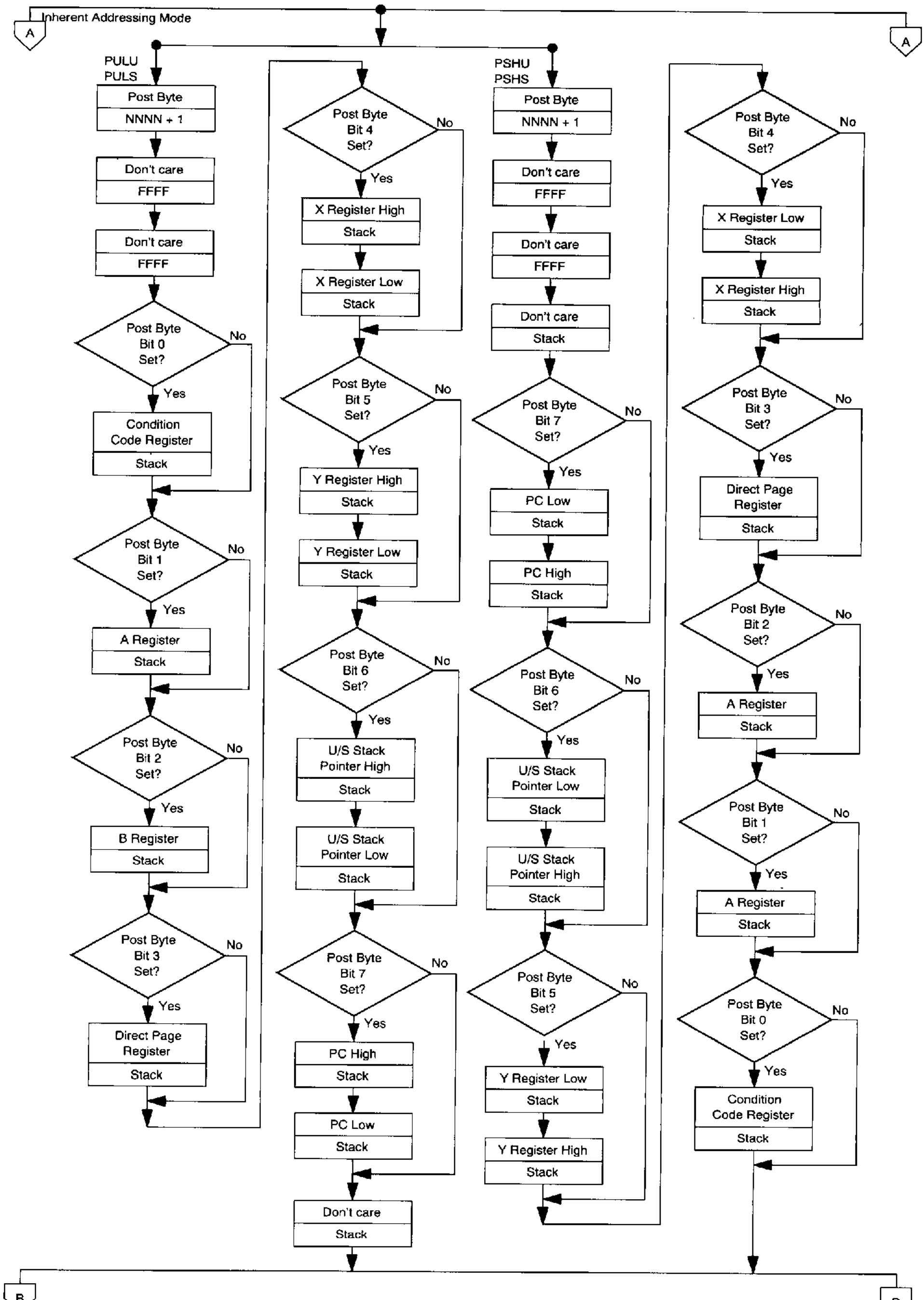


Figure 18 : Cycle-by-cycle performance (Sheet 4 of 9).

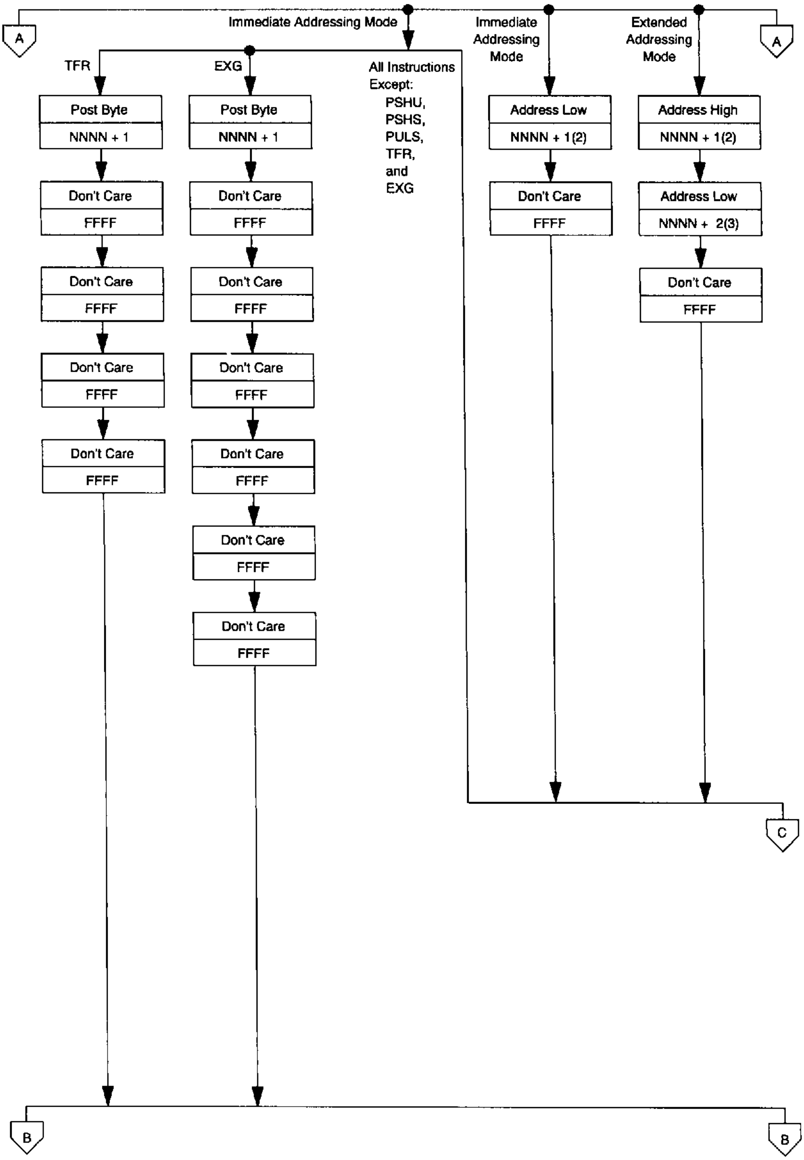
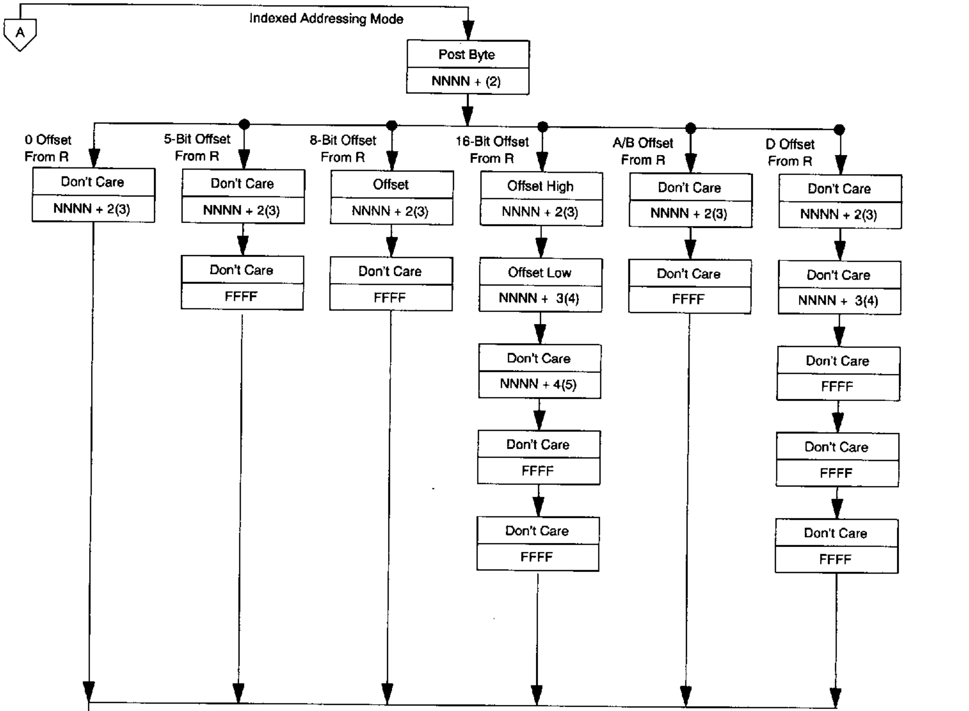
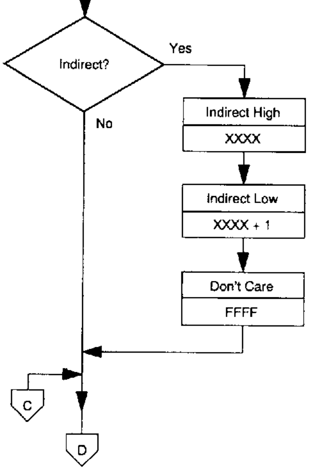


Figure 18 : Cycle-by-cycle performance (Sheet 5 of 9).

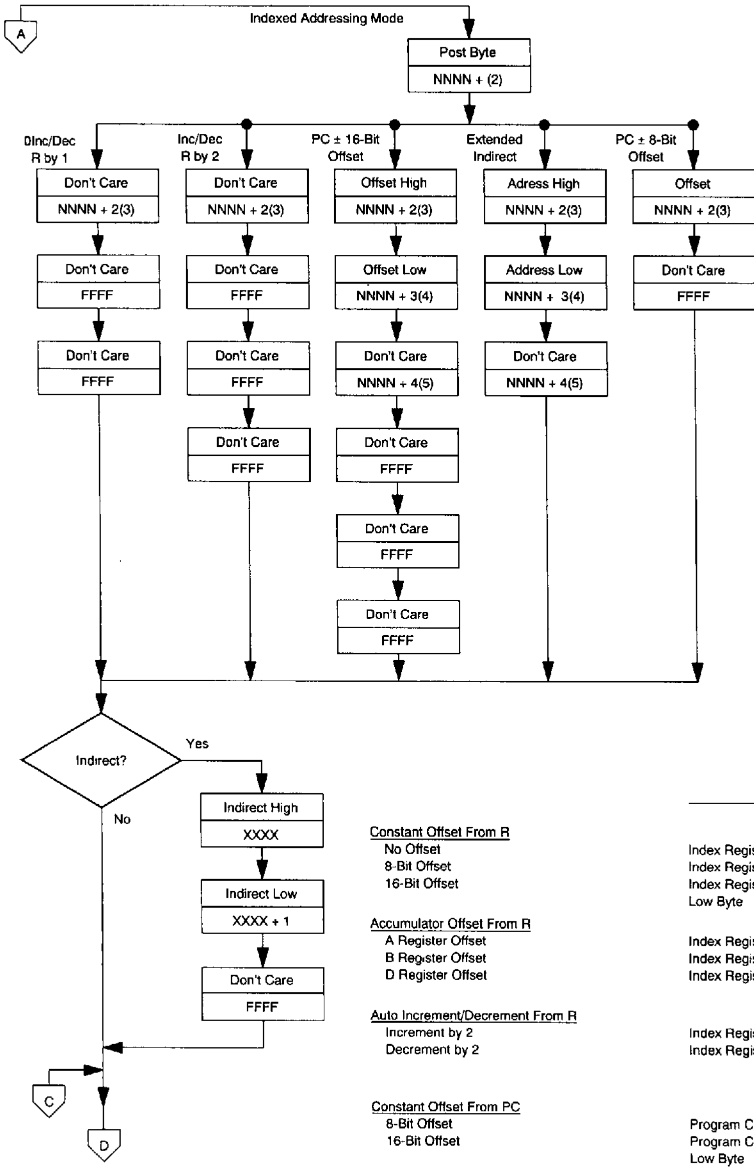




	XXXX
Constant Offset From R	
No Offset	Index Register
8-Bit Offset	Index Register + Offset Byte
16-Bit Offset	Index Register + Offset High Byte Offset Low Byte
Accumulator Offset From R	
A Register Offset	Index Register + A Register
B Register Offset	Index Register + B Register
D Register Offset	Index Register + D Register
Auto Increment/Decrement From R	
Increment by 2	Index Register*
Decrement by 2	Index Register - 2
Constant Offset From PC	
8-Bit Offset	Program Cunter + Offset Byte
16-Bit Offset	Program Cunter + Offset High Byte Offset Low Byte
Extended Indirect	
16-Bit Address	Address High Byte Addres Low Byte

* The index register is incremented following the indexed access

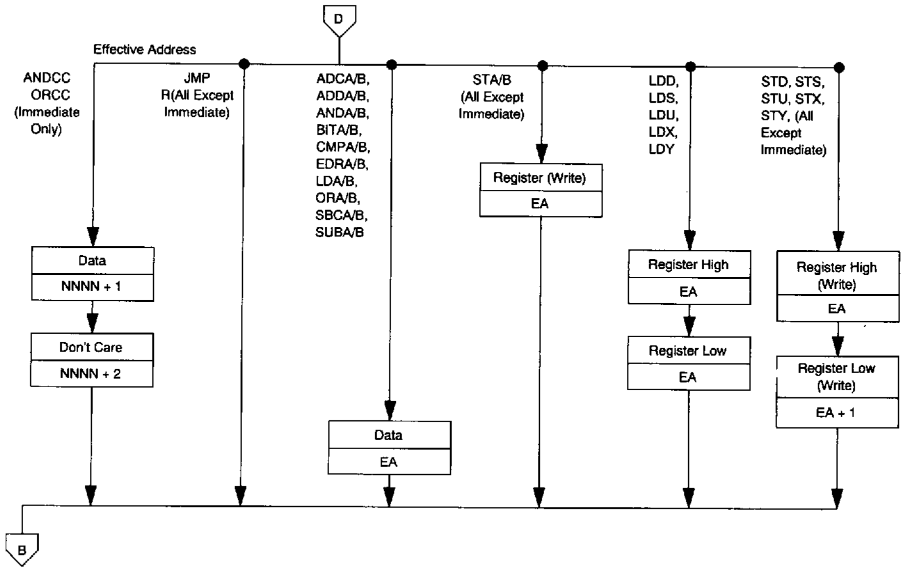
Figure 18 : Cycle-by-cycle performance (Sheet 6 of 9).



	XXXX
Constant Offset From R	
No Offset	Index Register
8-Bit Offset	Index Register + Offset Byte
16-Bit Offset	Index Register + Offset High Byte Offset Low Byte
Accumulator Offset From R	
A Register Offset	Index Register + A Register
B Register Offset	Index Register + B Register
D Register Offset	Index Register + D Register
Auto Increment/Decrement From R	
Increment by 2	Index Register*
Decrement by 2	Index Register - 2
Constant Offset From PC	
8-Bit Offset	Program Cunter + Offset Byte
16-Bit Offset	Program Cunter + Offset High Byte Offset Low Byte
Extended Indirect	
16-Bit Address	Address High Byte Addres Low Byte

* The index register is incremented following the indexed access

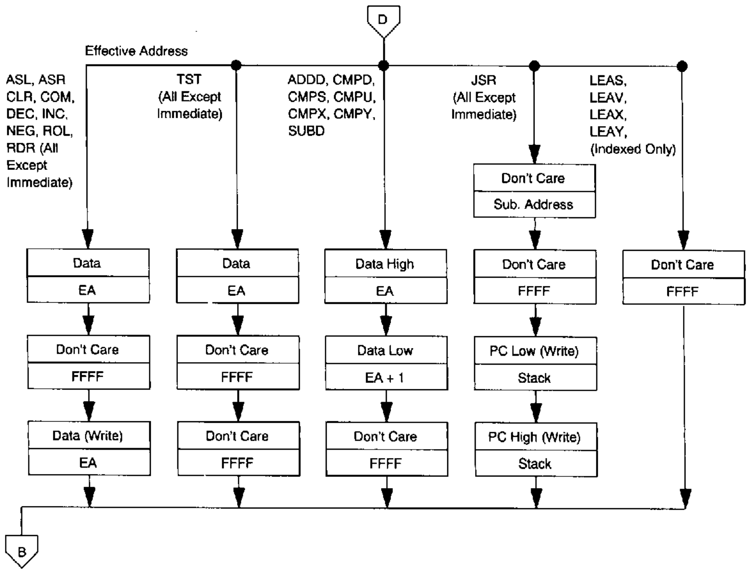
Figure 18 : Cycle-by-cycle performance (Sheet 7 of 9).



	Effective Address (EA)
Constant Offset From R	
No Offset	Index Register
5-Bit Offset	Index Register
8-Bit Offset	Index Register + Post Byte
16-Bit Offset	Index Register + Post Byte High Post Byte Low
Accumulator Offset From R	
A Register Offset	Index Register + A Register
B Register Offset	Index Register + B Register
D Register Offset	Index Register + D Register
Auto Increment/Decrement From R	
Increment by 1	Index Register*
Increment by 2	Index Register*
Decrement by 1	Index Register + 1
Decrement by 2	Index Register + 2
Constant Offset From PC	
8-Bit Offset	Program Cunter + Offset Byte
16-Bit Offset	Program Cunter + Offset High Byte Offset Low Byte
Direct	Direct Page Register Addres Low
Extended	Address High Addres Low
Immediate	NNNN + 1

* The index register is incremented following the indexed access

Figure 18 : Cycle-by-cycle performance (Sheet 8 of 9).



	Effective Address (EA)
Constant Offset From R	
No Offset	Index Register
5-Bit Offset	Index Register
8-Bit Offset	Index Register + Post Byte
16-Bit Offset	Index Register + Post Byte High Post Byte Low
Accumulator Offset From R	
A Register Offset	Index Register + A Register
B Register Offset	Index Register + B Register
D Register Offset	Index Register + D Register
Auto Increment/Decrement From R	
Increment by 1	Index Register*
Increment by 2	Index Register*
Decrement by 1	Index Register + 1
Decrement by 2	Index Register + 2
Constant Offset From PC	
8-Bit Offset	Program Cunter + Offset Byte
16-Bit Offset	Program Cunter + Offset High Byte Offset Low Byte
Direct	Direct Page Register Addres Low
Extended	Address High Addres Low
Immediate	NNNN + 1

* The index register is incremented following the indexed access

Figure 18 : Cycle-by-cycle performance (Sheet 9 of 9).

Table 8 - 8-Bit accumulator and memory instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory form accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)
Note : A, B, CC, or DP may be pushed to (pulled from) stack with either PSHS, PSHU (PULS, PULU) instructions.	

Table 9 - 16-Bit accumulator and memory instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, U, S, U, or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator

Table 9 - 16-Bit accumulator and memory instructions (Continued)

Mnemonic(s)	Operation
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U, or PC
TFR R, D	Transfer X, Y, S, U, or PC to D
Note : D may be pushed (pulled) to stack with either PSHS, PSHU (PULS, PULU) instructions.	

Table 10 - Index register/stack pointer instructions

Instruction	Description
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, X, U or PC with D, X, Y, S, U, or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S, or PC from hardware stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U, or PC
ABX	Add B accumulator to X (unsigned)

Table 11 - Branch instructions

Instruction	Description
SIMPLE BRANCHES	
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear

Table 11 · Branch instructions (Continued)

Instruction	Description
	SIGNED BRANCHES
BGT, LBGT	Branch if greater (signed)
BVS, LBVS	Branch if invalid 2s complement result
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BLE, LBLE	Branch if less than or equal (signed)
BVC, LBVC	Branch if valid 2s complement result
BLT, LBLT	Branch if less than (signed)
	USIGNED BRANCHES
BHI, LBHI	Branch if higher (unsigned)
BCC, LBCC	Branch if higher or same (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BLS, LBLS	Branch if lower or same (unsigned)
BCS, LBCS	Branch if lower (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	OTHER BRANCHES
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 12 · Miscellaneous instructions

Instruction	Description
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

Table 13 - Hexadecimal values of machine codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*				31	LEAY		4+	2+	61	*			
02	*				32	LEAS		4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Immed	5+	2	64	LSR		6+	2+
05	*				35	PULS	Immed	5+	2	65	*			
06	ROR		6	2	36	PSHU	Immed	5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU	Immed	5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*	—			68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS	Inherent	5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX		3	1	6A	DEC		6+	2+
0B	*				3B	RTI		6/15	1	6B	*			
0C	INC		6	2	3C	CWAI		≥ 20	2	6D	INC		6+	2+
0D	TST		6	2	3D	MUL	Inherent	11	1	6D	TST		6+	2+
0E	JMP		3	2	3E	*				6E	JMP		3+	2+
0F	CLR	Direct	6	2	3F	SWI	Inherent	19	1	6F	CLR	Indexed	6+	2+
10	Page 2				40	NEGA	Inherent	2	1	70	NEG	Extended	7	3
11	Page 3				41	*				71	*			
12	NOP	Inherent	2	1	42	*				72	*			
13	SYNC	Inherent	≥ 4	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Inherent	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*				4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Inherent	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG	Immed	8	2	4E	*				7E	JMP		4	3
1F	TFR	Immed	6	2	4F	CLRA	Inherent	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Inerent	2	1	80	SUBA	Immed	2	2
21	BRN		3	2	51	*				81	CMPA		2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX		4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Immed	7	2
2E	BGT		3	2	5E	*				8E	LDX	Relative	3	3
2F	BLE	Relative	3	2	5F	CLRB	Inherent	2	1	8F	*	Immed		

Table 13 - Hexadecimal values of machine codes (Continued)

OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2
91	CMPA		4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
97	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX		5	2
9F	STX	Direct	5	2
A0	SUBA	Indexed	4+	2+
A1	CMPA		4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	ANDA		4+	2+
A5	BITA		4+	2+
A6	LDA		4+	2+
A7	STA		4+	2+
A8	EORA		4+	2+
A9	ADCA		4+	2+
AA	ORA		4+	2+
AB	ADDA		4+	2+
AC	CMPX		6+	2+
AD	JSR		7+	2+
AE	LDX		5+	2+
AF	STX	Indexed	5+	2+
B0	SUBA	Extended	5	3
B1	CMPA		5	3
B2	SBCA		5	3
B3	SUBD		7	3
B4	ANDA		5	3
B5	BITA		5	3
B6	LDA		5	3
B7	STA		5	3
B8	EORA		5	3
B9	ADCA		5	3
BA	ORA		5	3
BB	ADDA		5	3
BC	CMPX		7	3
BD	JSR		8	3
BE	LDX		6	3
BF	STX	Extended	6	3

OP	Mnem	Mode	~	#
C0	SUBB	Immed	2	2
C1	CMPB		2	2
C2	SBCB		2	2
C3	ADDD		4	3
C4	ANDB		2	2
C5	BITB	Immed	2	2
C6	LDB	Immed	2	2
C7	*			
C8	EORB		2	2
C9	ADCB		2	2
CA	ORB		2	2
CB	ADDB		2	2
CC	LDD		3	3
CD	*			
CE	LDU	Immed	3	3
CF	*			
D0	SUBB	Direct	4	2
D1	CMPB		4	2
D2	SBCB		4	2
D3	ADDD		6	2
D4	ANDB		4	2
D5	BITB		4	2
D6	LDB		4	2
D7	STB		4	2
D8	EORB		4	2
D9	ADCB		4	2
DA	ORB		4	2
DB	ADDB		4	2
DC	LDD		5	2
DD	STD		5	2
DE	LDU		5	2
DF	STU	Direct	5	2
E0	SUBB	Indexed	4+	2+
E1	CMPB		4+	2+
E2	SBCB		4+	2+
E3	ADDD		6+	2+
E4	ANDB		4+	2+
E5	BITB		4+	2+
E6	LDB		4+	2+
E7	STB		4+	2+
E8	EORB		4+	2+
E9	ADCB		4+	2+
EA	ORB		4+	2+
EB	ADDB		4+	2+
EC	LDD		5+	2+
ED	STD		5+	2+
EE	LDU		5+	2+
EF	STU	Indexed	5+	2+
F0	SUBB	Extended	5	3
F1	CMPB		5	3
F2	SBCB		5	3
F3	ADDD		7	3
F4	ANDB		5	3
F5	BITB		5	3
F6	LDB		5	3
F7	STB		5	3
F8	EORB		5	3
F9	ADCB		5	3
FA	ORB		5	3
FB	ADDB	Extended	5	3
FC	LDD	Extended	6	3
FD	STD		6	3
FE	LDU		6	3
FF	STU	Extended	6	3

OP	Mnem	Mode	~	#
Page 2 and 3 Machine Codes				
1021	LBRN	Relative	5	4
1022	LBHI		5(6)	4
1023	LBLS		5(6)	4
1024	LBHS, LBCC		5(6)	4
1025	LBCS, LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT		5(6)	4
102E	LBGT		5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI2	Inherent	20	2
1083	CMPD	Immed	5	4
108C	CMPY		5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY		7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY		7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY		8	4
10BE	LDY		7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI3	Inherent	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

Note : All unused opcodes are undefined and illegal.

Legend :

~ Number of MPU cycles (less possible push pull or indexed-mode cycles)

\# Number of program bytes

* Denotes unused opcode

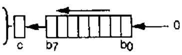
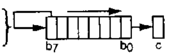
Instruction	Forms	Immediate Op	Immediate ~	Immediate #	Direct Op	Direct ~	Direct #	Indexed Op	Indexed ~	Indexed #	Extended Op	Extended ~	Extended #	Inherent Op	Inherent ~	Inherent #	Description	5 H	3 N	2 Z	1 V	0 C
ABX														3A	3	1	B + X → X (Unsigned)	•	•	•	•	•
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C → A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M → A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M → B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M:M + 1 → D	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A Λ M → A	•	↕	↕	0	•
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B Λ M → B	•	↕	↕	0	•
	ANDCC	1C	3	2													CC Λ IMM → CC					7
ASL	ASLA													48	2	1	 A	8	↕	↕	↕	↕
	ASLB													58	2	1	B	8	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3				M	8	↕	↕	↕	↕
ASR	ASRA													47	2	1	 A	8	↕	↕	•	↕
	ASRB													57	2	1	B	8	↕	↕	•	↕
	ASR				07	6	2	67	6+	2+	77	7	3				M	8	↕	↕	•	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M Λ A)	•	↕	↕	0	•
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M Λ B)	•	↕	↕	0	•
CLR	CLRA													4F	2	1	0 → A	•	0	1	0	0
	CLRB													5F	2	1	0 → B	•	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0 → M	•	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	8	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	8	↕	↕	↕	↕
	CMPD	10 83	5	4	10 93	7	3	10 A3	7+	3+	10 B3	8	4				Compare M:M + 1 from D	•	↕	↕	↕	↕
	CMPS	11 8C	5	4	11 9C	7	3	11 AC	7+	3+	11 BC	8	4				Compare M:M + 1 from S	•	↕	↕	↕	↕
	CMPU	11 83	5	4	11 93	7	3	11 A3	7+	3+	11 B3	8	4				Compare M:M + 1 from U	•	↕	↕	↕	↕
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M:M + 1 from X	•	↕	↕	↕	↕
	CMPY	10 8C	5	4	10 9C	7	3	10 AC	7+	3+	10 BC	8	4				Compare M:M + 1 from Y	•	↕	↕	↕	↕
COM	COMA													43	2	1	$\overline{A}$ → A	•	↕	↕	0	1
	COMB													53	2	1	$\overline{B}$ → B	•	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M}$ → M	•	↕	↕	0	1
CWAI		3C	≥20	2													CC Λ IMM → Wait for interrupt					7
DAA														19	2	1	Decimal adjust A	•	↕	↕	0	↕
DEC	DECA													4A	2	1	A − 1 → A	•	↕	↕	↕	•
	DECB													5A	2	1	B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M + 1 → M	•	↕	↕	↕	•
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ∀M → A	•	↕	↕	0	•
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ∀M → B	•	↕	↕	0	•
EXG	R1, R2	1E	8	2													R1 → R2[1]	•	•	•	•	•
INC	INCA													4C	2	1	A + 1 → A	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA[3] → PC	•	•	•	•	•
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to subroutine	•	•	•	•	•
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M → A	•	↕	↕	0	•
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M → B	•	↕	↕	0	•
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M:M + 1 → D	•	↕	↕	0	•
	LDS	10 CE	4	4	10 DE	6	3	10 EE	6+	3+	10 FE	7	4				M:M + 1 → S	•	↕	↕	0	•
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M + 1 → U	•	↕	↕	0	•
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M + 1 → X	•	↕	↕	0	•
	LDY	10 8E	4	4	10 9E	6	3	10 AE	6+	3+	10 BE	7	4				M:M + 1 → Y	•	↕	↕	0	•

Figure 19 : Programming AID.

5

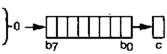
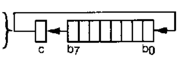
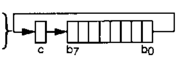
Instruction	Forms	Addressing Modes															Description	5	3	2	1	0
		Immediate			Direct			Indexed			Extended			Inherent								
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	N	Z	V	C
LEA	LEAS							32	4+	2+							EA[3] → S	•	•	•	•	•
	LEAU							33	4+	2+							EA[3] → U	•	•	•	•	•
	LEAX							30	4+	2+							EA[3] → X	•	•	↕	•	•
	LEAY							31	4+	2+							EA[3] → Y	•	•	↕	•	•
LSL	LSLA													48	2	1	A 	•	↕	↕	↕	↕
	LSLB													58	2	1	B	•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3				M	•	↕	↕	↕	↕
LSR	LSRA													44	2	1	A 	•	0	↕	•	↕
	LSRB													54	2	1	B	•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74	7	3				M	•	0	↕	•	↕
MUL														3D	11	1	A × B → D (unsigned)	•	•	↕	•	9
NEG	NEGA													40	2	1	$\bar{A} + 1 \rightarrow A$	8	↕	↕	↕	↕
	NEGB													50	2	1	$\bar{B} + 1 \rightarrow B$	8	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\bar{M} + 1 \rightarrow M$	8	↕	↕	↕	↕
NOP														12	2	1	No operation	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A V M → A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	4+	2+	FA	5	3				B V M → B	•	↕	↕	0	•
	ORCC	1A	3	2													CC V IMM → CC				7	
PSH	PSHS	34	5+[3]	2													Pull registers on S stack	•	•	•	•	•
	PSHU	36	5+[3]	2													Pull registers on U stack	•	•	•	•	•
PUL	PULS	35	5+[3]	2													Push registers from S stack	•	•	•	•	•
	PULU	37	5+[3]	2													Push registers from S stack	•	•	•	•	•
ROL	ROLA													49	2	1	A 	•	↕	↕	↕	↕
	ROLB													59	2	1	B	•	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3				M	•	↕	↕	↕	↕
ROR	RORA													46	2	1	A 	•	↕	↕	•	↕
	RORB													56	2	1	B	•	↕	↕	•	↕
	ROR				06	6	2	66	6+	2+	76	7	3				M	•	↕	↕	•	↕
RTI														38	6/15	1	Return from interrupt					7
RTS														39	5	1	Return from subroutine	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A − M − C → A	8	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B − M − C → B	8	↕	↕	↕	↕
SEX														1D	2	1	Sign extend B into A	•	↕	↕	0	•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A → M	•	↕	↕	0	•
	STB				D7	4	2	E7	4+	2+	F7	5	3				B → M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D → M:M + 1	•	↕	↕	0	•
	STS				10 DF	6	3	10 EF	6+	3+	10 FF	7	4				S → M:M + 1	•	↕	↕	0	•
	STU				DF	5	2	EF	5+	2+	FF	6	3				U → M:M + 1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X → M:M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 AF	6+	3+	10 BF	7	4				Y → M:M + 1	•	↕	↕	0	•
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A − M → A	8	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B − M → B	8	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D − M:M + 1 → D	•	↕	↕	↕	↕
SWI	SWI[4]													3F	19	1	Software interrupt 1	•	•	•	•	•
	SWI2[4]													10 3F	20	2	Software interrupt 2	•	•	•	•	•
	SWI3[4]													11 3F	20	1	Software interrupt 3	•	•	•	•	•

Figure 19 : Programming AID (continued).

Instruction	Forms	Addressing Modes															Description	5	3	2	1	0
		Immediate			Direct			Indexed			Extended			Inherent								
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	N	Z	V	C
SYNC														13	≥4	1	Synchronize to interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2													R1 → R2[2]	•	•	•	•	•
TST	TSTA TSTB TST				0D	6	2	6D	6+	2+	7D	7	3	4D 5D	2 2	1 1	Test A Test B Test M	• • •	↕ ↕ ↕	↕ ↕ ↕	0 0 0	• • •

Note 1: This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEDEX ADDRESSING MODE table, Table 2.

Note 2: R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are : A, B, CC, DP
The 16 bit registers are : X, Y, U, S, D, PC.

Note 3: EA is the effective address.

Note 4: The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.

Note 5: 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).

Note 6: SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.

Note 7: Conditions Codes set as a direct reseult of the instruction.

Note 8: Vaue of half-carry flag is undefined.

Note 9: Special Case – Carry set if b7 is SET.

Legend :

- OP Operation Code (Hexadecimal)
- ~ Number of MPU Cycles
- # Number of Program Bytes
- \+ Arithmetic Plus
- − Arithmetic Minus
- • Multiply
- $\overline{M}$ Complement of M
- → Transfer Into
- H Half-carry (from bit 3)
- N Negative (sign byte)
- Z Zero result
- V Overflow, 2's complement
- C Carry from ALU
- ↕ Test and set if true, cleared otherwise
- • Not Affected
- CC Condition Code Register
- : Concatenation
- V Logical or
- Λ Logical and
- ⊻ Logical Exclusive or

Figure 19 : Programming AID (continued).

Branch Instructions

Instruction	Forms	Addressing Modes Relative			Description	5	3	2	1	0
		Op	~5	#		H	N	Z	V	C
BCC	BCC LBCC	24 10 24	3 5(6)	2 4	Branch C = 0 Long branch C = 0	• •	• •	• •	• •	• •
BCS	BCS LBCS	25 10 25	3 5(6)	2 4	Branch C = 1 Long branch C = 1	• •	• •	• •	• •	• •
BEQ	BEQ LBEQ	27 10 27	3 5(6)	2 4	Branch Z = 1 Long branch Z = 0	• •	• •	• •	• •	• •
BGE	BGE LBGE	2C 10 2C	3 5(6)	2 4	Branch ⩾ Zero Long branch ⩾ Zero	• •	• •	• •	• •	• •
BGT	BGT LBGT	2E 10 2E	3 5(6)	2 4	Branch > Zero Long branch > Zero	• •	• •	• •	• •	• •
BHI	BHI LBHI	22 10 22	3 5(6)	2 4	Branch higher Long branch higher	• •	• •	• •	• •	• •
BHS	BHS LBHS	24 10 24	3 5(6)	2 4	Branch higher or Same Long branch higher or Same	• •	• •	• •	• •	• •
BLE	BLE LBLE	2F 10 2F	3 5(6)	2 4	Branch ⩽ Zero Long branch ⩽ Zero	• •	• •	• •	• •	• •
BLO	BLO LBLO	25 10 25	3 5(6)	2 4	Branch lower Long branch Lower	• •	• •	• •	• •	• •

Branch Instructions (Continued)

Instruction	Forms	Addressing Modes Relative			Description	5	3	2	1	0
		Op	~5	#		H	N	Z	V	C
BLS	BLS LBLS	23 10 23	3 5(6)	2 4	Branch lower or Same Long branch lower or same	• •	• •	• •	• •	• •
BLT	BLT LBLT	2D 10 2D	3 5(6)	2 4	Branch < Zero Long branch < Zero	• •	• •	• •	• •	• •
BMI	BMI LBMI	2B 10 2B	3 5(6)	2 4	Branch minus Long branch < Minus	• •	• •	• •	• •	• •
BNE	BNE LBNE	26 10 26	3 5(6)	2 4	Branch Z = 0 Long branch Z ≠ 0	• •	• •	• •	• •	• •
BPL	BPL LBPL	2A 10 2A	3 5(6)	2 4	Branch plus Long branch plus	• •	• •	• •	• •	• •
BRA	BRA LBRA	20 16	3 5	2 3	Branch always Long branch always	• •	• •	• •	• •	• •
BRN	BRN LBRN	21 10 21	3 5	2 4	Branch never Long branch never	• •	• •	• •	• •	• •
BSR	BSR LBSR	8D 17	7 9	2 3	Branch to subroutine Long branch to subroutine	• •	• •	• •	• •	• •
BVC	BVC LBVC	28 10 28	3 5(6)	2 4	Branch V = 0 Long branch V = 0	• •	• •	• •	• •	• •
BVS	BVS LBVS	29 10 29	3 5(6)	2 4	Branch V = 1 Long branch V = 1	• •	• •	• •	• •	• •

Simple branches

	OP	~	#
BRA	20	3	2
LBRA	16	5	3
BRN	21	3	2
LBRN	1021	5	4
BSR	8D	7	2
LBSR	17	9	3

Signed conditional branches (Notes 1-4)

Test	True	OP	False	OP
r > m	BGT	2E	BLE	2F
r ⩾ m	BGE	2C	BLT	2D
r =	BEQ	27	BNE	26
r ⩽ m	BLE	2F	BGT	2E
r < m	BLT	2D	BGE	2C

Simple conditional branches (Notes 1-4)

Test	True	OP	False	OP
N = 1	BMI	2B	BPL	2A
Z = 1	BEQ	27	BNE	26
V = 1	BVS	29	BVC	28
C = 1	BCS	25	BCC	24

Unsigned condtional branches (Notes 1-4)

Test	True	OP	False	OP
r > m	BHI	22	BLS	23
r ⩾ m	BHS	24	BLO	25
r = m	BEQ	27	BNE	26
r ⩽ m	BLS	23	BHI	22
r < m	BLO	25	BHS	24

Note 1 : All conditional branches have both short and long variations.
Note 2 : All short branches are two bytes and require three cycles.
Note 3 : All conditional long branches are formed by prefixing the short opcode with $10 and using a 16-bit destination offset.
Note 4 : All conditional long branches require four bytes and six cycles if the branch is taken or five if the branch is not taken.

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 prodcuts are in compliance either with MIL-STD-883 or CECC 90000 and guarantying the parameters are tested at extreme temperatures for the entire temperature range.

8 - HANDLING

MOS devices must be handled with certain precautions to avoid damage due to accumulation of static charge. Input proctection devices have been designed in the chip to minimize the effect of this static buildup. Howerver, 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.

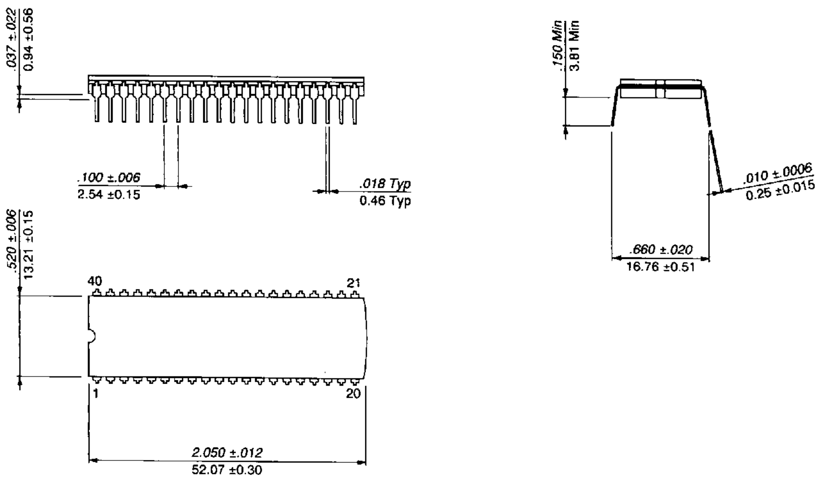
d) Store devices in conductive foam or carriers.

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

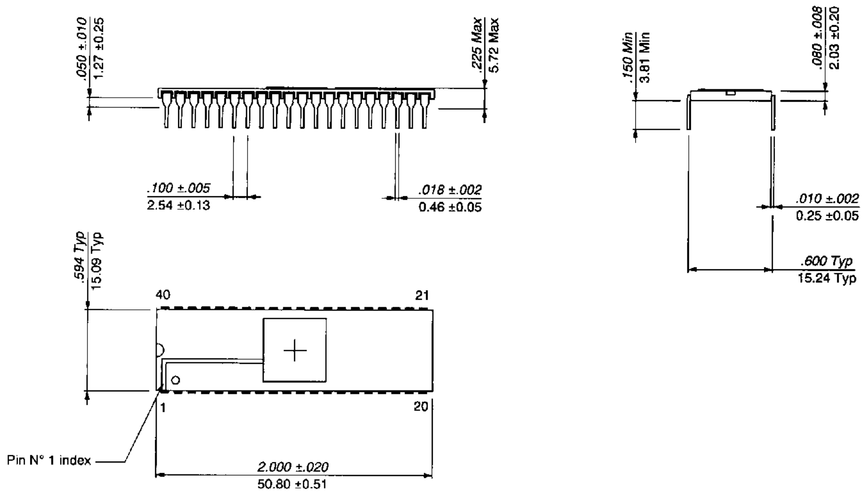
f) Maintain relative humidity above 50%, if practical.

9 - PACKAGE MECHANICAL DATA

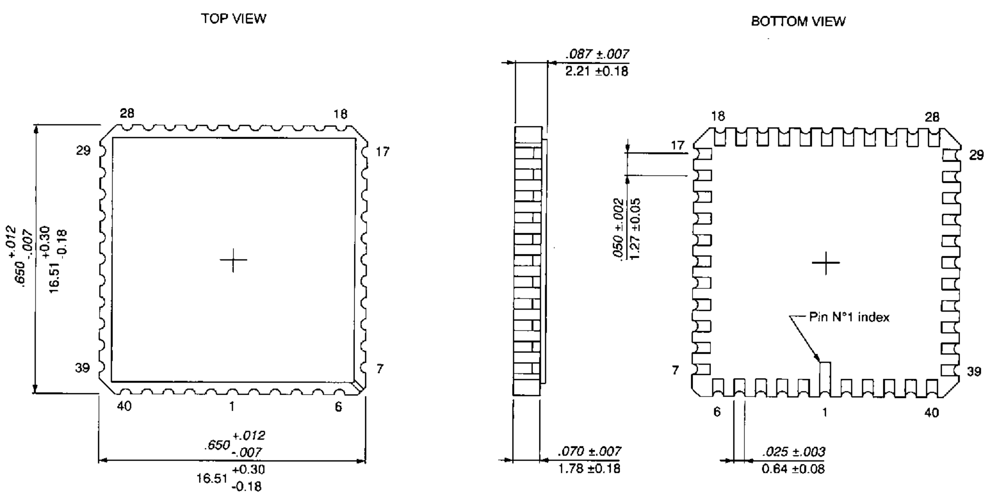
9.1 - DIL 40 : Ceramic Cerdip package



9.2 - DIL 40 : Ceramic Side Brazed package

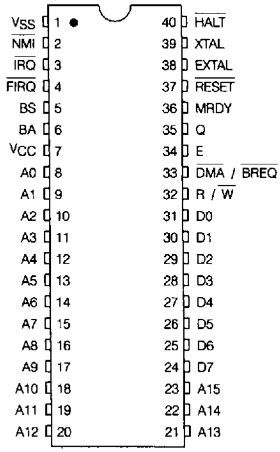


9.3 - LCCC 44 : Leadless Ceramic Chip Carrier

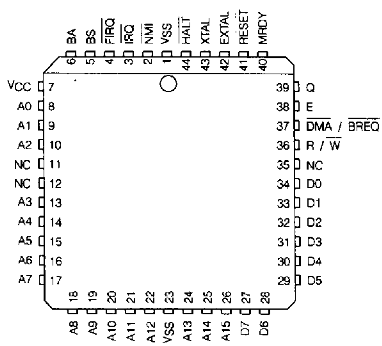


10 · TERMINAL CONNECTIONS

10.1 · DIL 40 pin assignment



10.2 · LCCC 44 pin assignment



5

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
EF6809JMG/B*	NFC 96883 - Class G	DIL Cerdip	– 55 / + 125	1	Data-sheet
EF6809CMG/B	NFC 96883 - Class G	DIL side brazed	– 55 / + 125	1	Data-sheet
EF6809EMG/B	NFC 96883 - Class G	LCCC	– 55 / + 125	1	Data-sheet
EF68A09JMG/B*	NFC 96883 - Class G	DIL Cerdip	– 55 / + 125	1.5	Data-sheet
EF68A09CMG/B	NFC 96883 - Class G	DIL side brazed	– 55 / + 125	1.5	Data-sheet
EF68A09EMG/B	NFC 96883 - Class G	LCCC	– 55 / + 125	1.5	Data-sheet
EF6809JMB/C*	MIL STD 883 - Class B	DIL Cerdip	– 55 / + 125	1	Data-sheet
EF6809CMB/C	MIL STD 883 - Class B	DIL side brazed	– 55 / + 125	1	Data-sheet
EF6809E1MB/T	TMS - Class B	LCCC	– 55 / + 125	1	Data-sheet
EF68A09JMB/C*	MIL STD 883 - Class B	DIL Cerdip	– 55 / + 125	1.5	Data-sheet
EF68A09CMB/C	MIL STD 883 - Class B	DIL side brazed	– 55 / + 125	1.5	Data-sheet
EF68A09E1MB/T	TMS - Class B	LCCC	– 55 / + 125	1.5	Data-sheet

* : Preferred package.
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
EF6809CV	TMS standard	DIL side brazed	– 40 / + 85	1	Data sheet
EF6809JV*	TMS standard	Cerdip DIL	– 40 / + 85	1	Data sheet
EF68A09CV	TMS standard	DIL side brazed	– 40 / + 85	1.5	Data sheet
EF68A09JV*	TMS standard	Cerdip DIL	– 40 / + 85	1.5	Data sheet
EF6809JM*	TMS standard	Cerdip DIL	– 55 / + 125	1	Data sheet
EF6809EM	TMS standard	LCCC	– 55 / + 125	1	Data sheet
EF6809CM	TMS standard	Side brazed DIL	– 55 / + 125	1	Data sheet
EF68A09JM*	TMS standard	Cerdip DIL	– 55 / + 125	1.5	Data sheet
EF68A09EM	TMS standard	LCCC	– 55 / + 125	1.5	Data sheet
EF68A09CM	TMS standard	Side brazed DIL	– 55 / + 125	1.5	Data sheet
EF6809C	TMS standard	DIL side brazed	0 / + 70	1	Data sheet
EF6809J*	TMS standard	Cerdip DIL	0 / + 70	1	Data sheet
EF68A09C	TMS standard	DIL side brazed	0 / + 70	1.5	Data sheet
EF68A09J*	TMS standard	Cerdip DIL	0 / + 70	1.5	Data sheet
EF68B09J*	TMS standard	Cerdip DIL	0 / + 70	2	Data sheet

* : J package is the preferred package.
Note : THOMSON COMPOSANTS MILITAIRES ET SPATIAUX.

EF6809 E 1 M B/T

Type — EF6809

Packages :
C = Ceramic DIL
E = Ceramic LCCC
E1 = LCC + Hot solder DIP
J = Tin dipped cerdip DIL (*)

Lead finish
__ = GOLD (exept cerdip)
1 = Tinned 883C B/C
7 = Tinned CECC B/Y
8 = Golded CECC B/Y

(*) preferred package

Temperature / Tcase:
M = -55°C / +125°C
V = -40°C / +85°C
__ = 0 / +70°C

Screening:
___ = Standard
G/B = NFC 96883 Cl.G
B/C = MIL STD 883 Cl.B
B/Y = CECC 90000 Cl.B
B/T = according to MIL STD 883 class B

5