Wally Webb 28-5-84

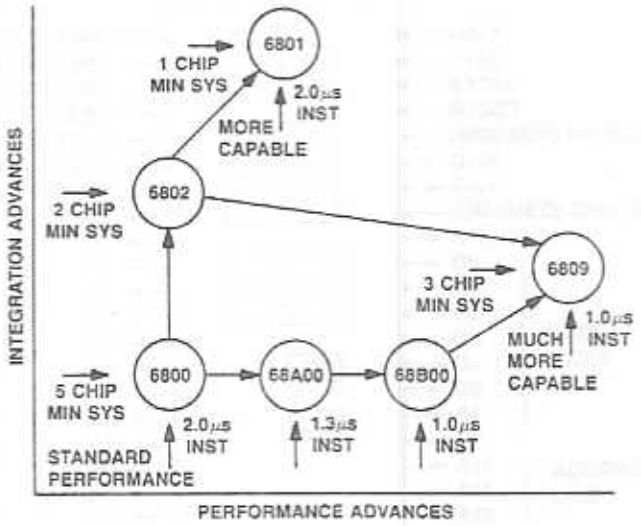
Motorola Microcomputers

MC6809
COURSE OUTLINE

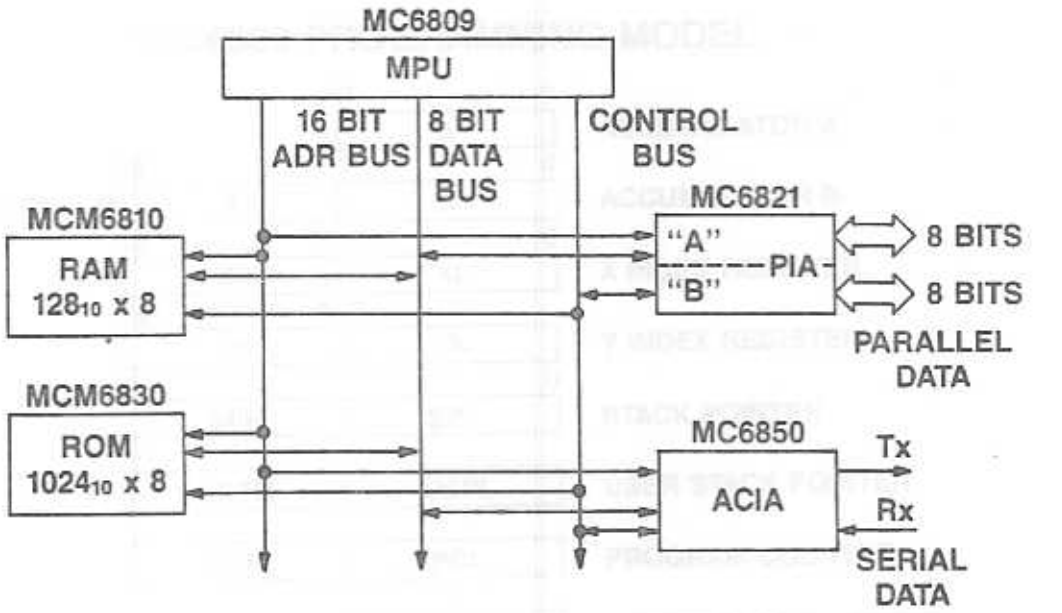
- INTRODUCTION/OVERVIEW
- PROGRAMMING MODEL
- SIGNAL AND CONTROL LINES
- INSTRUCTION SET
- ADDRESSING MODES
- COMPARISONS WITH OTHER 68XX'S
- MACRO ASSEMBLER
- PROGRAMMING EXAMPLES
- PROBLEM SET

G642

M6800 Microprocessor/Microcomputer Family Evolution

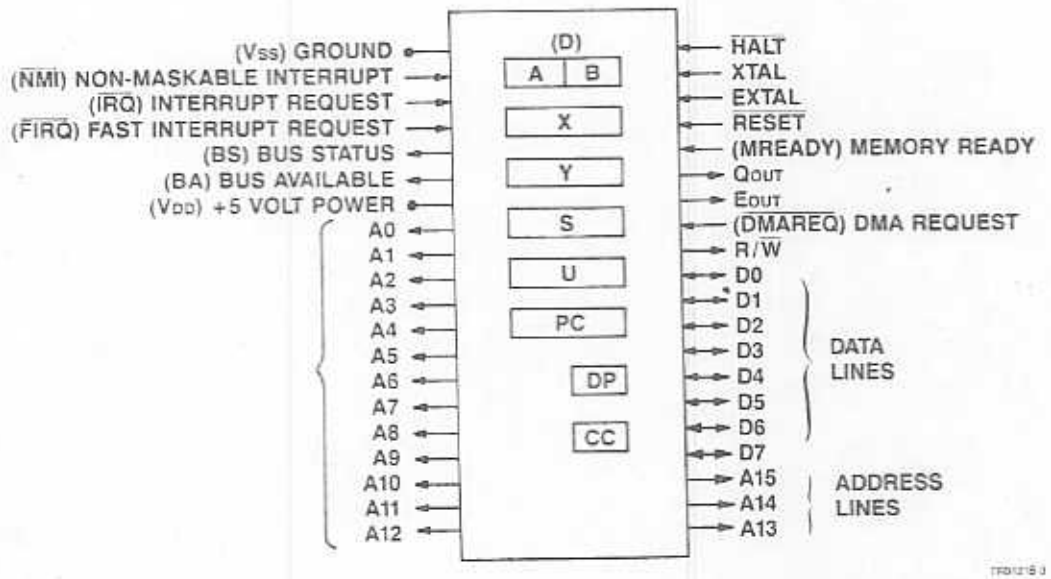


◗ A CONSISTENT, COMPATIBLE FAMILY

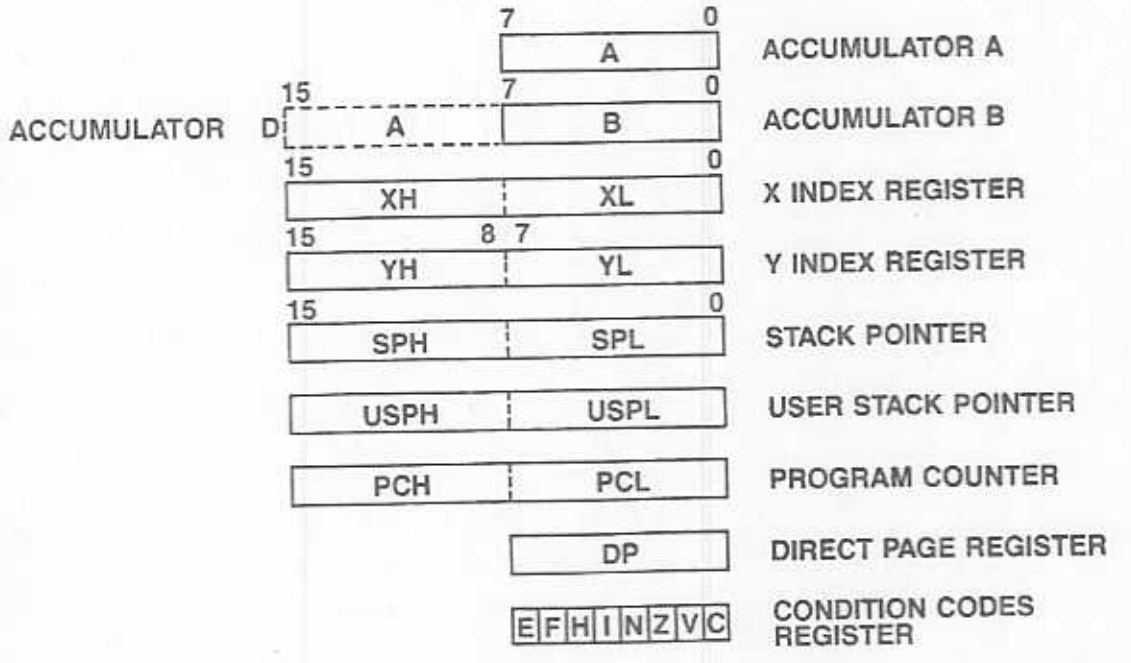


TR9120

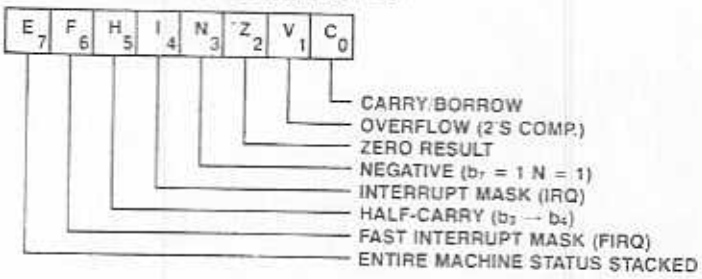
MC6809



MC6809 PROGRAMMING MODEL



CONDITION CODE REGISTER

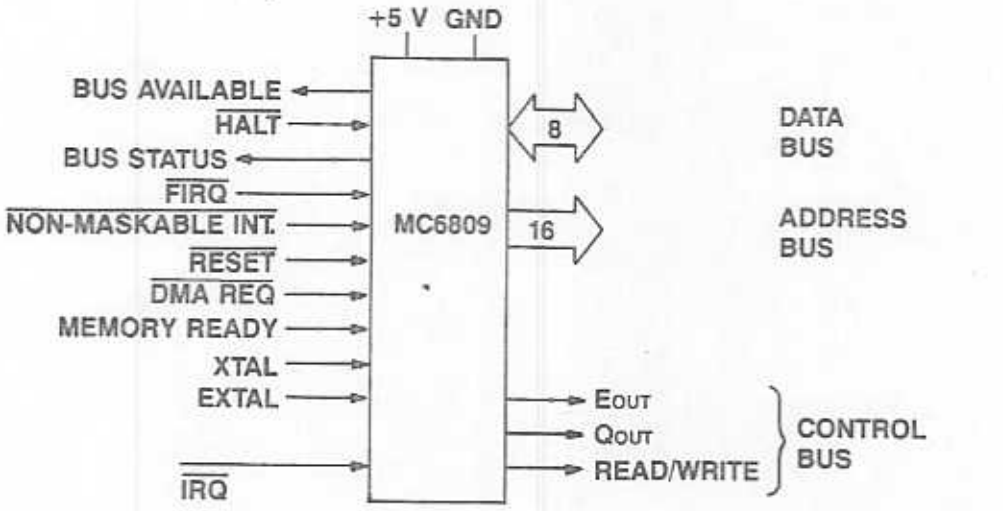


BITS SET AS A RESULT OF OPERATION!

ADDA	A = 1000 1000 M = 1000 1000	A + M = 1 0001 0000	; H = 1, Z = 0 C = 1, N = 0 V = 1
DECA	A = 0000 0001	A-1 = 0000 0000	; Z = 1, N = 0
LDA	#$80	A = 1000 0000	; N = 1, Z = 0, V = 0
COMA OR DECA	A = 1000 0000	$\overline{A}$ = 0111 1111	; N = 0, V = 0, C = 1, Z = 0 ; N = 0, V = 1, C = UNCHANGED, Z = 0
ADDA	A = 1000 0010 = -126_{10} M = 1000 0010 = -126_{10}	A + M = 1 0000 0100 = $+4_{10}$	; V = 1, Z = 0 C = 1, N = 0 H = 0

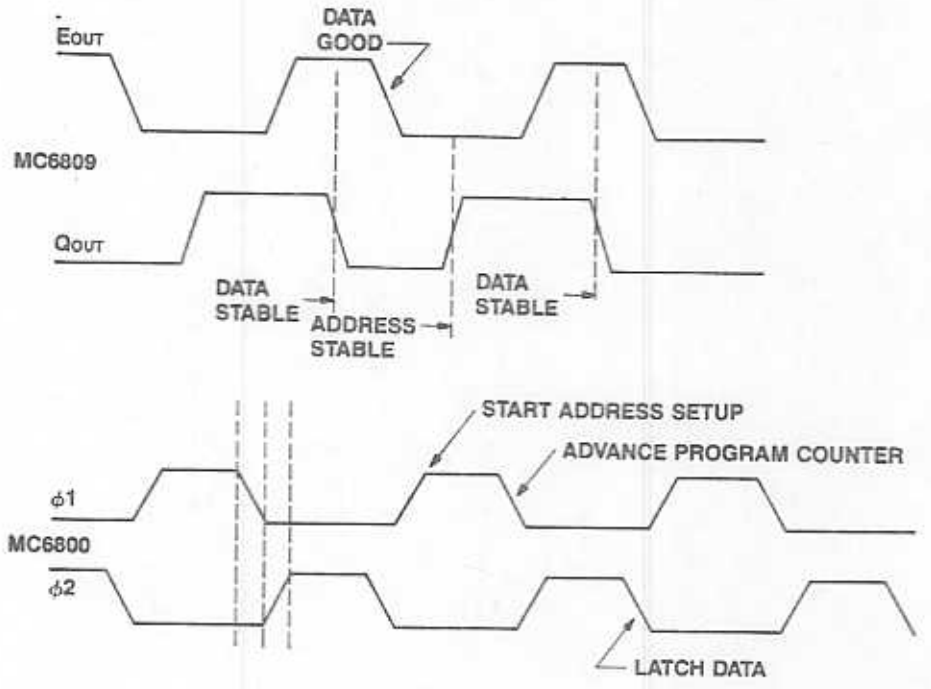
TR9123-1

MC6809 BUS & CONTROL SIGNALS

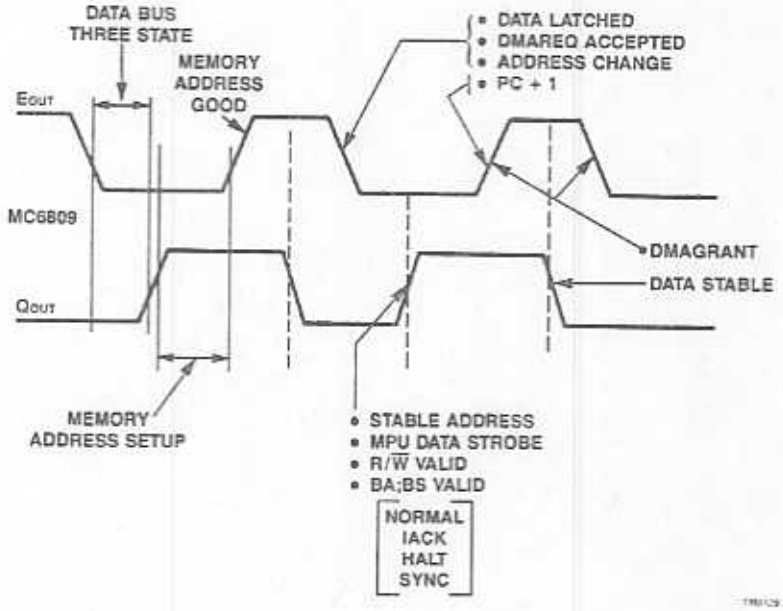


TR9124B-2

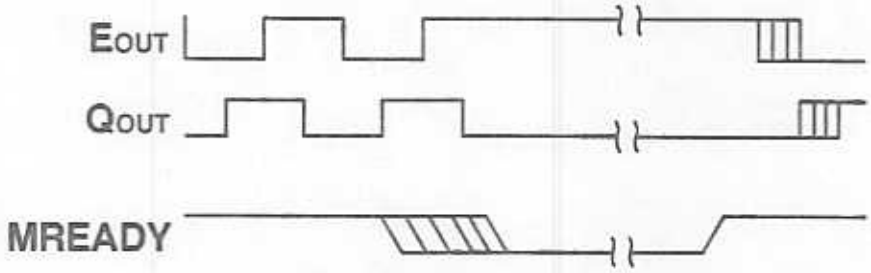
MPU CLOCK WAVE FORM



MPU CLOCK WAVE FORM



MC6809
MREADY
(MEMORY READY)



MREADY LOW $\leq 10\mu s$

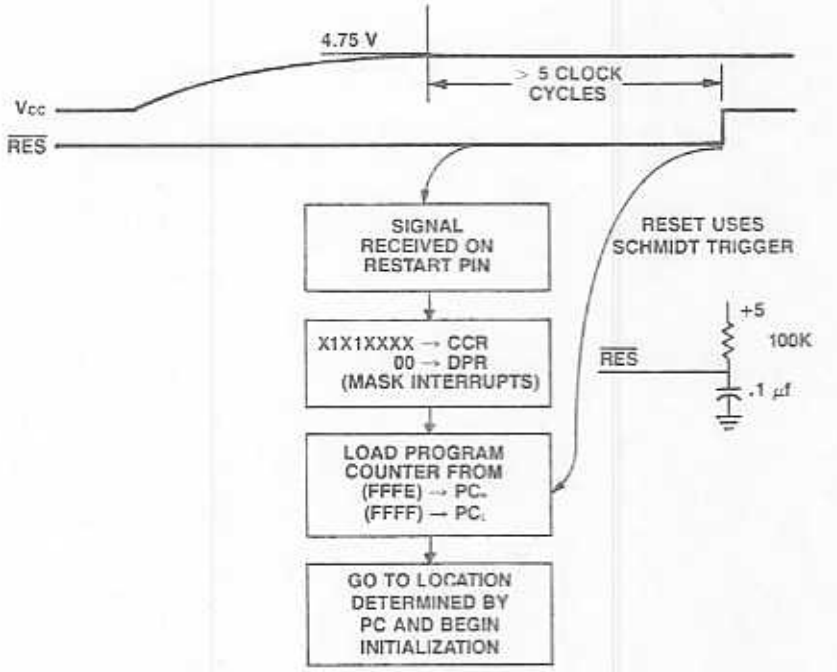
TR9125

MC6809
$\overline{\text{DMAREQ}}$

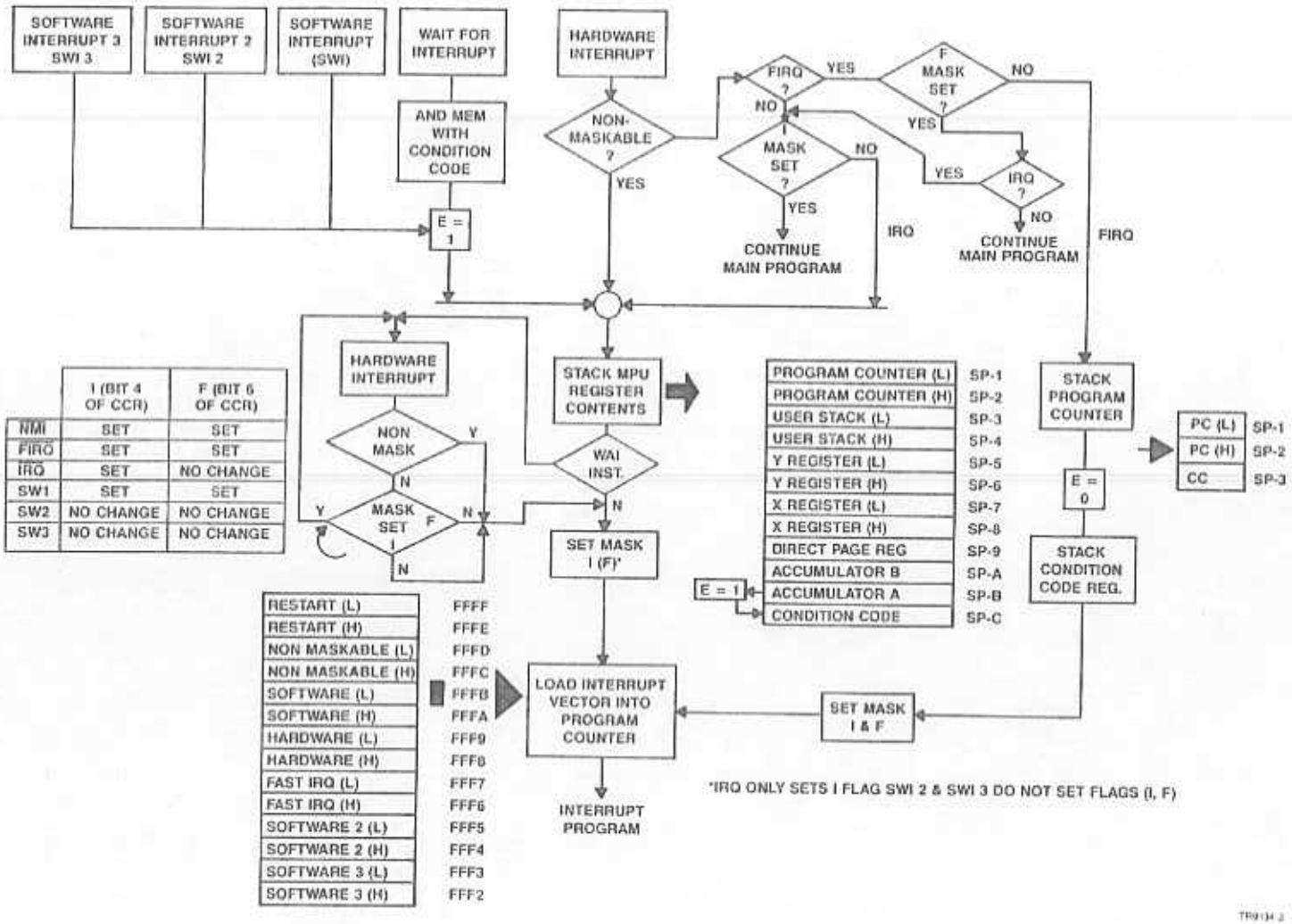
- STRETCHES INTERNAL CLOCK
- EOUT AND QOUT FREE RUN
- BA = 1; BS = 1
- ADDRESS BUS, DATA BUS, R/W GO THREE STATE
- EVERY 16 CYCLES 6809 WILL EXECUTE ONE CYCLE
- BA = 0 FOR THIS CYCLE. AUTOMATIC MPU REFRESH
- BS = 0

TR9183-2

RESTART SEQUENCE



INTERRUPT FLOW CHART MC6809

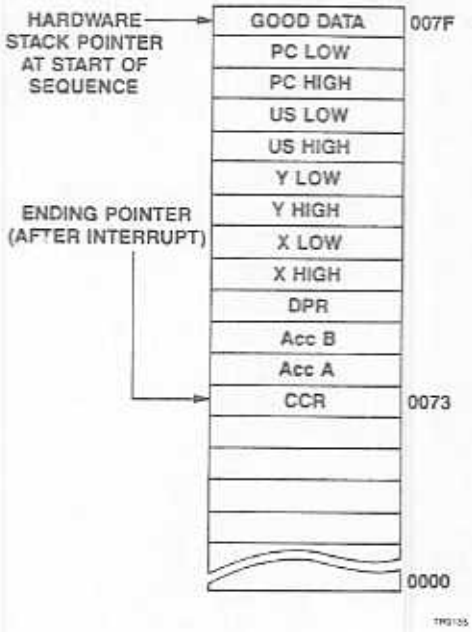


	I (BIT 4 OF CCR)	F (BIT 6 OF CCR)
NMI	SET	SET
FIRQ	SET	SET
IRQ	SET	NO CHANGE
SW1	SET	SET
SW2	NO CHANGE	NO CHANGE
SW3	NO CHANGE	NO CHANGE

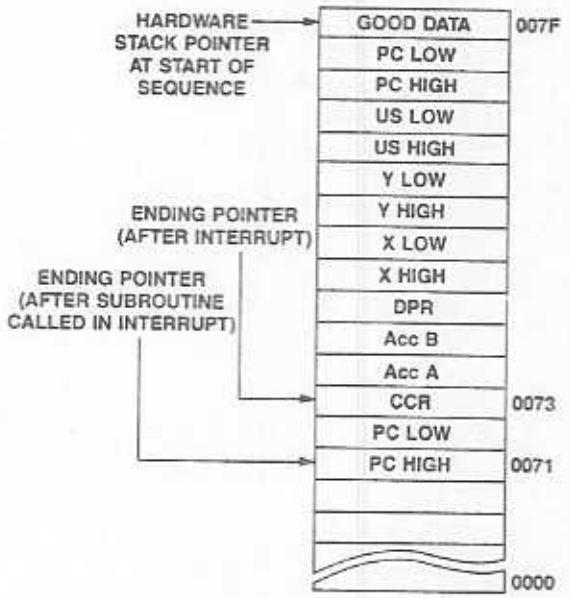
*IRQ ONLY SETS I FLAG SWI 2 & SWI 3 DO NOT SET FLAGS (I, F)

TR9134.2

RAM USED FOR STACK STORAGE
RAM ADDRESS $0000–$007F

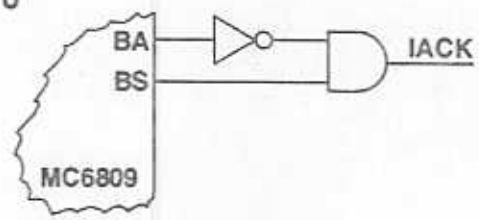


RAM USED FOR STACK STORAGE
RAM ADDRESS $0000–$007F

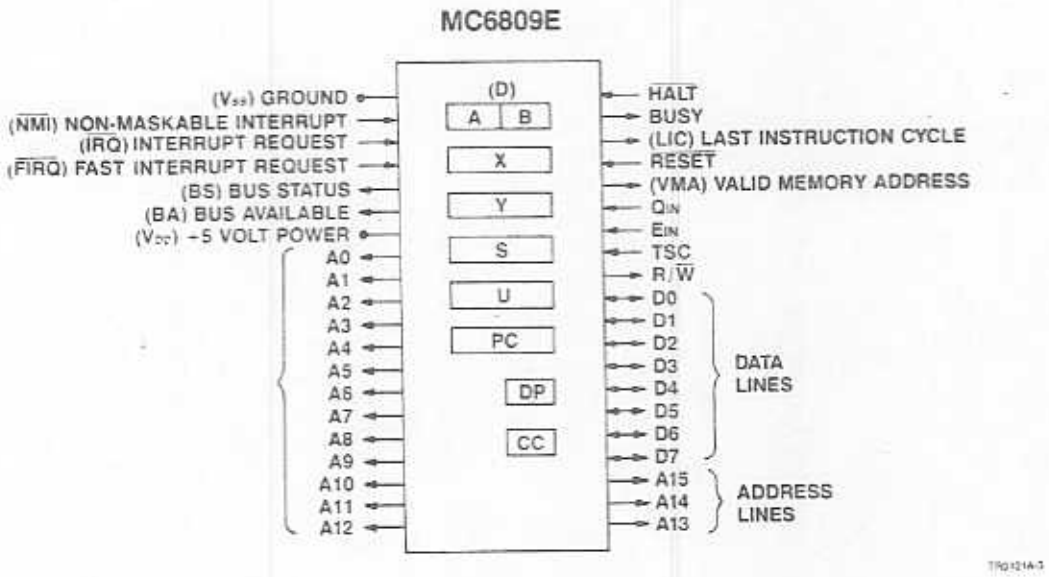


INTERRUPT ACKNOWLEDGE (IACK) FUNCTION

- IACK GOES HIGH TO INDICATE A VECTOR IS BEING FETCHED
- THIS SIGNAL COULD BE USED TO ACTIVATE ADDRESS DECODING LOGIC
- THEREFORE, GENERATING NEW VECTORS
- BS = 1; BA = 0

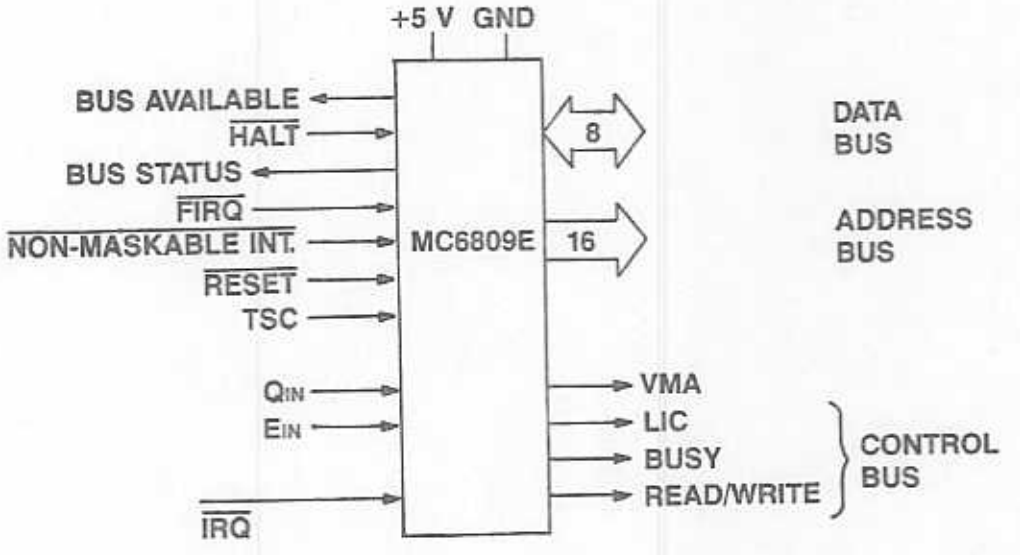


TR9131-1



TR6121A-1

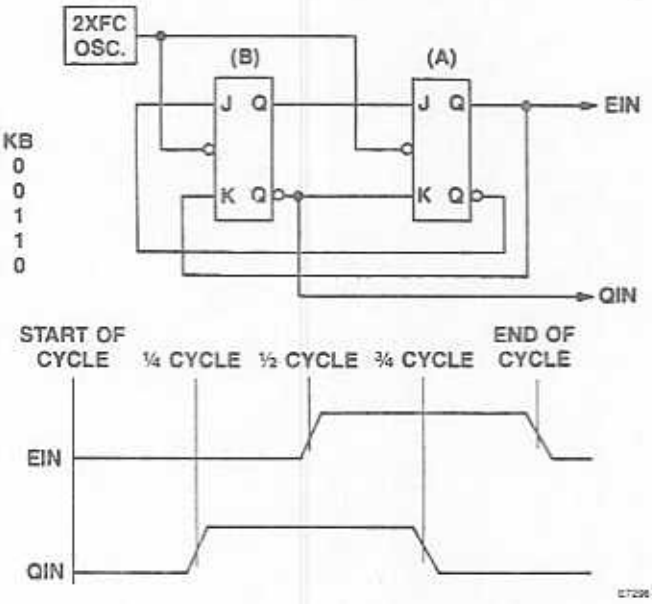
MC6809E BUS & CONTROL SIGNALS



TR6124A-2

BASIC EXTERNAL CLOCK GENERATOR FOR MC6809E

A	B	JA	KA	JB	KB
0	0	0	1	1	0
0	1	1	0	1	0
1	1	1	0	0	1
1	0	0	1	0	1
0	0	0	1	1	0



MC6809E TSC

- ADDRESS; R/W GO THREE STATE
- DATA BUS GOES THREE STATE
- BA = 0; BS = 0
- CLOCK MUST BE HELD LOW EXTERNALLY.

TR9126-1

MC6809E
NEW CONTROL LINES

LAST INSTRUCTION CYCLE (LIC)

- LIC GOES HIGH AT THE BEGINNING OF THE LAST INSTRUCTION CYCLE
- NEXT CYCLE WILL BE AN OP CODE FETCH
- USED TO INDICATE WHEN TO READ BUS FOR DIAGNOSTIC PURPOSES

TR9130-2

MC6809E
NEW CONTROL LINES
BUSY

- GOES HIGH DURING READ-MODIFY-WRITE INSTRUCTIONS
- GOES HIGH DURING DOUBLE BYTE OPERATIONS
- GOES HIGH DURING INDIRECT OPERATIONS
- BUSY = 0 FOR CWAI
  BUSY = 0 FOR HALT
- BUSY IS A COMMON BUS LOCKOUT SIGNAL

TR9129-1

SUMMARY
CONTROL LINE STATES

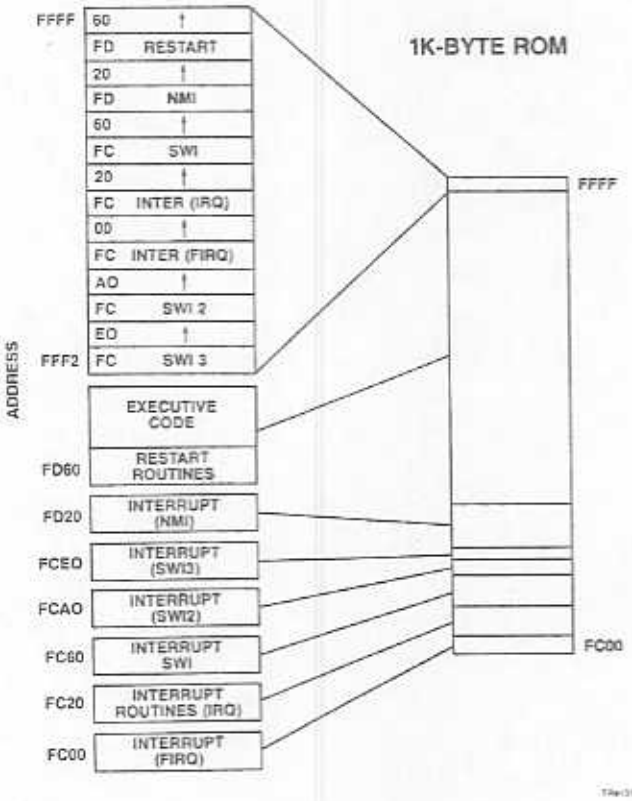
	HALT	$\overline{\text{DMA REQ}}$*	CWAI	TSC	SYNC	MRDY
BA	1	1	0	0	1	0
LIC	1	NA	1	1/0	1	NA
BS	1	1	0	0	0	0
BUSY	0	NA	0	0	0	NA
ENABLE OUT	⎍	⎍	⎍	NA	⎍	1
QUAD OUT	⎍	⎍	⎍	NA	⎍	0
VMA	0	NA	0	0	0	NA
DATA BUS	TS	TS	ACTIVE	TS	TS	ACTIVE
ADD BUS R/W	TS	TS	ACTIVE	TS	TS	ACTIVE

*EVERY 16 CYCLES, $\overline{\text{DMAREQ}}$ AUTOMATICALLY REFRESHES 6809, BA = 0

STATUS SIGNALS

BA	BS	FUNCTION
0	0	NORMAL OPERATION
0	1	INTERRUPT ACKNOWLEDGE
1	0	SYNC ACKNOWLEDGE
1	1	DMA GRANT OR HALT ACKNOWLEDGE

TR9138-1



M6809

59 INSTRUCTIONS

9 ADDRESSING MODES

TR9036-1

ARITHMETIC INSTRUCTIONS

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
ADD	ADD ACCUMULATOR TO MEMORY	ADDA ADDB	A + (M) → A B + (M) → B
	ADD DOUBLE ACCUMULATOR TO MEMORY	ADDD	D_H + (M) → D_H D_L + (M + 1) → D_L
ADC	ADD ACCUMULATOR WITH CARRY TO MEMORY	ADCA ADCB	A + (M) + C → A B + (M) + C → B
ABX	ADD B ACCUMULATOR TO THE X REGISTER	ABX	B + X_L → X_L 0 + C + X_H → X_H
DA	DECIMAL ADDITION ADJUST	DAA	CONVERTS BINARY ADD. OF BCD CHARACTERS INTO BCD FORMAT
SUB	SUBTRACT MEMORY FROM ACCUMULATOR	SUBA SUBB	A − (M) → A B − (M) → B
	SUBTRACT MEMORY FROM DOUBLE ACCUMULATORS	SBBD	D_L − (M + 1) → D_L D_H − M → D_H
SBC	SUBTRACT MEMORY FROM ACCUMULATOR WITH CARRY	SBCA SBCB	A − (M) − C → A B − (M) − C → B
SEX	EXTEND THE SIGN OF B INTO A	SEX	Acc B BIT_7 = 1 A = FF Acc B BIT_7 = 0 A = 0

TR9037-1

DATA HANDLING INSTRUCTIONS (ALTER DATA)

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
CLR	CLEAR MEMORY CLEAR Acc A CLEAR Acc B	CLR CLRA CLRB	00 → M 00 → A 00 → B
DEC	DECREMENT MEMORY DECREMENT Acc A DECREMENT Acc B	DEC DECA DECB	(M) − 1 → M A − 1 → A B − 1 → B
INC	INCREMENT MEMORY INCREMENT Acc A INCREMENT Acc B	INC INCA DECB	(M) + 1 → M A + 1 → A B + 1 → B
COM	COMPLEMENT MEMORY COMPLEMENT Acc A COMPLEMENT Acc B	COM COMA COMB	$\overline{(M)}$ → M $\overline{A}$ → A $\overline{B}$ → B
NEG	2'S COMPLEMENT MEMORY 2'S COMPLEMENT Acc A 2'S COMPLEMENT Acc B	NEG NEGA NEGB	00 − (M) → M 00 − A → A 00 − B → B

TR9038-1

DATA HANDLING INSTRUCTIONS (DATA MOVEMENT)

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
LD	LOAD Acc. LOAD 16 BIT REG	LDA LDB LDD LDX LDY LDS LDU	(M) → A (M) → B (M) → R_H (M + 1) → R_L
ST	STORE Acc STORE 16 BIT REG	STA STB STD STX STY STS STU	A → M B → M R_H → M R_L → M + 1

TR9039

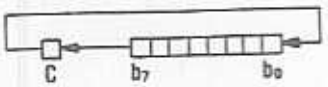
DATA HANDLING INSTRUCTIONS (DATA MOVEMENT)

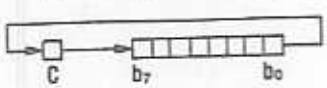
INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
PSH	PUSH REGISTER(S) TO HARDWARE STACK PUSH REGISTER(S) TO USER STACK	PSHS PSHU	$SP - 1 \rightarrow SP$ $R \rightarrow M_{SP}$ $USP - 1 \rightarrow USP$ $R \rightarrow M_{USP}$
PUL	PULL REGISTER(S) FROM HARDWARE STOCK PULL REGISTER(S) FROM USER STACK	PULS PULU	$(M_{SP}) \rightarrow R$ $SP + 1 \rightarrow SP$ $(M_{USP}) \rightarrow R$ $USP + 1 \rightarrow USP$
TFR	TRANSFER REGISTER TO REGISTER	TFR	$R_1 \rightarrow R_2$
EXG	EXCHANGE REGISTERS	EXG	$R_1 \leftrightarrow R_2$

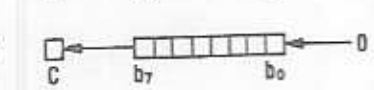
TR9040-1

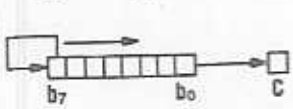
DATA HANDLING INSTRUCTIONS (SHIFT AND ROTATE)

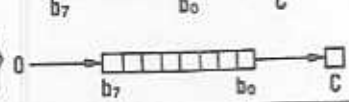
INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
ROL	ROTATE LEFT	ROL ROLA ROLB	M A B
ROR	ROTATE RIGHT	ROR RORA RORB	M A B
LSL (ASL)	LOGICAL SHIFT LEFT	LSL LSLA LSLB	M A B
ASR	ARITHMETIC SHIFT RIGHT	ASR ASRA ASRB	M A B
LSR	LOGICAL SHIFT RIGHT	LSR LSRA LSRB	M A B











TR9041

LOGIC INSTRUCTIONS

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
AND	AND ACCUMULATOR WITH MEMORY	ANDA ANDB	A • (M) → A B • (M) → B
EOR	EXCLUSIVE OR ACCUMULATOR WITH MEMORY	EORA EORB	A ⊕ (M) → A B ⊕ (M) → B
OR	INCLUSIVE OR ACCUMULATOR WITH MEMORY	ORA ORA	A + (M) → A B + (M) → B

TR9042-1

DATA TEST INSTRUCTIONS

INSTRUCTION	FUNCTION	MNEMONIC	TEST
BIT	BIT TEST (AND) ACCUMULATOR WITH MEMORY	BITA BITB	A • (M) B • (M)
CMP	COMPARE (SUBTRACT) MEMORY WITH ACCUMULATOR	CMPA CMPB	A − (M) B − (M)
	COMPARE (SUBTRACT) MEMORY WITH 16 BIT REGISTER	CMPD CMPX CMPY CMPU CMPS	R_L − (M + 1) R_H − (M_H)
TST	TEST MEMORY FOR ZERO OR MINUS TEST ACCUMULATOR FOR ZERO OR MINUS	TST TSTA TSTB	(M) − 00 A − 00 B − 00

TR9044-2

BRANCH INSTRUCTIONS

INSTRUCTION	FUNCTION	MNEMONIC	BRANCH TEST
BRA	BRANCH ALWAYS	BRA	NONE
BRN	BRANCH NEVER	BRN	NONE
BCC (BHS)	BRANCH IF CARRY CLEAR (HIGHER OR SAME)	BCC	C = 0
BCS (BLO)	BRANCH IF CARRY SET (IF LOWER)	BCS	C = 1
BEQ	BRANCH IF = ZERO	BEQ	Z = 1
BNE	BRANCH IF NOT EQUAL ZERO	BNE	Z = 0
BGE	BRANCH IF ≥ ZERO*	BGE	(N ⊕ V) = 0
BLT	BRANCH IF < ZERO*	BLT	(N ⊕ V) = 1
BGT	BRANCH IF > ZERO*	BGT	Z + (N ⊕ V) = 0
BLE	BRANCH IF ≤ ZERO*	BLE	Z + (N ⊕ V) = 1
BHI	BRANCH IF HIGHER	BHI	(C + Z) = 0
BLS	BRANCH IF LOWER OR SAME	BLS	(C + Z) = 1
BMI	BRANCH IF MINUS	BMI	N = 1
BPL	BRANCH IF PLUS	BPL	N = 0

*SIGNED DATA ONLY!

TP9045

JUMP AND BRANCH INSTRUCTIONS

INSTRUCTION	FUNCTION	MNEMONIC	BRANCH TEST/OPERATION
BVC	BRANCH IF OVERFLOW CLEAR	BVC	V = 0
BVS	BRANCH IF OVERFLOW SET	BVS	V = 1
BSR	BRANCH TO SUBROUTINE	BSR	NONE
JMP	JUMP TO MEMORY ADDRESS	JMP	(M) — PC$_H$; (M + 1) → PC$_L$
JSR	JUMP TO SUBROUTINE MEMORY ADDRESS	JSR	SP — 1 — SP; PC$_L$ — M$_{SP}$ SP — 1 → SP; PC$_H$ → M$_{SP}$ (M) → PC$_H$; (M + 1) → PC$_L$
RTS	RETURN FROM SUBROUTINE	RTS	(M$_{SP}$) → PC$_H$; SP + 1 → SP (M$_{SP}$) → PC$_L$; SP + 1 → SP

CONDITION CODE REGISTER INSTRUCTION

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
OR	SET ANY CONDITION CODE FLAG	ORCC	CC + (M) → CC E F H I N Z V C 0 0 0 0 0 0 0 0 + M = 0 0 0 0 1 1 1 1 E F H I N Z V C 0 0 0 0 1 1 1 1
AND	CLEAR ANY CONDITION CODE FLAG	ANDCC	CC • (M) → CC E F H I N Z V C 0 0 0 0 1 1 1 1 M = 1 1 1 1 0 0 0 0 E F H I N Z V C 0 0 0 0 0 0 0 0

TR9047-1

INTERRUPT HANDLING INSTRUCTION

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
RTI	RETURN FROM INTERRUPT	RTI	$(M_{SP}) \rightarrow CC$ SP + 1 → SP IF E = 1 $(M_{SP}) \rightarrow$ REGS SP + B → SP IF E = 0 $(M_{SP}) \rightarrow P_C$ SP + 1 → SP $M_{SP} \rightarrow P_C$ SP + 1 → SP
CWAI	CLEAR AND WAIT FOR INTERRUPT	CWAI	CC • M E = 1 REGS → (M_{SP}) SP − C → SP
SYNC	SYNCHRONIZE TO INTERRUPT	SYNC	STOPS MPU UNTIL INTERRUPT OCCURS THEN CONTINUES

TR9048-2

INTERRUPT HANDLING INSTRUCTION

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
SWI	SOFTWARE INTERRUPT	SWI	E = 1 REGS → M_{SP} SP − C → SP I = 1 F = 1 (FFFA) → P_{CH} (FFFB) → P_{CL}
		SWI2	E = 1 REGS → M_{SP} SP − C → SP (FFF4) = P_{CH} (FFF5) = P_{CL}
		SWI3	E = 1 REGS → M_{SP} SP − C → SP (FFF2) = P_{CH} (FFF3) = P_{CL}

TR9049-1

POINTER REGISTER INSTRUCTIONS

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
LEA	LOAD POINTER REGISTER WITH EFFECTIVE ADDRESS	LEAX LEAY LEAS LEAU	EA → X EA → Y EA → S EA → U

NOTE ONLY USES INDEXED ADDRESSING MODE

TR9050

MISCELLANEOUS INSTRUCTION

INSTRUCTION	FUNCTION	MNEMONIC	OPERATION
MUL	MULTIPLY ACCUMULATORS	MUL	A * B → D
NOP	NO OPERATION	NOP	PC + 1 → PC

TR9051

INPUT/OUTPUT INSTRUCTIONS STILL

NONE!

TR9052

6809
EXECUTABLE INSTRUCTIONS — ALPHABETIC LIST

ABX	ADD ACCB INTO X-REG
ADC	ADD WITH CARRY
ADD	ADD
AND	LOGICAL AND
ASL	ARITHMETIC SHIFT LEFT (LSL)
ASR	ARITHMETIC SHIFT RIGHT
BCC	BRANCH IF CARRY CLEAR
BCS	BRANCH IF CARRY SET
BEQ	BRANCH IF EQUAL TO ZERO
BGE	BRANCH IF GREATER OR EQUAL ZERO
BGT	BRANCH IF GREATER THAN ZERO
BHI	BRANCH IF HIGHER
BIT	BIT TEST
BLE	BRANCH IF LESS OR EQUAL
BLS	BRANCH IF LOWER OR SAME
BLT	BRANCH IF LESS THAN ZERO
BMI	BRANCH IF MINUS
BNE	BRANCH IF NOT EQUAL TO ZERO
BPL	BRANCH IF PLUS
BRA	BRANCH ALWAYS
BRN	BRANCH NEVER
BSR	BRANCH TO SUBROUTINE
BVC	BRANCH IF OVERFLOW CLEAR
BVS	BRANCH IF OVERFLOW SET
CLR	CLEAR
CMP	COMPARE
COM	COMPLEMENT
CWAI	CLEAR AND WAIT FOR INT.
DAA	DECIMAL ADJUST
DEC	DECREMENT
EOR	EXCLUSIVE OR
EXG	EXCHANGE REGISTERS
INC	INCREMENT
JMP	JUMP
JSR	JUMP TO SUBROUTINE
LD	LOAD ACCUMULATORS
LD	LOAD POINTER REGISTERS
LEA	LOAD EFFECTIVE ADDRESS
LSR	LOGICAL SHIFT RIGHT
MUL	MULTIPLY ACCUMULATORS
NEG	NEGATE
NOP	NO OPERATION
OR	INCLUSIVE OR ACCUMULATOR/CONDITION CODE REGISTER
PSH	PUSH DATA TO HARDWARE OR USER STACK
PUL	PULL DATA FROM HARDWARE OR USER STACK
ROL	ROTATE LEFT
ROR	ROTATE RIGHT
RTI	RETURN FROM INTERRUPT
RTS	RETURN FROM SUBROUTINE
SBC	SUBTRACT WITH CARRY
SEX	SIGN EXTENDED
ST	STORE ACCUMULATORS
ST	STORE POINTER REGISTER
SUB	SUBTRACT
SWI	SOFTWARE INTERRUPT
SWI2	SOFTWARE INTERRUPT
SWI3	SOFTWARE INTERRUPT
SYNC	SYNCHRONIZE TO INTERRUPT
TFR	TRANSFER REGISTERS
TST	TEST FOR ZERO

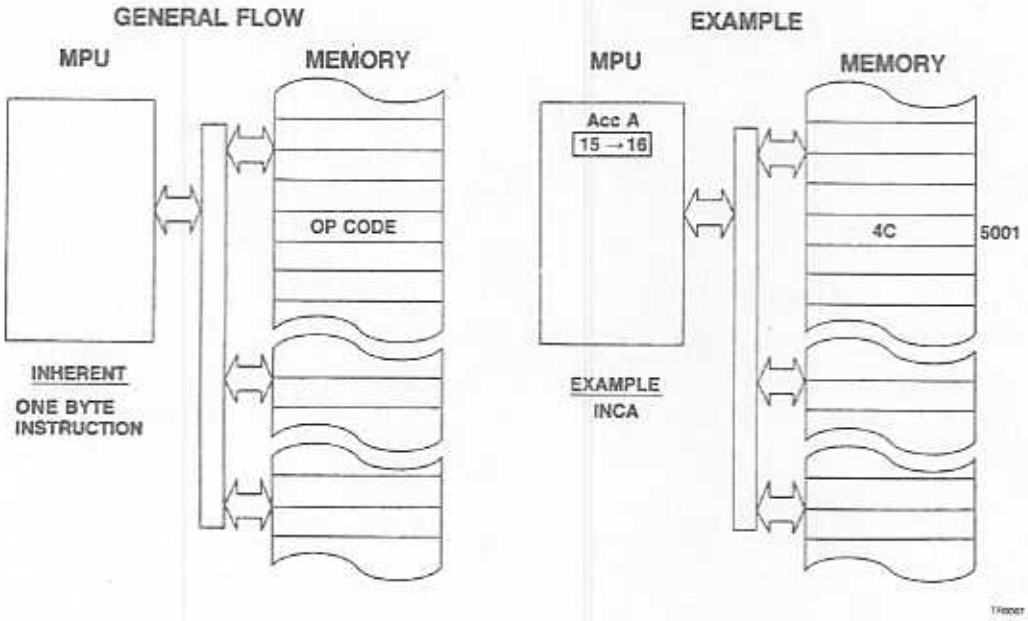
TR9117

6809
ADDRESSING MODES

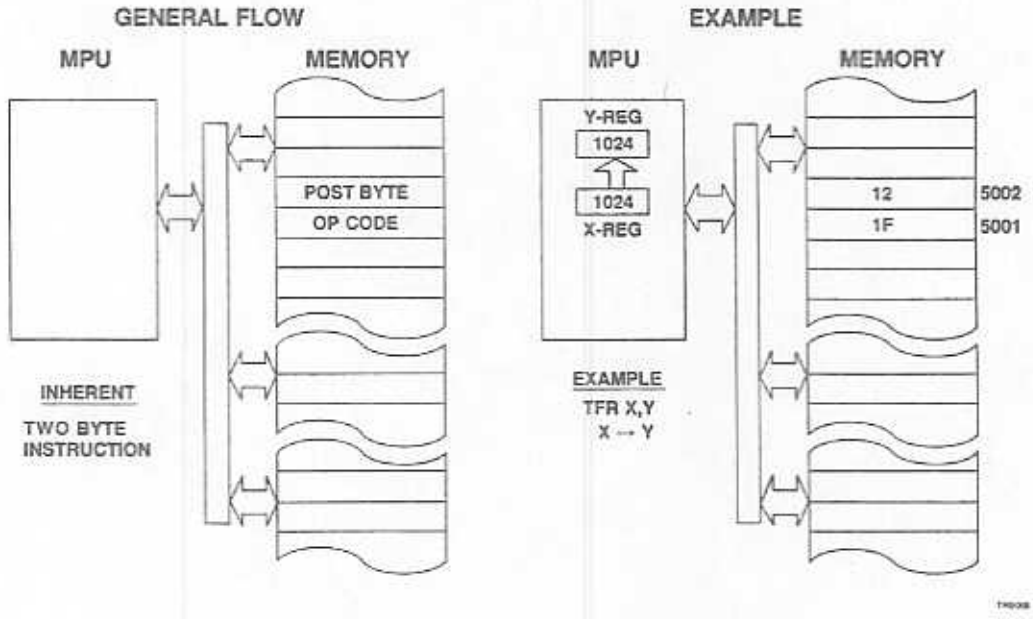
- IMMEDIATE
- INHERENT
- DIRECT
- EXTENDED
- INDEXED
- RELATIVE
- EXTENDED INDIRECT
- INDEXED INDIRECT
- LONG RELATIVE

TR9118

INHERENT ADDRESSING



INHERENT ADDRESSING



SPECIAL CASES OF INHERENT ADDRESSING MODE

TR9116

- **EXCHANGE, TRANSFER REGISTERS**

ExG X,Y

1E 12 (X ↔ Y)

OPCODE REGISTER

POST BYTE

HEX CHAR.	HEX CHAR.
REG.1	REG. 2

HEX CHAR.	REG.
0	D
1	X
2	Y
3	U
4	S
5	PC
8	A
9	B
A	CC
B	DP

TR9114-2

• PUSH AND PULL

PULS A, B, X

35 16 (0 0 0 1 0 1 1 0)

OPCODE REGISTERS

PULS #STACK

POST BYTE

BIT POSITION

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
PC	U/S	Y	X	DP	B	A	CC

REGISTERS PUSHED OR PULLED

TR9115

• CWAI

CWAI #$FF

3C OPCODE FF POST BYTE

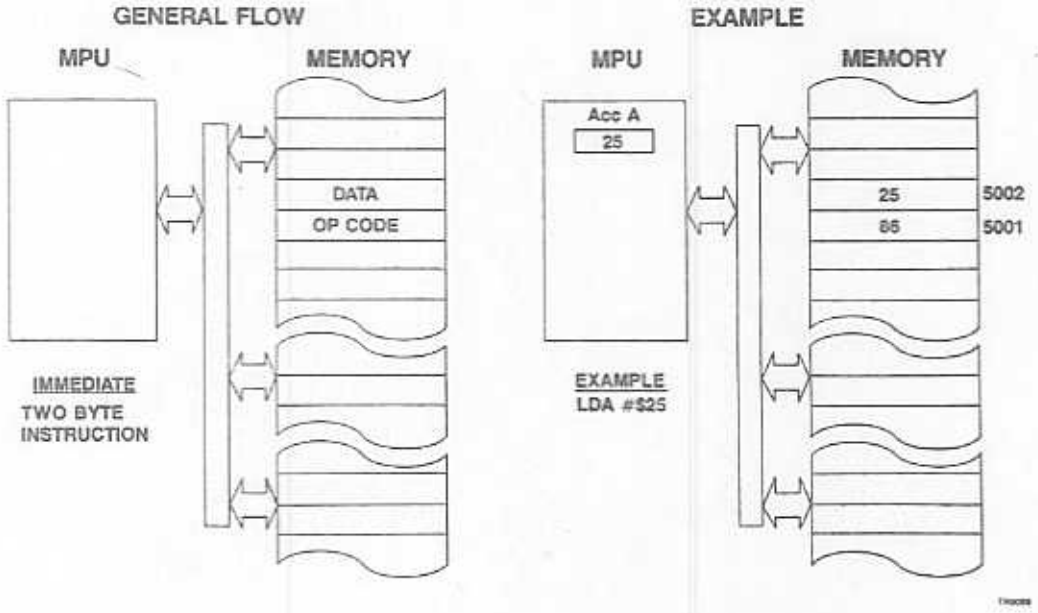
CONDITION CODE REG.

FF • E F H I N Z V C

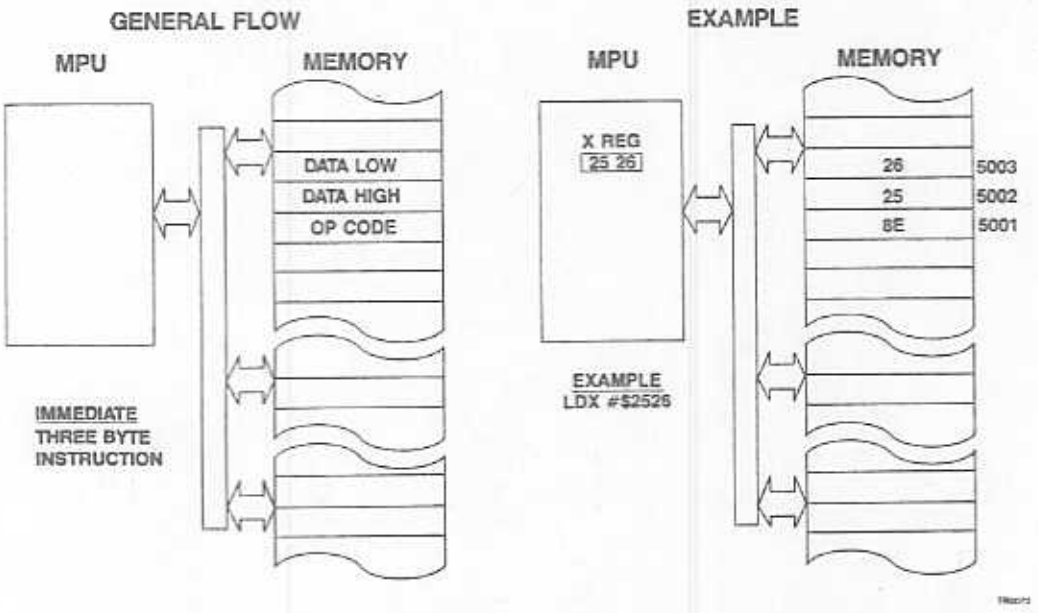
CWAI #MASKI

TR9113-1

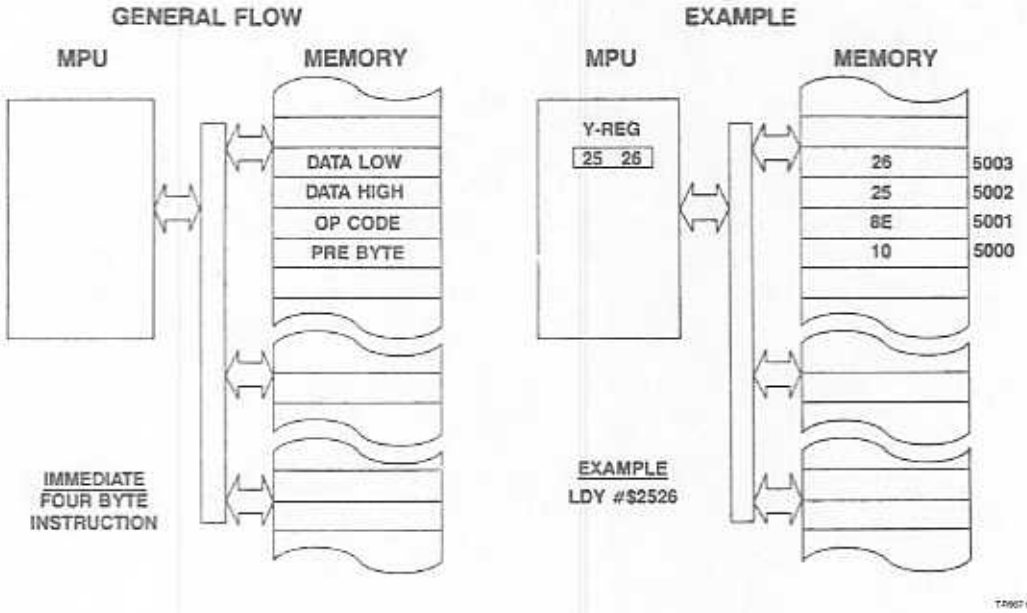
IMMEDIATE ADDRESSING



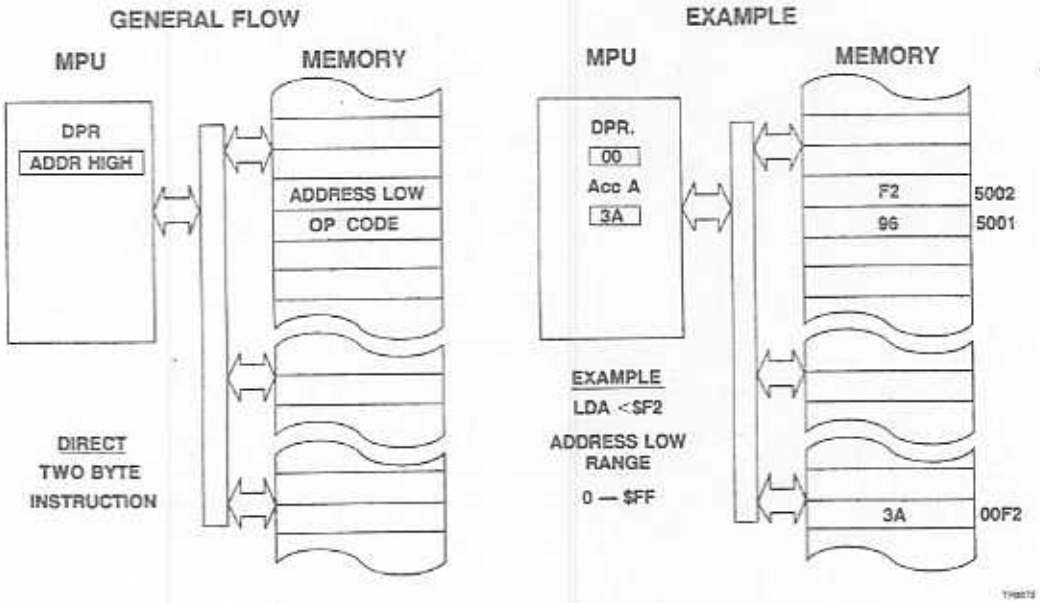
IMMEDIATE ADDRESSING



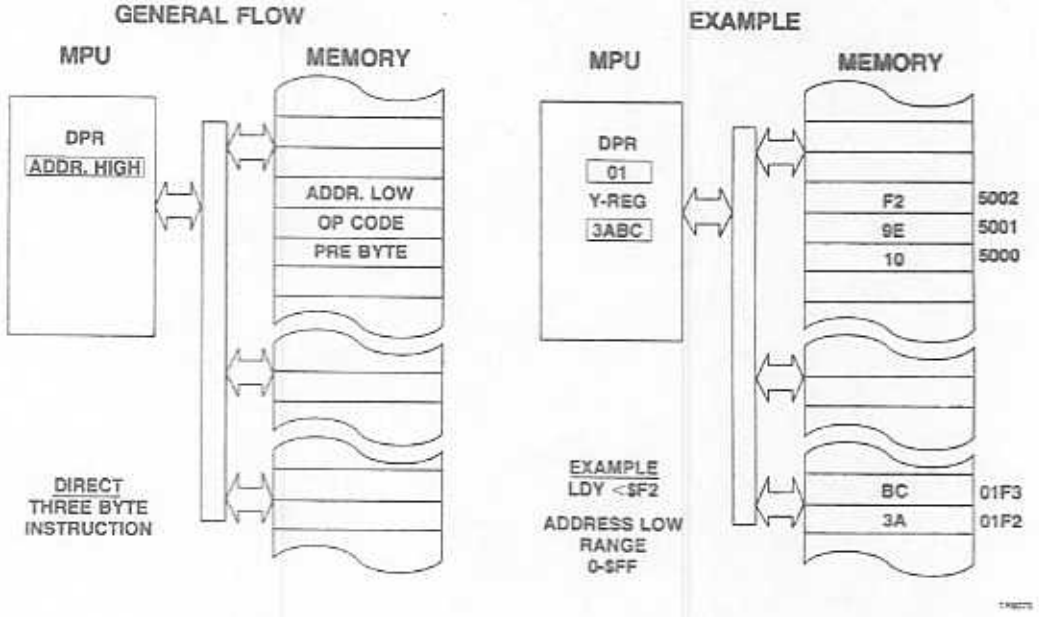
IMMEDIATE ADDRESSING



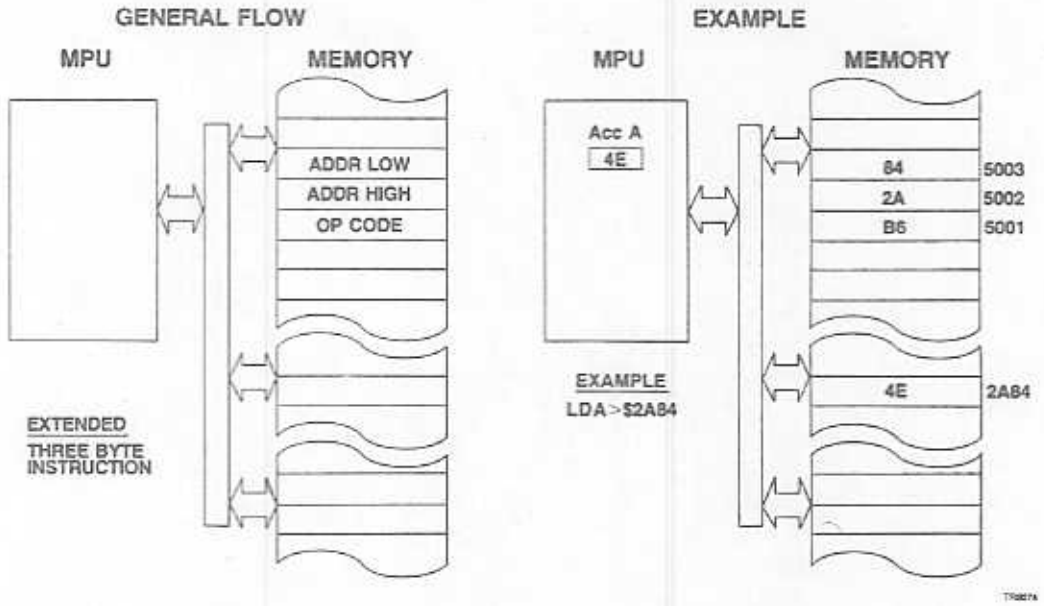
DIRECT ADDRESSING



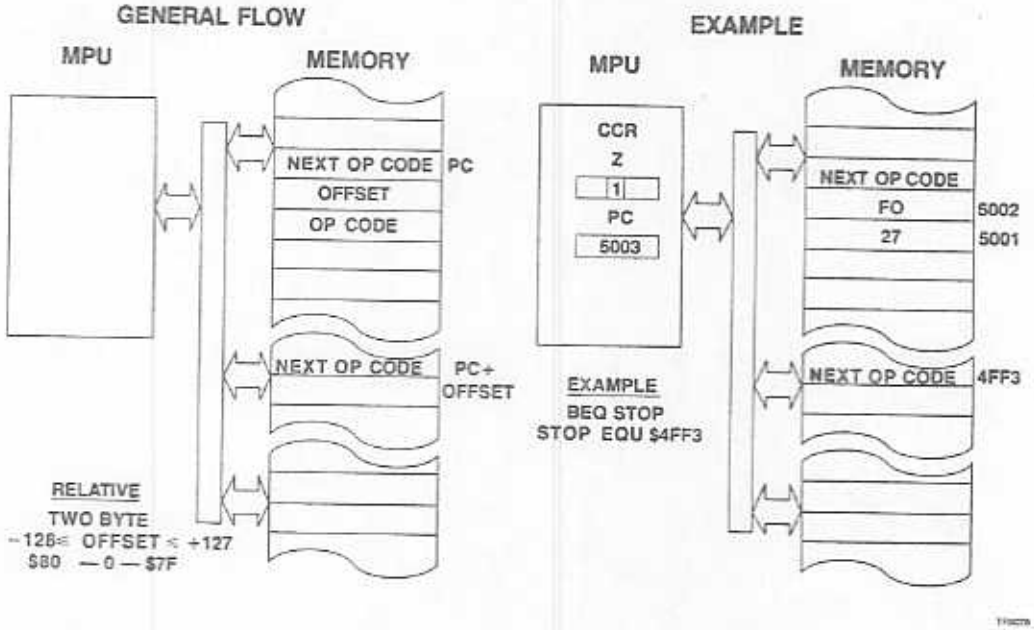
DIRECT ADDRESSING



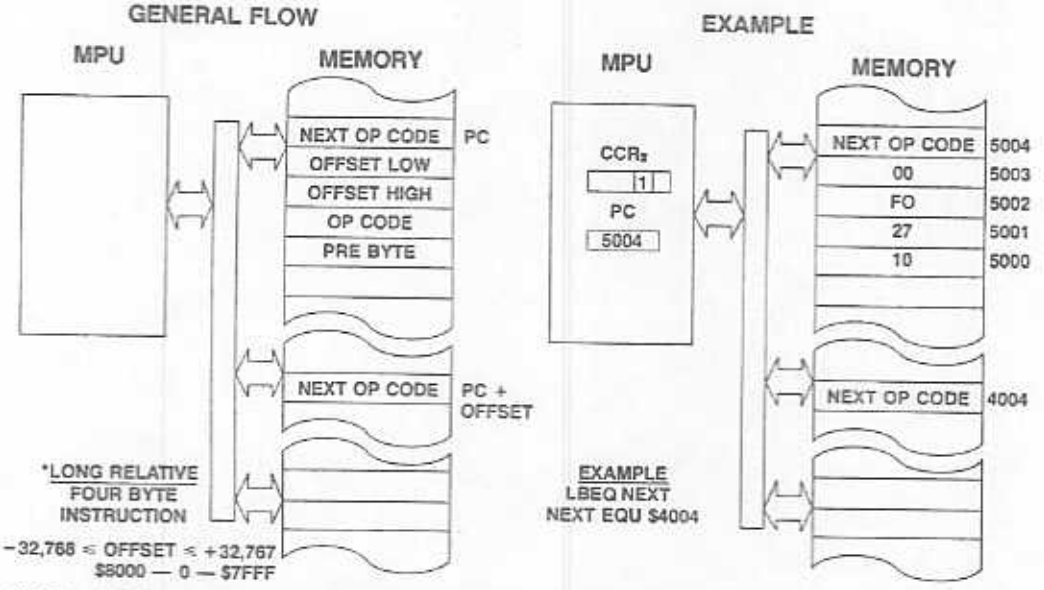
EXTENDED ADDRESSING



RELATIVE ADDRESSING

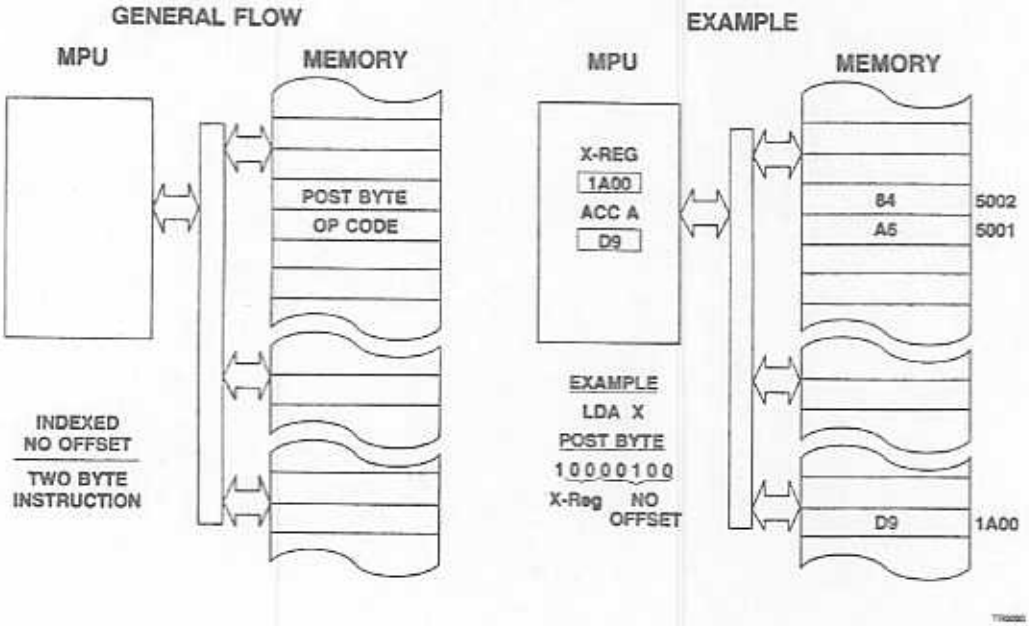


LONG RELATIVE ADDRESSING

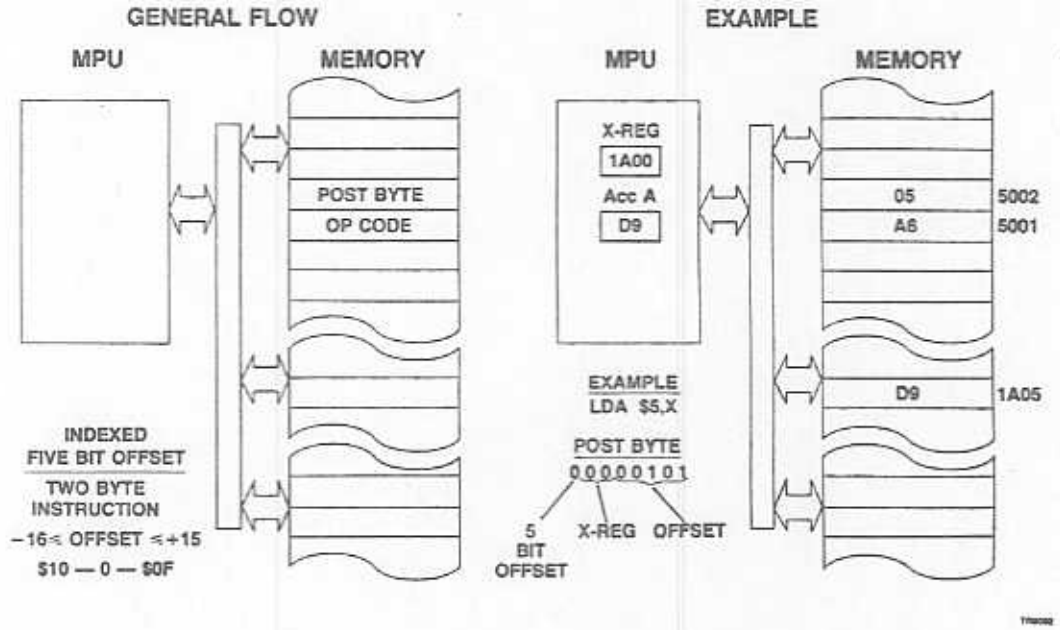


*LBRA and LBSR have their own op code; therefore, these instructions require no pre-byte and are 3 bytes long.

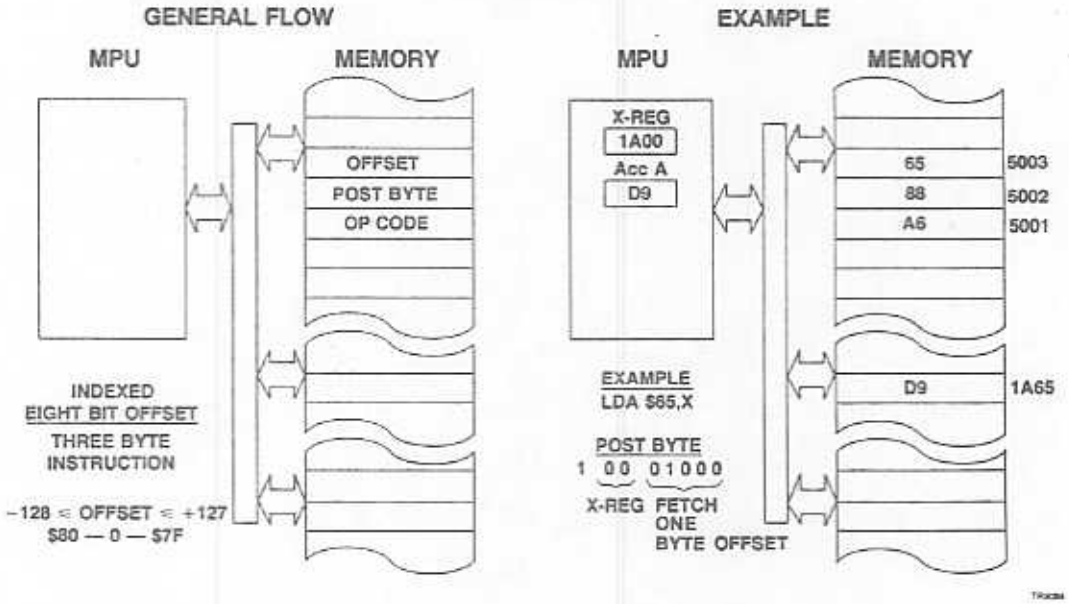
INDEXED ADDRESSING
(NO OFFSET)



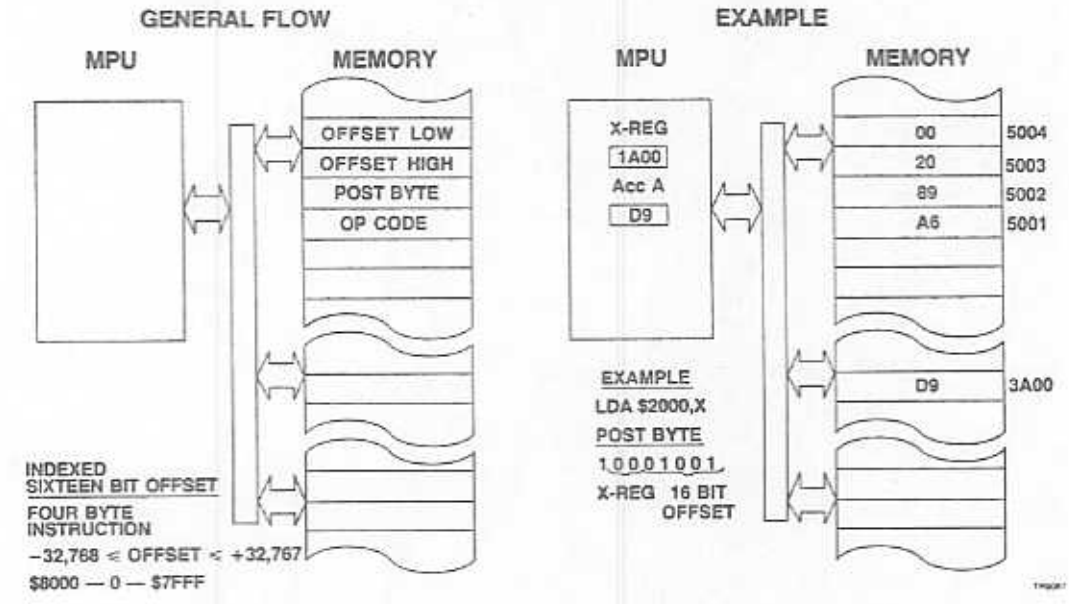
INDEXED ADDRESSING
(5 BIT OFFSET)



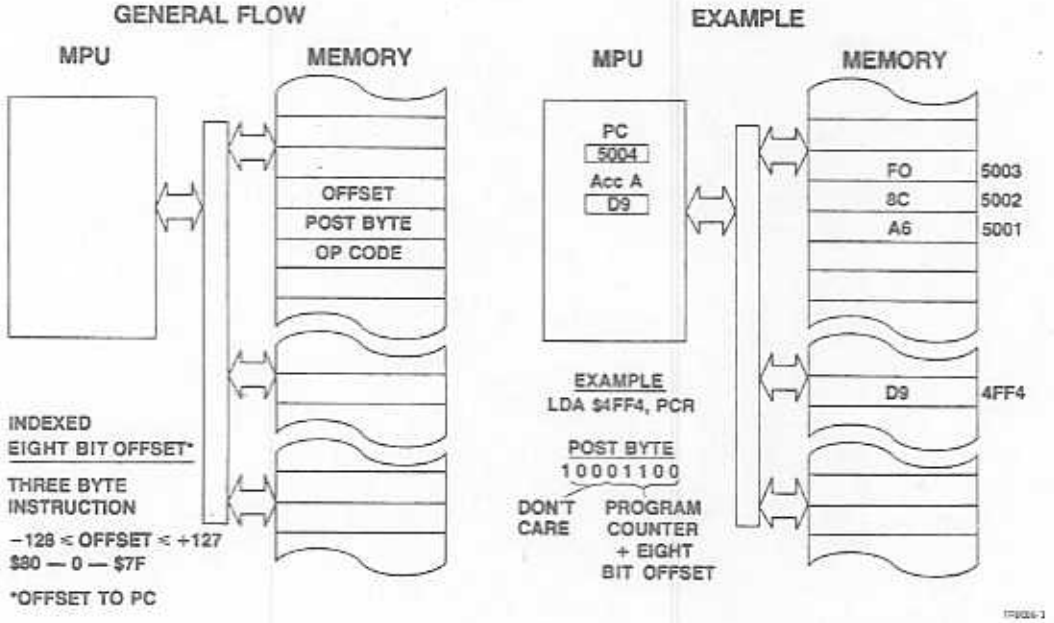
INDEXED ADDRESSING
(8-BIT OFFSET)



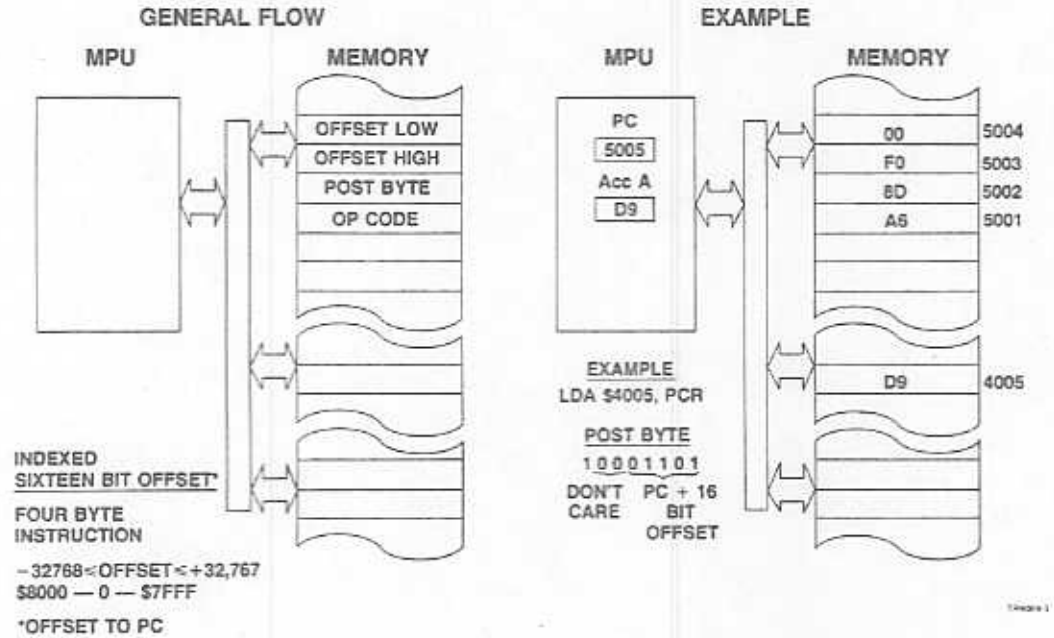
INDEXED ADDRESSING
(16-BIT OFFSET)



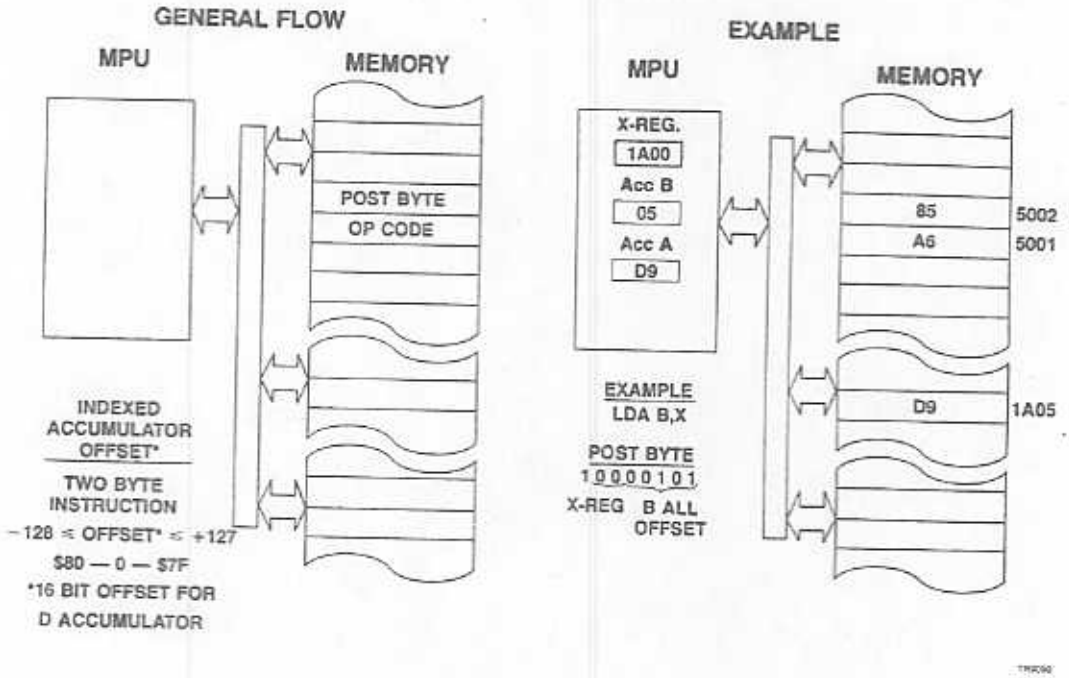
INDEXED ADDRESSING
(8-BIT OFFSET TO PC)



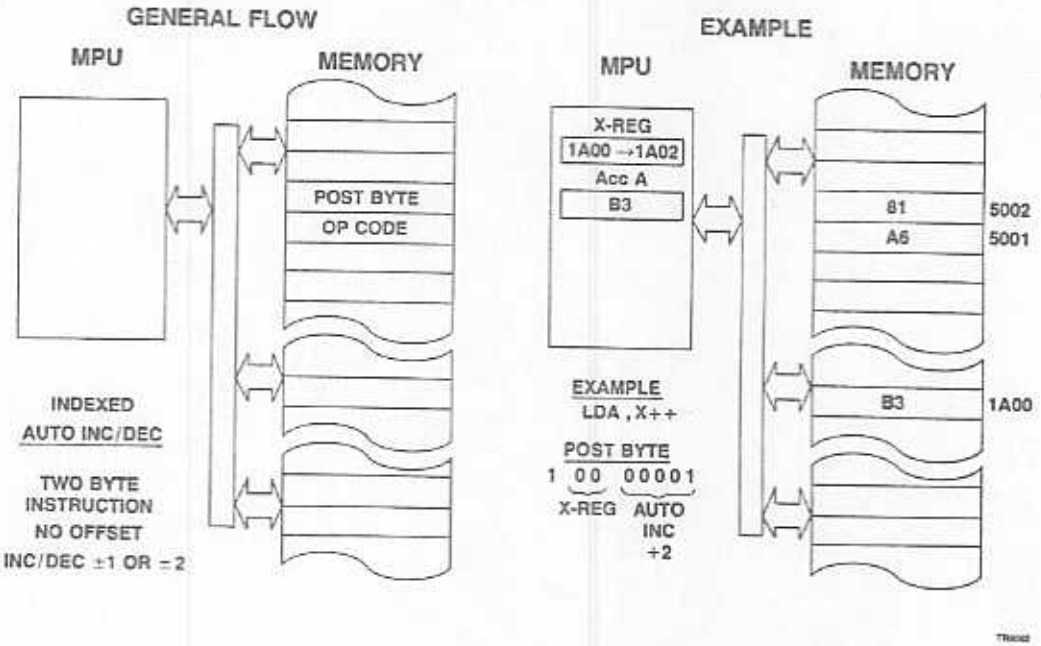
INDEXED ADDRESSING
(16-BIT OFFSET TO PC)



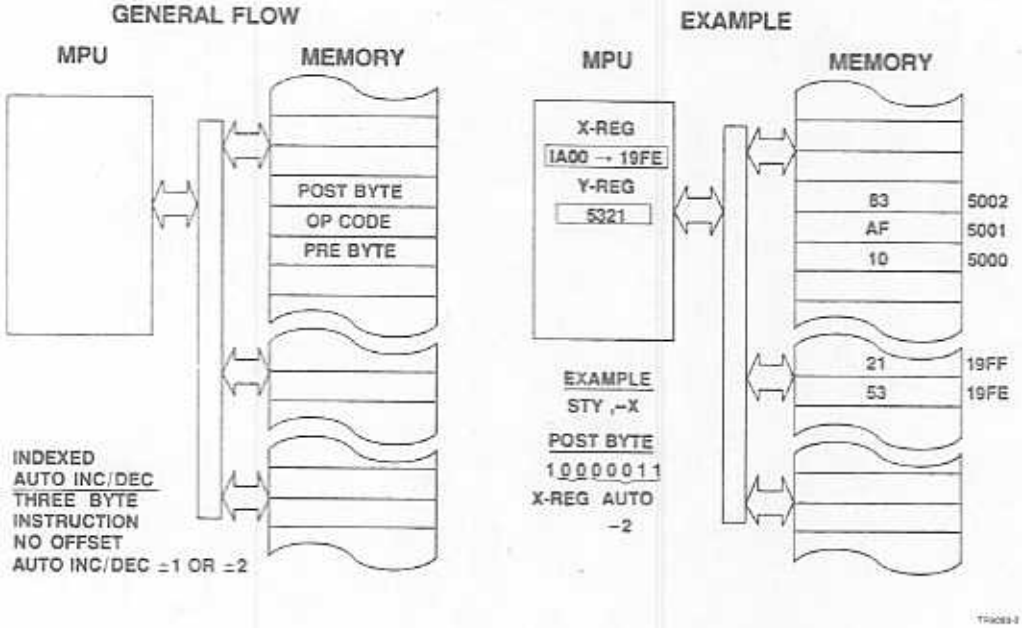
INDEXED ADDRESSING
(ACCUMULATOR OFFSET A, B OR D)



INDEXED ADDRESSING
(AUTO INC/DEC)



INDEXED ADDRESSING
(AUTO INC/DEC)



VALID INDEX MODE COMBINATIONS

OPERAND REGISTER / OFFSET	X	Y	S	U	PCR	– – X – X X+ X+ +	– – Y – Y Y+ Y+ +	– – S – S S+ S+ +	– – U – U U+ U+ +
0									
CONSTANT + OR – UP TO 16 BITS					SEE NOTE 3				
A									
B									
D									

1. OPERATION REGISTER MAY BE A, B, D, X, Y, S, OR U.
2. SHADED AREAS ARE INVALID COMBINATIONS.
3. WHEN PCR IS THE OPERAND REGISTER, THE NUMBER IN THE OFFSET POSITION OF THE ASSEMBLY LANGUAGE STATEMENT IS THE MEMORY LOCATION ADDRESS OR LABEL. THE ASSEMBLER CALCULATES THE OFFSET.

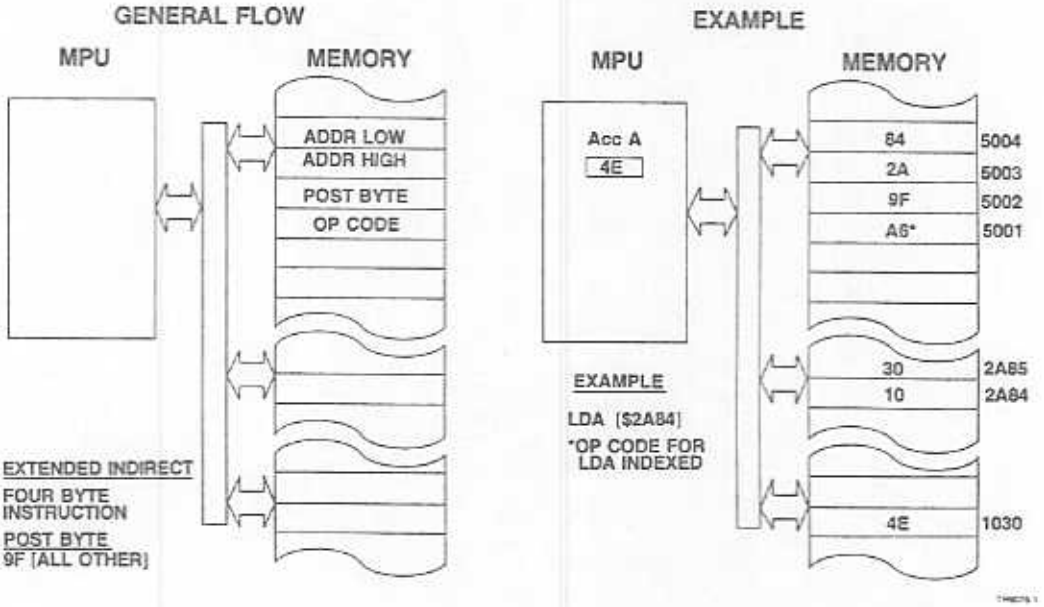
6809 INDEXED ADDRESSING MODES

TYPE	FORMS	NON INDIRECT ASSEMBLER FORM	NON INDIRECT POST-BYTE OPCODE	NON INDIRECT + ~	NON INDIRECT + #	INDIRECT ASSEMBLER FORM	INDIRECT POST-BYTE OPCODE	INDIRECT + ~	INDIRECT + #
CONSTANT OFFSET FROM R	NO OFFSET	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 BIT OFFSET	n, R	ORRNNNNN	1	0	DEFAULTS TO 8-BIT			
	8 BIT OFFSET	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 BIT OFFSET	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
ACCUMULATOR OFFSET FROM R	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	[--R]	1RR10011	6	0
CONSTANT OFFSET FROM PC	8 BIT OFFSET	n, PCR	1XX01100	3	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]	1XX11111	5	2

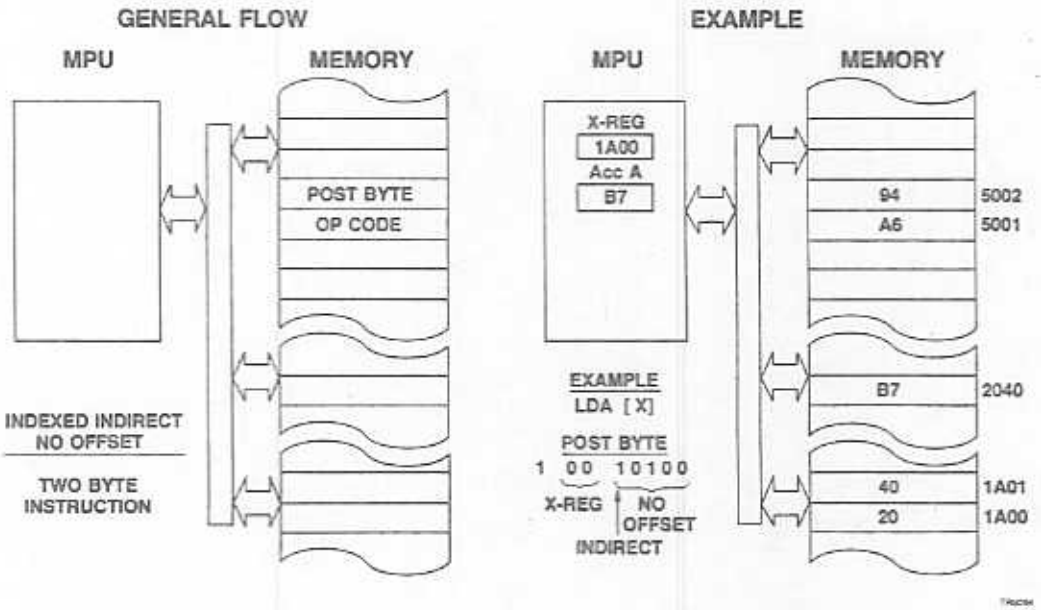
RR — 00 = X
01 = Y
10 = U
11 = S

TR9112-1

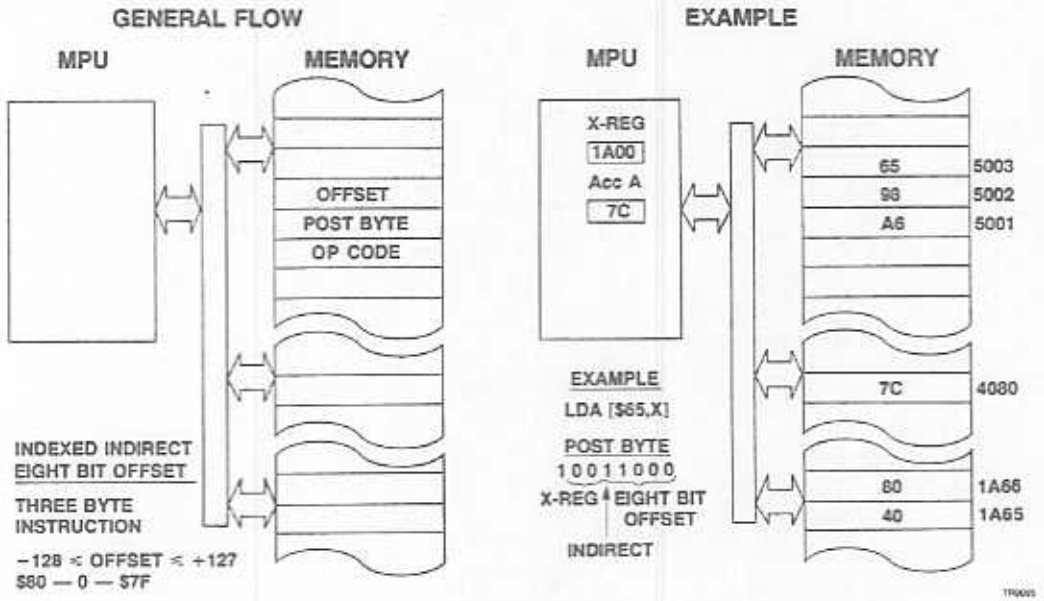
EXTENDED INDIRECT ADDRESSING



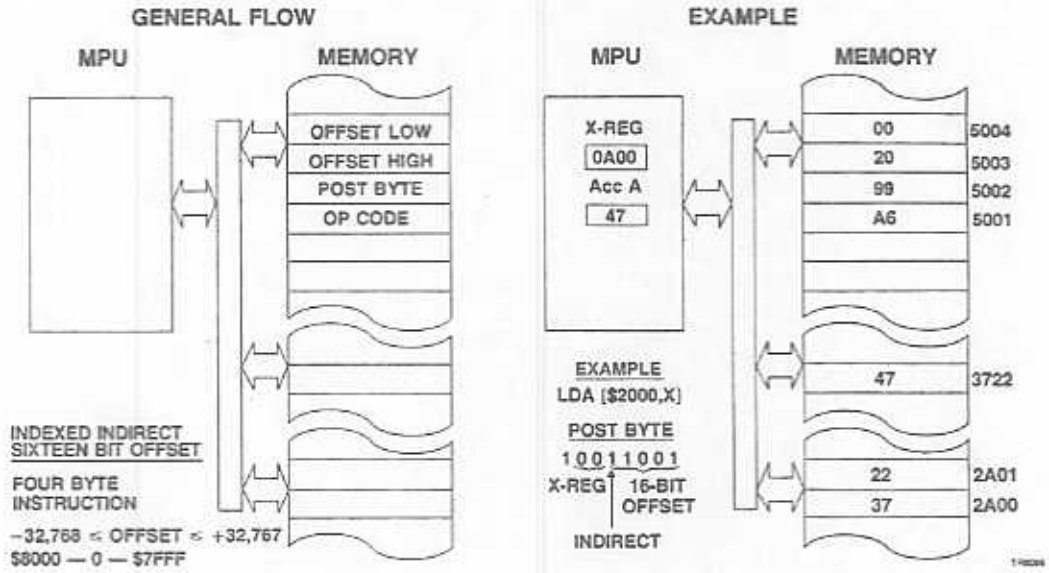
INDEXED INDIRECT ADDRESSING
(NO OFFSET)



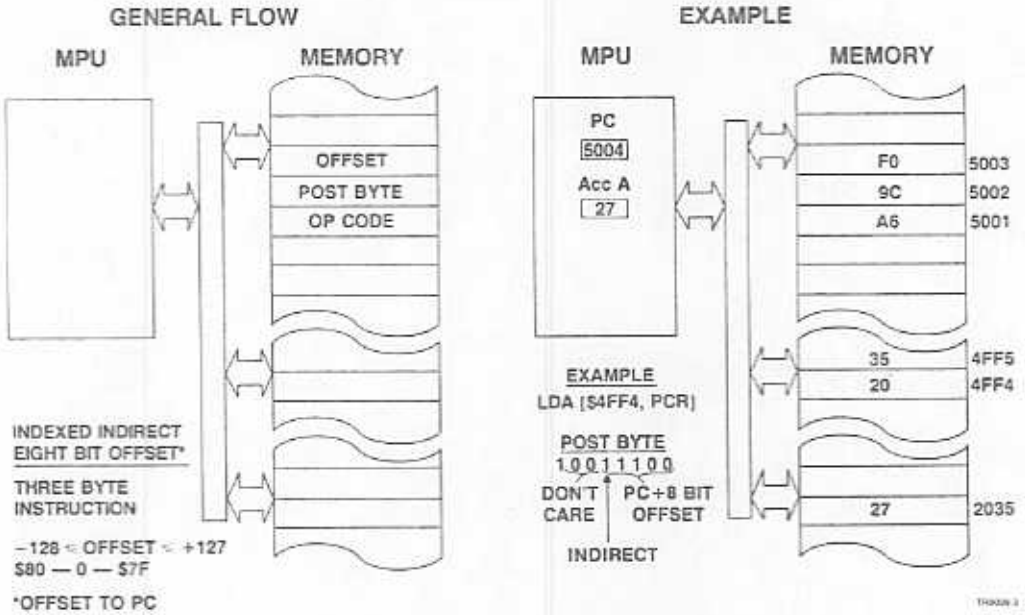
INDEXED INDIRECT ADDRESSING
(8-BIT OFFSET)



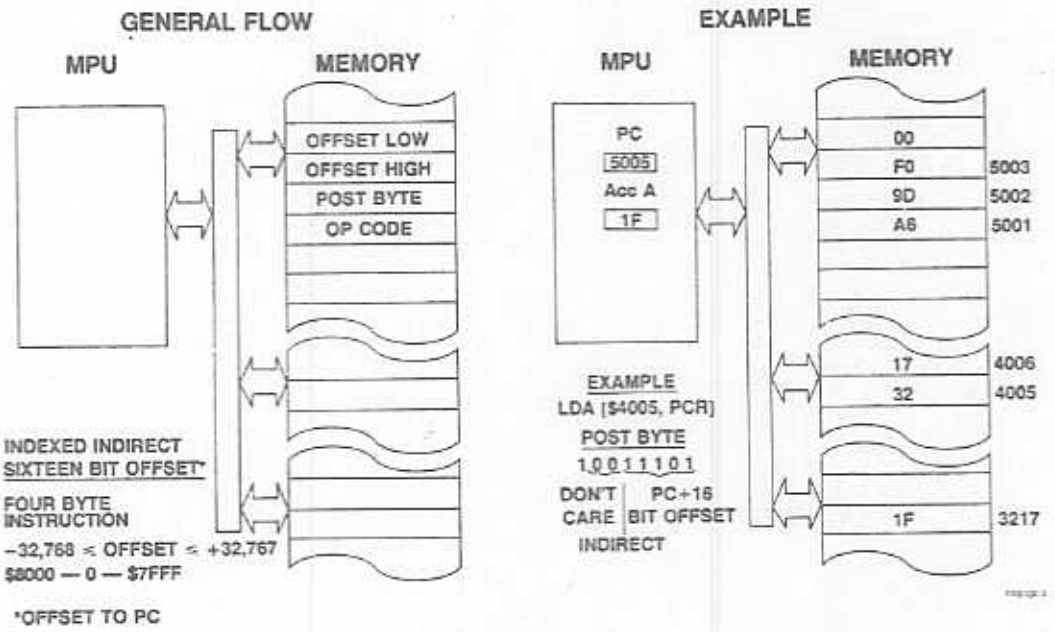
INDEXED INDIRECT ADDRESSING
(16-BIT OFFSET)



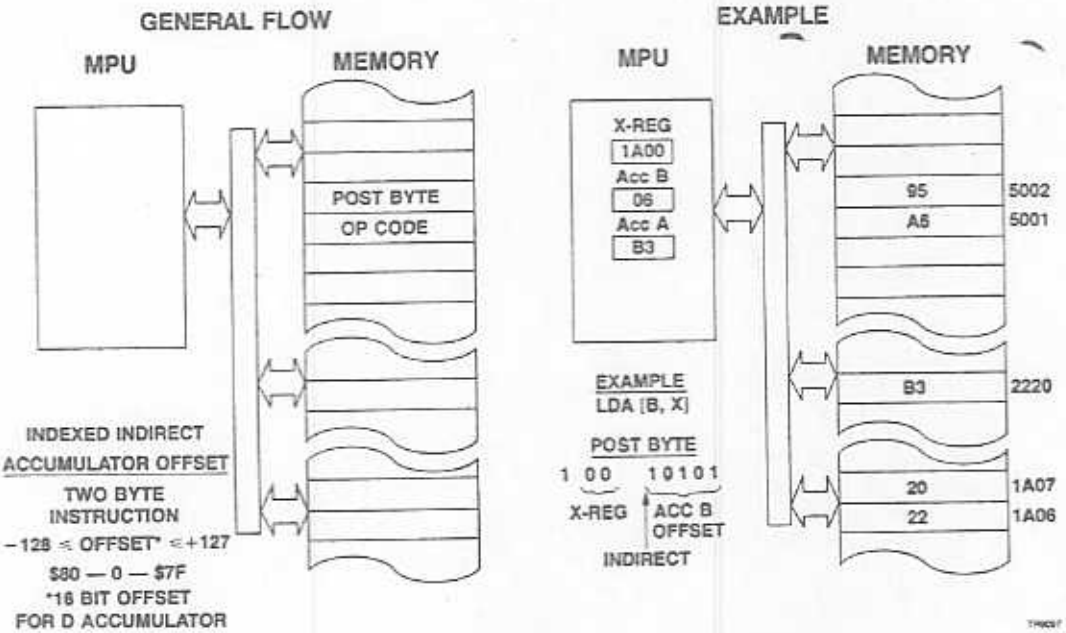
INDEXED INDIRECT ADDRESSING
(8-BIT OFFSET TO PC)



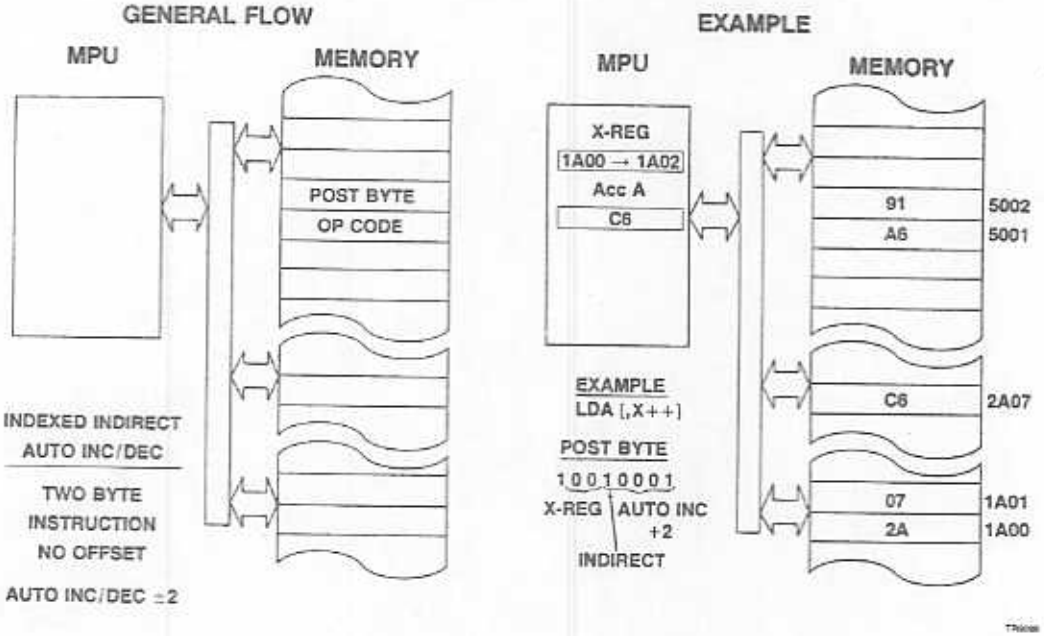
INDEXED INDIRECT ADDRESSING
(16-BIT OFFSET TO PC)



INDEXED INDIRECT ADDRESSING
(ACCUMULATOR OFFSET A, B OR D)



INDEXED INDIRECT ADDRESSING
(AUTO INC/DEC ±2)



BYTES	INHERENT	IMMEDIATE	DIRECT	EXTENDED	RELATIVE	INDEXED	EXTENDED INDIRECT	INDEXED INDIRECT
1 2 3 4	OP CODE	OP CODE DATA 8 BIT	OP CODE* ADDR L	OP CODE* ADDR H ADDR L	OP CODE OFFSET	OP CODE* POST BYTE NO OFFSET	OP CODE* POST BYTE ADDR H ADDR L	OP CODE* POST BYTE NO OFFSET
1 2 3 4	OP CODE POST BYTE TFR, EXG, PSH, PUL	OP CODE* DATA H DATA L 16 BIT			OP CODE* OFST H OFST L LONG	OP CODE* POST BYTE 5 BIT OFFSET		
1 2 3 4						OP CODE* POST BYTE OFFSET 8 BIT OFFSET		OP CODE* POST BYTE OFFSET 8 BIT OFFSET-PC
1 2 3 4						OP CODE* POST BYTE OFFSET 8 BIT OFFSET-PC		OP CODE* POST BYTE OFFSET 8 BIT OFFSET-PC
1 2 3 4						OP CODE* POST BYTE OFFSET HIGH OFFSET LOW 16 BIT OFFSET		OP CODE* POST BYTE OFFSET H OFFSET L 16 BIT OFFSET
1 2 3 4						OP CODE* POST BYTE OFFSET HIGH OFFSET LOW 16 BIT OFFSET-PC		OP CODE* POST BYTE OFFSET HIGH OFFSET LOW 16 BIT OFFSET-PC
1 2 3 4						OP CODE* POST BYTE ACC. OFFSET		OP CODE* POST BYTE ACC. OFFSET
1 2 3 4			*AN ADDITIONAL PRE-BYTE WILL BE REQUIRED FOR SOME INSTRUCTIONS			OP CODE* POST BYTE AUTO INC/DEC		OP CODE* POST BYTE AUTO INC/DEC

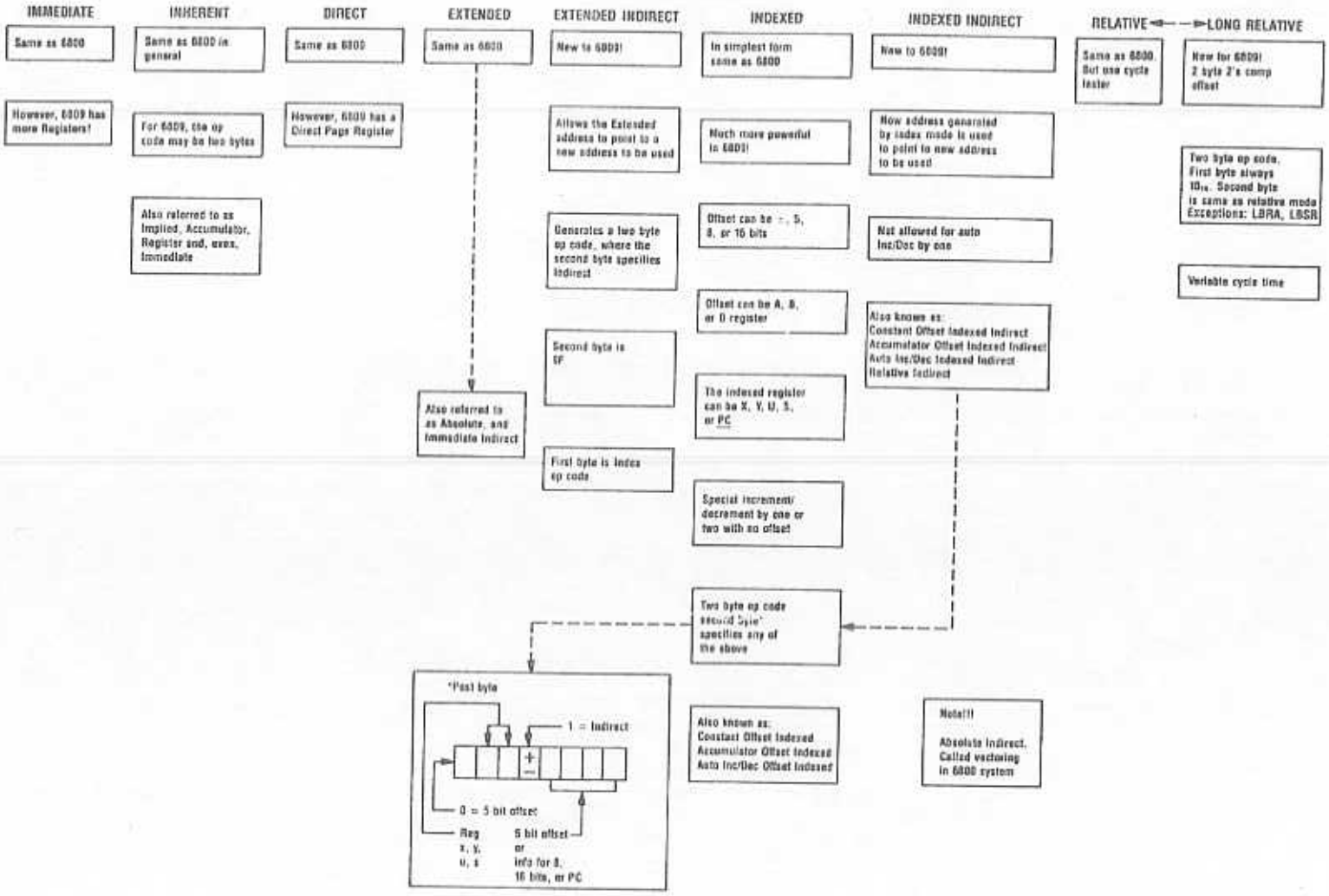
SUMMARY OF PRE-BYTE RELATED INSTRUCTIONS

NO PRE-BYTE	10 PRE-BYTE	11 PRE-BYTE
SWI	SWI2	SWI3
SUBD	CMPD	CMPU
CMPX	CMPY	CMPS
LDX	LDY	—
STX	STY	—
LDU	LDS	—
STU	STS	—
BRANCH	*LONGBRANCH	

*EXCEPT FOR BRA, LBRA, BSR, AND LBSR

E9072-1

6809 ADDRESSING MODES



TR9140

MC6800/01/02/09 COMPARISONS

		SLOW MEMORIES										
6800	VMA	DELAY CLOCK	ϕ_1, ϕ_2	STACK POINTER OPERATIONS	INTERRUPT STACKING	CONDITION CODE REGISTER	INTERRUPT VECTORS	INSTRUCTION SET	16 BIT COMPARE X REGISTER	CONDITION CODE BITS FOR TST	CONDITION CODE BITS FOR ASR LSR,ROR	ASLA ASLB LSRA LSRB
6802	SAME AS 6800	MEMORY READY INPUT	CRYSTAL INPUT ÷4	SAME AS 6800	SAME AS 6800	SAME AS 6800	SAME AS 6800	SAME AS 6800	SAME AS 6800	SAME AS 6800	SAME AS 6800	SAME AS 6800
6801	ADDRESS BUS AND R/W GO TO ALL 1'S	DELAY CLOCK	-6801- CRYSTAL INPUT ÷4 -6801S- ENABLE INPUT	SAME AS 6800	SAME AS 6800	SAME AS 6800	FFF8-7 IS INPUT CAPTURE FFF4-5 IS OUTPUT COMPARE FFF2-3 IS TIMER OVERFLOW FFF0-1 IS UART	UPWARD COMPATIBLE IN MACHINE CODE FEWER MACHINE CYCLES	C,V, & N AS WELL AS Z AFFECTED X REGISTER	SAME AS 6800	SAME AS 6800	ASLD LSRD
6809	ADDRESS BUS AND R/W GO TO ALL 1'S -6809- SAME AS 6800	-6809- MEMORY READY INPUT -6809E- DELAY CLOCK	-6809- CRYSTAL INPUT &OUT, COUT -6809E- EIN, OIN	PUSH-DECREMENT & STORE PULL-LOAD & INCREMENT	-5 MORE BYTES -A & B ARE SWAPPED -FIRQ- ONLY PC AND CCR	BIT 6- F MASK BIT BIT 7- E(ENTIRE)	FFF6-7 IS FIRQ FFF4-5 IS SWI2 FFF2-3 IS SWI1 FFF0-1 IS RESERVED	UPWARD COMPATIBLE IN ASSEMBLY LANGUAGE FEWER MACHINE CYCLES	C,V, & N AS WELL AS Z AFFECTED X,Y,U, & S REGISTERS	C IS NOT AFFECTED	V IS NOT AFFECTED	SAME AS 6800

TR5184

PROGRAMS WRITTEN FOR THE 6800/01 HAVE AN INCREASED CHANCE OF RUNNING ON THE 6809 IF THE FOLLOWING CONDITIONS ARE MET:

- PROGRAM MUST BE REASSEMBLED (ALL OP CODES ARE NOT THE SAME)
- ONLY STANDARD USE OF THE STACK (I.E. NOTHING OTHER THAN SIMPLE INTERRUPTS AND SUBROUTINE CALLS)
- ADDRESS FFF0 → FFF7 NOT USED
- SOFTWARE TIMING LOOPS NOT CRITICAL (DIFFERENT CYCLE TIMES)
- DATA WRITTEN TO/FROM CCR CAN NOT ASSUME HIGHER ORDER BITS EQUAL TO ONES

TR5065-1

```
PAGE  001  DIRPGE   SA:0  DIRPGE

00001                          *    SET DIRECT PAGE DIRECTIVE
00002                          *THIS PROGRAM SHOWS THE EFFECT OF
00003                          *THE DIRECT PAGE DIRECTIVE IN ASSEMBLY
00004                          *OF A PROGRAM.
00005                                 NAM    DIRPGE
00006                001A      A        SETDP  $1A
00007A 0000 96   00        A          LDA    $1A00
00008A 0002 B6   1B00      A          LDA    $1B00
00009A 0005 96   00        A          LDA    <$1A00
****WARNING   002--00000
00010A 0007 96   00        A          LDA    <$1B00
00011A 0009 B6   1A00      A          LDA    >$1A00
00012A 000C B6   1B00      A          LDA    >$1B00
00013A 000F 9E   FF        A          LDX    <$1AFF
00014                001B      A        SETDP  $1B
00015A 0011 96   00        A          LDA    $1B00
00016                                 END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00001--00010
```

E9246

```
PAGE  001  DBEINH   SA:0  DBEINH

00001                          *   LABELS FOR PSH, PUL, EXG, TFR, CWAI
00002                          *THIS PROGRAM SHOWS HOW LABELS
00003                          *AFFECT THE ASSEMBLY OF THE
00004                          *DOUBLE-BYTE INHERENT MODE INSTRUC-
00005                          *TIONS.
00006                                 NAM    DBEINH
00007                0006      A ACCAB  REG    A,B
00008                00BF      A CLEAR  EQU    #BF
00009                01BF      A CLEAR1 EQU    #1BF
00010A 0000 34   06        A          PSHS   #ACCAB
00011A 0002 37   06        A          PULU   #ACCAB
****ERROR   178--00000
00012A 0004 34   00        A  PSHS ACCAB
****ERROR   177--00012
00013A 0006 34   50        A  PSHS U,S,X

00015A 0008 1E   89        A          EXG    A,B
00016A 000A 1F   89        A          TFR    A,B
****ERROR   179--00013
00017A 000C 1E   00        A  EXG X,A
****ERROR   179--00017
00018A 000E 1F   00        A  TFR A,X
****WARNING   006--00000
00019A 0010 1F   18        A          TFR    X,A
****ERROR   178--00018
00020A 0012 1E   00        A  EXG #ACCAB
****ERROR   178--00020
00021A 0014 1F   00        A  TFR #ACCAB

00023A 0016 3C   BF        A          CWAI   #CLEAR
****ERROR   209--00021
00024A 0018 3C   00        A  CWAI CLEAR
****ERROR   216--00024
****ERROR   209--00025
00025A 001A 3C   00        A  CWAI
****ERROR   210--00025
00026A 001C 3C   BF        A  CWAI #CLEAR1
00027                                 END
TOTAL ERRORS 00010--00026
TOTAL WARNINGS 00001--00019
```

E9248

```
PAGE  001  OFSETS  .SA:0  OFSETS

00001                          *   OFFSETS
00002                          *THIS PROGRAM SHOWS HOW THE
00003                          *ASSEMBLER HANDLES BRANCH AND
00004                          *LONGBRANCH OFFSETS, AND 16-BIT
00005                          *REGISTER OFFSETS.
00006                                  NAM     OFSETS
00007                0000      A LONG  EQU     $0000
00008                0020      A SHORT EQU     $20
00009A 0000 20   1E   0020           BRA     SHORT
++++ERROR   200--00000
00010A 0002 20   FE   0000   BRA LONG
++++WARNING   001--00000
00011A 0004 16   0019 0020           LBRA    SHORT
00012A 0007 16   7FF6 0000           LBRA    LONG

00014A 0000 27   14   0020           BEQ     SHORT
++++ERROR   200--00010
00015A 0002 27   FE   0000   BEQ LONG
++++WARNING   001--00011
00016A 0004 1027 0008 0020           LBEQ    SHORT
00017A 0012 1027 7FEA 0000           LBEQ    LONG

00019A 0016 A6   10        A         LDA     $FFF0,X
00020A 0018 A6   88 10     A         LDA     -$FFF0,X
00021                                END
TOTAL ERRORS 00002--00015
TOTAL WARNINGS 00002--00016
```



```
PAGE 001   PCRSEC .SA:0   PCRSEC

00001                          *  SECTIONING AND PCR
00002                          *THIS PROGRAM SHOWS HOW THE
00003                          *PCR INDEX MODE IS AFFECTED
00004                          *BY SECTIONING
00005                                  NAM     PCRSEC
00006                                  OPT     REL

00008P   0000          0001    A TEMPO  RMB     1
00009P   0001   A6  8C FC              LDA     TEMPO, PCR

00011                  0100    A TEMP   EQU     $100
****ERROR        241--00000
00012P   0004   A6  8D FFFC        LDA TEMP,PCR

00014B   0000                          BSCT
00015B   0000          0001    A NTEMP  RMB     1
00016P   0008                          PSCT
****ERROR        241--00012
00017P   0008   A6  8D FFFC        LDA NTEMP,PCR
00018                                  END

TOTAL ERRORS 00002--00017
TOTAL WARNINGS 00000--00000
```



```
PAGE  001  M6809A  .SA:0  TEST 6

00001
00002                          NAM     TEST     6809 2-14-79 WKJR4
00003                  000A    A       SETDP   10
00004                  0030    A INDEX REG     X,Y
00005                  0016    A STACK REG     X,A,B
00006                  2000    A HOUSE EQU     $2000
00007                  0064    A APPLE EQU     100
00008                  0000    A ORANGE EQU    0000
00008A 0000 B6   8030     A            LDA     $8030
00009A 0003 B6   8030     A            LDA     >$8030
****WARNING   002--00000
00010A 0006 96   80       A            LDA     <$8030
00011A 0008 B6   0064     A            LDA     >100
****WARNING   002--00010
00012A 000B 96   64       A            LDA     <100
00013A 000D B6   0064     A            LDA     100

00015A 0010 B6   7F80     A            LDA     -$8000
00016A 0013 B6   7F80     A            LDA     >-$8000
****WARNING   002--00012
00017A 0016 96   80       A            LDA     <-$8000
00018A 0019 B6   FF9C     A            LDA     -100
00019A 001B B6   FF9C     A            LDA     >-100
****WARNING   002--00017
00020A 001E 96   9C       A            LDA     <-100
00021                  ********************************************

00022A 0020 A6   88 10    A            LDA     $10,X
00024A 0023 A6   88 10    A            LDA     <$10,X
00025A 0026 A6   89 0010  A            LDA     >$10,X
00026A 002A A6   10       A            LDA     -$10,X
00027A 002C A6   88 F0    A            LDA     <-$10,X
00028A 002F A6   89 FFF0  A            LDA     >-$10,X
00029                  ********************************************
00030A 0033 A6   89 00FF  A            LDA     $FF,X
00031A 0037 A6   89 00FF  A            LDA     >$FF,X
****ERROR   210--00000
00032A 003B A6   88 FF    A  LDA <$FF,X
00033A 003E A6   89 FF01  A            LDA     -$FF,X
****ERROR   210--00032
00034A 0042 A6   88 01    A  LDA <-$FF,X
00035A 0045 A6   89 FF01  A            LDA     >-$FF,X
00036                  ********************************************

00039A 0049 A6   1E       A            LDA     $FFFE,X
00040A 004B A6   88 FE    A            LDA     <$FFFE,X
00041A 004L A6   89 FFFE  A            LDA     >$FFFE,X
00042A 0052 A6   82       A            LDA     -$FFFE,X
00043A 0054 A6   89 0002  A            LDA     >-$FFFE,X
00044A 0058 A6   88 02    A            LDA     <-$FFFE,X
00045A 005B AE   84       A            LDX     0,X
00046A 005D AE   89 0000  A            LDX     >00,X
00047A 0061 AE   88 00    A            LDX     <ORANGE,X
00048A 0064 34   FF       A            PSHS    #ALL
00049A 0066 35   16       A            PULS    #STACK
00050A 0068 3C   64       A            CWAI    #APPLE
00051A 006A 3C   80       A            CWAI    #80
****WARNING   002--00020
00052A 006C FC   90       A            LDD     <$0080
00053A 006E FC   1F90     A            LDD     >$0080
****WARNING   002--00052
00054A 0071 FC   F2       A            LDD     <1010
00055A 0073 FC   03F2     A            LDD     >1010
00056                  00FF     A ALL   REG     A,B,X,Y,U,DP,CC,PC
00057                           END
TOTAL ERRORS 00002--00014
TOTAL WARNINGS 00006--00054
```

CLEAR $70 →$78

```
        LDX     #9
LI      CLR     $6F, X
        LEAX    -1, X
        BNE     LI

        LDY     #0
        LDX     #8
L3      STY     $70, X
        LEAX    -2, X
        BNE     L3
        STY     $70, X

        CLRA
        LDU     #$79
L2      PSHU    A
        CMPU    #$70
        BNE     L2
```

TR9034

```
00050 00005                     *  CLEAR MEMORY $70-$78
00060 00006A 2000 8E   0009     A       LDX     #9
00070 00007A 2003 6F   99 6F    A L1    CLR     $6F,X
00080 00008A 2006 30   1F       A       LEAX    -1,X
00090 00009A 2008 26   F9   2003        BNE     L1



00110 00011A 200A 108E 0000     A       LDY     #0
00120 00012A 200E 8E   0008     A       LDX     #8
00130 00013A 2011 10AF 88 70    A L3    STY     $70,X
00140 00014A 2015 30   1E       A       LEAX    -2,X
00150 00015A 2017 26   FB   2011        BNE     L3
00160 00016A 2019 10AF 88 70    A       STY     $70,X



00180 00018A 201D 4F                    CLRA
00190 00019A 201E CE   0079     A       LDU     #$79
00200 00020A 2021 36   02       A L2    PSHU    A
00210 00021A 2023 1183 0070     A       CMPU    #$70
00220 00022A 2027 26   F8   2021        BNE     L2
```

TR9015

MOVE $80 BYTES
0 →100; $7F →$17F

```
        LDX     #0
MORE    LDA     X+
        STA     $FF, X
        CMPX    #$80
        BNE     MORE

        LDX     #$80
AGAIN   LDA     -1, X
        STA     $FF, X
        LEAX    -1, X
        BNE     AGAIN
```

TR9027

```
00500 00050                     *  MOVE $80 BYTES FROM $0 TO $100
00510 00051A 205D 8E   0000     A       LDX     #0
00520 00052A 2060 A6   80       A MORE  LDA     X+
00530 00053A 2062 A7   89 00FF  A       STA     $FF,X
00540 00054A 2066 8C   0080     A       CMPX    #$80
00550 00055A 2069 26   F5   2060        BNE     MORE



00570 00057A 206B 8E   0080     A       LDX     #$80
00580 00058A 206E A6   1F       A AGAIN LDA     -1,X
00590 00059A 2070 A7   89 00FF  A       STA     $FF,X
00600 00060A 2074 30   1F       A       LEAX    -1,X
00610 00061A 2076 26   F6   206E        BNE     AGAIN
```

TR9017

MOVE 100 BYTES
0 →$1000; $FF →$10FF

```
            LDU     #0
            LDX     #$1000
LC          PULU    A
            STA     X+
            CMPX    #$1100
            BNE     LC

            LDX     #$100
            LDY     #$1100
LD          LDA     —X
            STA     —Y
            CMPX    #$0
            BNE     LD

            LDX     #0
            LDY     #$1000
LA          LDA     0,X
            STA     0, Y
            LEAX    +1, X
            LEAY    +1, Y
            CMPX    #$100
            BNE     LA
```

TR0026-1

```
00630  00063                        *   MOVE $100 BYTES FROM $0 TO $1000
00640  00064A  207B  8E  0000        A   LDX  #0
00650  00065A  207D  108E  1000      A   LDY  #$1000
00660  00066A  207F  A5  84          A LA LDA  0, X
00670  00067A  2081  A7  A4          A   STA  0, Y
00680  00068A  2083  30  01          A   LEAX 1, X
00690  00069A  2085  31  21          A   LEAY 1, Y
00700  00070A  2087  8C  0100        A   CMPX #$100
00710  00071A  208A  26  F3      207F    BNE  LA

00830  00083A  20A1  CE  0000        A   LDU  #0
00840  00084A  20A4  8E  1000        A   LDX  #$1000
00850  00085A  20A7  37  02          A LC PULU A
00860  00086A  20A9  A7  80          A   STA  X+
00870  00087A  20AB  8C  1100        A   CMPX #$1100
00880  00088A  20AE  26  F7      20A7    BNE  LC

00900  00090A  20B0  8E  0100        A   LDX  #$100
00910  00091A  20B3  108E  1100      A   LDY  #$1100
00920  00092A  20B7  A6  82          A LD LDA  —X
00930  00093A  20B9  A7  A2          A   STA  —Y
00940  00094A  20BB  8C  0000        A   CMPX #$0
00950  00095A  20BE  26  F7      20B7    BNE  LD
```

TR0018-1

ODD — EVEN

```
            LDX     #5
LOOP        LDA     $7F, X
            COM     $7F, X
            ROR     A
            BCS     ODD
            CLR     $7F, X
ODD         LEAX    -1, X
            BNE     LOOP

            LDX     #$84
LOOP 1      COM     X
            LDA     -X
            ROR     A
            BCC     ODD 1
            CLR     +1, X
ODD 1       CMPX    #$7F
            BNE     LOOP 1
```

TR0024

ODD — EVEN

```
                LDX     #$7F
                LDA     #5
LOOP 2          COM     A, X
                LDB     A, X
                ROR     B
                BCC     ODD2
                CLR     A, X
ODD 2           DEC     A
                BNE     LOOP 2
```

TR9025

```
00000 00000                     *  ODD-EVEN SOLUTION
00000 00000A 20C0 8E   0005      A       LDX   #5
01000 00100A 20C2 A6   00 7F     A LOOP  LDA   #7F,X
01010 00101A 20C6 63   00 7F     A       COM   #7F,X
01018 00101A 20C8 46                     RORA
01020 00102A 20C9 46                     RORA
01030 00103A 20CA 25   03   20CF         BCS   ODD
01040 00104A 20CC 6F   80 7F     A       CLR   #7F,X
01050 00105A 20CF 30   1F        A ODD   LEAX  -1,X
01060 00106A 20D1 26   F0   20C3         BNE   LOOP
```

```
01000 00108A 20D3 8E   0004      A       LDX   #$04
01090 00109A 20D6 63   04        A LOOP1 COM   X
01100 00110A 20D8 A6   02        A       LDA   -X
01110 00111A 20DA 46                     RORA
01120 00112A 20DB 24   02   20DF         BCC   ODD1
01130 00113A 20DD 6F   01        A       CLR   1,X
01150 00114A 20DF 8C   007F      A ODD1  CMPX  #$7F
01160 00115A 20E2 26   F2   20D6         BNE   LOOP1
```

```
01180 00117A 20E4 8E   007F      A       LDX   #$7F
01190 00118A 20E7 86   05        A       LDA   #5
01200 00119A 20E9 63   06        A LOOP2 COM   A,X
01210 00120A 20EB E6   06        A       LDB   A,X
01220 00121A 20ED 56                     RORB
01230 00122A 20EE 24   02   20F2         BCC   ODD2
01240 00123A 20F0 6F   86        A       CLR   A,X
01250 00124A 20F2 4A             ODD2    DECA
01260 00125A 20F3 26   F4   20E9         BNE   LOOP2
```

TR9026

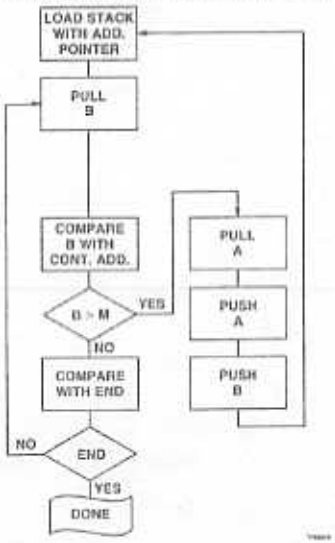

```
PAGE  001  CSRCH    SA:0  CSRCH

00001                           *       CHARACTER SEARCH
00002                           *SEARCH A TABLE OF N CHARACTERS FOR
00003                           *A SPECIFIC CHARACTER.  IF FOUND, RETURN
00004                           *THE ADDRESS OF THE MATCH. ELSE RETURN
00005                           *ZERO.  LET N BE 40  CHARACTER IS IN A
00006                                   NAM     CSRCH
00007                                   OPT     REL
00008P 0000 8E   0012      P CSRCH      LDX     #TABLE
00009P 0003 C6   28        A            LDB     #40
00010P 0005 A1   80        A AGAIN      CMPA    X+
00011P 0007 27   06   000F              BEQ     DEX
00012P 0009 5A                          DECB
00013P 000A 26   F9   0005              BNE     AGAIN
00014P 000C 8E   0001      A            LDX     #1
00015P 000F 30   1F        A DEX        LEAX    -1,X
00016P 0011 39                          RTS
00017P 0012       00       A TABLE      FCB     0,.....,0
00018P 0026       00       A            FCB     0,.....,0
00019                                   END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

E9249-1

```
PAGE 001  CODECON  SA:0

*    CODE CONVERSION EXAMPLE
*A HEXADECIMAL NUMBER,0-F, IN ACCB IS TO BE
*CONVERTED TO A BCD NUMBER UPON RTS.  THE
*BCD NUMBER IS IN ACCB
 NAM CODECON
 OPT REL
TABLE FCB 0,1,2,3,4,5,6,7,8,9,$10,$11,$12,$13,$14,$15
 LDX #TABLE
 LDA B,X
 TFR A,B
 RTS
 END
```

```
PAGE  001  CODECON  SA:0  CODECON

00001                           *    CODE CONVERSION EXAMPLE
00002                           *A HEXADECIMAL NUMBER,0-F, IN ACCB IS TO BE
00003                           *CONVERTED TO A BCD NUMBER UPON RTS.  THE
00004                           *BCD NUMBER IS IN ACCB
00005                                   NAM     CODECON
00006                                   OPT     REL
00007P 0000       00       A TABLE      FCB     0,1,2,3,4,5,6,7,8,9,$10,$11,$12
       P 0001     01       A
       P 0002     02       A
       P 0003     03       A
       P 0004     04       A
       P 0005     05       A
       P 0006     06       A
       P 0007     07       A
       P 0008     08       A
       P 0009     09       A
       P 000A     10       A
       P 000B     11       A
       P 000C     12       A
       P 000D     13       A
       P 000E     14       A
       P 000F     15       A
00008P 0010 8E   0000      P            LDX     #TABLE
00009P 0013 A6   85        A            LDA     B,X
00010P 0015 1F   89        A            TFR     A,B
00011P 0017 39                          RTS
00012                                   END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

SEQUENCE A GROUP OF NUMBERS IN ASCENDING ORDER.



PAGE 001 SEQUX .SA:0 SEQUX

```
00001                                   *  SEQUENCER
00002                                   *THIS PROGRAM SEQUENCES A SERIES OF NUMBERS
00003                                   *SUCH THAT THE LARGEST NUMBER GOES TO THE
00004                                   *LOWEST ADDRESS. THE U REGISTER FUNCTIONS
00005                                   *AS BOTH A STACK POINTER AND AN INDEX
00006                                   *REGISTER. UPON ENTERING THIS SUBROUTINE,
00007                                   *U CONTAINS THE STARTING ADDRESS OF THE
00008                                   *NUMBERS TO BE SEQUENCED, AND X, THE
00009                                   *ENDING ADDRESS.
00010                                       NAM   SEQUX
00011A 2000                                ORG   $2000
00012A 2000 34  50          A          PSHS  U,X
00013A 2002 EE  62          A       L1 LDU   2, S
00014A 2004 37  04          A       L3 PULU  B
00015A 2006 E1  C4          A          CMPB  U
00016A 2008 24  06       2010          BCC   L2
00017A 200A 37  02          A          PULU  A
00018A 200C 36  06          A          PSHU  A, B
00019A 200E 20  F2       2002          BRA   L1
00020A 2010 11A3 E4         A       L2 CMPU  S
00021A 2013 26  EF       2004          BNE   L3
00022A 2015 32  64          A          LEAS  4, S
00023A 2017 39                         RTS
00024                                  END
TOTAL ERRORS 00000---00000
TOTAL WARNINGS 00000---00000
```


PAGE 001 TWOSMULT. SA:0 TWOSMU

```
00001                                  NAM   TWOSMLT
00002A 2000                            ORG   $2000
00003A 2000 6F  C2          A          CLR   - U
00004A 2002 4D                         TSTA
00005A 2003 2B  09       200E          BMI   COM1
00006A 2005 5D                      CONT TSTB
00007A 2006 2B  0B       2013          BMI   COM2
00008A 2008 3D                      MULT MUL
00009A 2009 6D  C0          A          TST   U+
00010A 200B 26  0B       2018          BNE   NEG1
00011A 200D 39                         RTS
00012A 200E 40                      COM1 NEGA
00013A 200F 63  C4          A          COM   U
00014A 2011 20  F2       2005          BRA   CONT
00015A 2013 50                      COM2 NEGB
00016A 2014 63  C4          A          COM   U
00017A 2016 20  F0       2008          BRA   MULT
00018A 2018 43                      NEG1 COMA
00019A 2019 53                           COMB
00020A 201A C3  0001        A          ADDD  #$01
00021A 201D 39                         RTS
00022                                  END
TOTAL ERRORS 00000---00000
TOTAL WARNINGS 00000---00000
```

16 X 16 MULTIPLY ROUTINE

```
   3344                3344
   1122                1122
   6688                  88 } = 44 X 22 =      908
  6688    =             88
  3344                  66 }
 3344                 66   } = 33 X 22 =     6C6
036E5308              44 }
                      44   } = 44 X 11 =     484
                    33 }
                    33   } = 33 X 11 =    363
                  036E5308                036E5308
```

```
      V1H  V1L
       \   /
V1 =   33 44
V2 =   11 22
       /   \
      V2H  V2L
```



```
PAGE 001  MULA     SA:0

*   16X16 MULTIPLY-ABSOLUTE MACHINE CODE
*MULTIPLY TWO 16-BIT POSITIVE VALUES
*TO GENERATE A 32-BIT PRODUCT.  AT TER-
*MINATION, BOTH INPUT VALUES AND THE RE-
*SULT WILL BE IN MEMORY.
*(V1H,V1L)X(V2H,V2L)=                          HP(V1L X V2L),LP(V1H X V2L)
*                    +              HP(V1H X V2L),LP(V1H X V2L)
*                    +              HP(V1L X V2H),LP(V1L X V2H)
*                   +HP(V1H X V2H),LP(V1H X V2H)
*                   ----------------------------------------------------
*                         32-BIT PRODUCT
*V1=16-BIT FIRST VARIABLE
*V2=16-BIT SECOND VARIABLE
*H=HIGH BYTE
*L=LOW BYTE
*P=PRODUCT
 NAM MULA
 ORG $2000
MV1 FDB $3344
MV2 FDB $1122
M32P FDB 0,0
SETUP LDX #MV1     *MV1=MEMORY LOCATION OF V1
 LDY #MV2          *MV2=MEMORY LOCATION OF V2
 LDU #M32P         *M32P=MEMORY LOCATION OF 32-BIT PRODUCT
 SPC
 CLR 0
 CLR 1,U
 LDA 1,X           *V1L-->A
 LDB 1,Y           *V2L-->B
 MUL
 STD 2,U           *HP(V1L X V2L)-->M32P+2,LP(V1L X V2L)-->M32P+3
 LDA ,X            *V1H-->A
 LDB 1,Y           *V2L-->B
 MUL
 ADDD 1,U
 STD 1,U           *HP(V1H X V2L)-->M32P+1,
+                        *LP(V1H X V2L)+HP(V1L X V2L)-->M32P+2
 BCC NEXT1
 INC 0,U           *STORE C=1 IN M32P
NEXT1 LDA 1,X      *V1L-->A
 LDB ,Y            *V2H-->B
 MUL
 ADDD 1,U
 STD 1,U           *HP(V1L X V2H)+HP(V1H X V2L)-->M32P+1
+                        *LP(V1L X V2H)+LP(V1H X V2L)+HP(V1L X V2L)-->M32P+2
 BCC NEXT2
 INC 0,U           *ADD C=1 TO (M32P)
NEXT2 LDA ,X       *V1H-->A
 LDB ,Y            *V2H-->B
 MUL
 ADDD 0,U
 STD 0,U           *HP(V1H X V2H)-->M32P,
+                        *LP(V1H X V2H)+HP(V1L X V2H)+HP(V1H X V2L) -->M32P+1
 RTS
 END
```

```
PAGE  001  MULA     SA 0  MULA

00001                          *   16X16 MULTIPLY-ABSOLUTE MACHINE CODE
00002                          *MULTIPLY TWO 16-BIT POSITIVE VALUES
00003                          *TO GENERATE A 32-BIT PRODUCT. AT TER-
00004                          *MINATION, BOTH INPUT VALUES AND THE RE-
00005                          *SULT WILL BE IN MEMORY.
00006                          *(V1H,V1L)X(V2H,V2L)=
00007                          *                    HP(V1H X V
00008                          *                    HP(V1L X V
00009                          *                  +HP(V1H X V2H),LP(V1H X V
00010                          *                  ----------------------
00011                          *                        32-BIT PRODUCT
00012                          *V1=16-BIT FIRST VARIABLE
00013                          *V2=16-BIT SECOND VARIABLE
00014                          *H=HIGH BYTE
00015                          *L=LOW BYTE
00016                          *P=PRODUCT
00017                                  NAM   MULA
00018A 2000                            ORG   $2000
00019A 2000       3344      A MV1      FDB   $3344
00020A 2002       1122      A MV2      FDB   $1122
00021A 2004       0000      A M32P     FDB   0,0
00022A 2008 8E    2000      A SETUP    LDX   #MV1      *MV1=MEMORY LOCATION O
00023A 200B 10CE  2002      A          LDY   #MV2      *MV2=MEMORY LOCATION O
00024A 200F CE    2004      A          LDU   #M32P     *M32P=MEMORY LOCATION

00026A 2012 6F    C4        A          CLR   U
00027A 2014 6F    41        A          CLR   1,U
00028A 2016 A6    01        A          LDA   1,X       *V1L-->A
00029A 2018 E6    21        A          LDB   1,Y       *V2L-->B
00030A 201A 3D                         MUL
00031A 201B ED    42        A          STD   2,U       *HP(V1L X V2L)-->M32P+
00032A 201D A6    04        A          LDA   X         *V1H-->A
00033A 201F E6    21        A          LDB   1,Y       *V2L-->B
00034A 2021 3D                         MUL
00035A 2022 E3    41        A          ADDD  1,U
00036A 2024 ED    41        A          STD   1,U       *HP(V1H X V2L)-->M32P+
00037                  *                               *LP(V1H X V2L)+HP(V1L
00038A 2026 24    02      202A         BCC   NEXT1
00039A 2028 6C    C4        A          INC   U         *STORE C=1 IN M32P
00040A 202A A6    01        A NEXT1    LDA   1,X       *V1L-->A
00041A 202C E6    A4        A          LDB   Y         *V2H-->B
00042A 202E 3D                         MUL
00043A 202F E3    41        A          ADDD  1,U
00044A 2031 ED    41        A          STD   1,U       *HP(V1L X V2H)+HP(V1H
00045                  *                               *LP(V1L X V2H)+LP(V1H
00046A 2033 24    02      2037         BCC   NEXT2
00047A 2035 6C    C4        A          INC   U         *ADD C=1 TO (M32P)
00048A 2037 A6    01        A NEXT2    LDA   X         *V1H-->A
00049A 2039 E6    A4        A          LDB   Y         *V2H-->B
00050A 203B 3D                         MUL
00051A 203C E3    C4        A          ADDD  U
00052A 203E ED    C4        A          STD   U         *HP(V1H X V2H)-->M32P+
00053                  *                               *LP(V1H X V2H)+HP(V1L
00054A 2040 39                         RTS
00055                                  END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```



```
PAGE 001  MULRRR   SA 0

*   16X16 MULTIPLY-RE-ENTRANT
*MULTIPLY TWO 16-BIT POSITIVE VALUES
*TO GENERATE A 32-BIT PRODUCT.
*(V1H,V1L)X(V2H,V2L)=
*                                  HP(V1L X V2L),LP(V1L X V2L)
*                    HP(V1H X V2L),LP(V1H X V2L)
*                    HP(V1L X V2H),LP(V1L X V2H)
*                  +HP(V1H X V2H),LP(V1H X V2H)
*                  ----------------------------------------------
*                              32-BIT PRODUCT
*
*V1=16-BIT FIRST VARIABLE
*V2=16-BIT SECOND VARIABLE
*H=HIGH BYTE
*L=LOW BYTE
*P=PRODUCT
*PRIOR TO CALLING THE SUBROUTINE, X=V1 AND Y=V2.  UPON RTS, X CONTAINS
*THE UPPER TWO BYTES OF THE ANSWER, AND D CONTAINS THE LOWER TWO BYTES
 NAM MULRRR
 OPT REL
SETUP PSHS X,Y
 LEAS -4,S
 LEAU 0,S       *U POINTS TO MEMORY LOCATION OF 32-BIT PRODUCT
 SPC
 CLR U
 CLR 1,U
 LDA 1,X        *V1L-->A
 LDB 1,Y        *V2L-->B
 MUL
 STD 2,U        *HP(V1L X V2L)-->M32P+2,LP(V1L X V2L)-->M32P+3
 LDA X          *V1H-->A
 LDB 1,Y        *V2L-->B
 MUL
 ADDD 1,U
 STD 1,U        *HP(V1H X V2L)-->M32P+1,
*                      +LP(V1H X V2L)+HP(V1L X V2L)-->M32P+2
 BCC NEXT1
 INC U          *STORE C=1 IN M32P
NEXT1 LDA 1,X   *V1L-->A
 LDB Y          *V2H-->B
 MUL
 ADDD 1,U
 STD 1,U        *HP(V1L X V2H)+HP(V1H X V2L)-->M32P+1
*                     +LP(V1L X V2H)+LP(V1H X V2L)+HP(V1L X V2L)-->M32P+2
 BCC NEXT2
 INC U          *ADD C=1 TO (M32P)
NEXT2 LDA X     *V1H-->A
 LDB Y          *V2H-->B
 MUL
 ADDD U
 STD U          *HP(V1H X V2H)-->M32P,
*                     +LP(V1H X V2H)+HP(V1L X V2H)+HP(V1H X V2L)-->M32P+1
 LDX 0,U        *UPPER 2 BYTES-->X
 LDD 2,U        *LOWER 2 BYTES-->D
 LEAS 6,S       *CLEAN-UP HARDWARE STACK
 RTS
 END
```

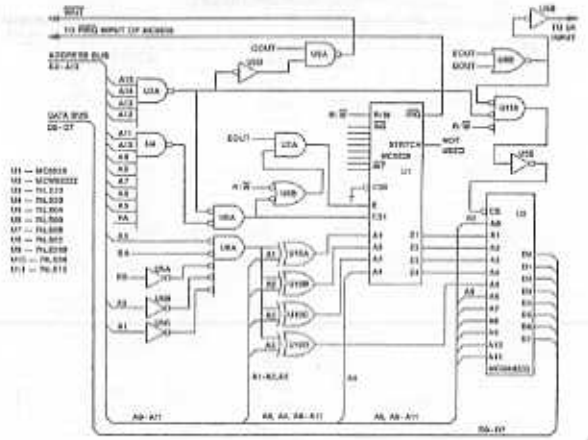
```
PAGE  001  MULRRR   SA:0  MULRRR

00001                          *   16X16 MULTIPLY-RE-ENTRANT
00002                          *MULTIPLY TWO 16-BIT POSITIVE VALUES
00003                          *TO GENERATE A 32-BIT PRODUCT
00004                          *(V1H,V1L)X(V2H,V2L)=
00005                          *                          HP(V1H X V
00006                          *                        + HP(V1L X V
00007                          *                        +HP(V1H X V2H),LP(V1H X V
00008                          *                        ------------------------
00009                          *                          32-BIT PRODUCT
00010                          *V1=16-BIT FIRST VARIABLE
00011                          *V2=16-BIT SECOND VARIABLE
00012                          *H=HIGH BYTE
00013                          *L=LOW BYTE
00014                          *P=PRODUCT
00015                          *PRIOR TO CALLING THE SUBROUTINE,X=V1 AND Y=V
00016                          *THE UPPER TWO BYTES OF THE ANSWER, AND D CONT
00017                                  NAM   MULRRR
00018                                  OPT   REL
00019P 0000 34   30        A SETUP     PSHS  X,Y
00020P 0002 30   E4        A           LEAX  0,S       *X POINTS TO MEMORY LO
00021P 0004 31   62        A           LEAY  2,S       *Y POINTS TO MEMORY LO
00022P 0006 32   7C        A           LEAS  -4,S
00023P 0008 33   E4        A           LEAU  0,S       *U POINTS TO MEMORY LO

00025P 000A 6F   C4        A           CLR   U
00026P 000C 6F   41        A           CLR   1,U
00027P 000E A6   01        A           LDA   1,X       *V1L-->A
00028P 0010 E6   21        A           LDB   1,Y       *V2L-->B
00029P 0012 3D                         MUL
00030P 0013 ED   42        A           STD   2,U       *HP(V1L X V2L)-->M32P+
00031P 0015 A6   84        A           LDA   ,X        *V1H-->A
00032P 0017 E6   21        A           LDB   1,Y       *V2L-->B
00033P 0019 3D                         MUL
00034P 001A E3   41        A           ADDD  1,U
00035P 001C ED   41        A           STD   1,U       *HP(V1H X V2L)-->M32P+
00036                                                  *LP(V1H X V2L)+HP(V1L
00037P 001E 24   02        0022        BCC   NEXT1
00038P 0020 6C   C4        A           INC   U         *STORE C=1 IN M32P
00039P 0022 A6   01        A NEXT1     LDA   1,X       *V1L-->A
00040P 0024 E6   84        A           LDB   ,Y        *V2H-->B
00041P 0026 3D                         MUL
00042P 0027 E3   41        A           ADDD  1,U
00043P 0029 ED   41        A           STD   1,U       *HP(V1L X V2H)+HP(V1H
00044                                                  *LP(V1L X V2H)+LP(V1H
00045P 002B 24   02        002F        BCC   NEXT2
00046P 002D 6C   C4        A           INC   U         *ADD C=1 TO (M32P)
00047P 002F A6   84        A NEXT2     LDA   ,X        *V1H-->A
00048P 0031 E6   04        A           LDB   ,Y        *V2H-->B
00049P 0033 3D                         MUL
00050P 0034 E3   C4        A           ADDD  U
00051P 0036 ED   C4        A           STD   U         *HP(V1H X V2H)-->M32P,
00052                                                  *LP(V1H X V2H)+HP(V1L
00053P 0038 AE   C4        A           LDX   0,U       *UPPER 2 BYTES-->X
00054P 003A EC   42        A           LDD   2,U       *LOWER 2 BYTES-->D
00055P 003C 32   68        A           LEAS  8,S       *CLEAN-UP HARDWARE STA
00056P 003E 39                         RTS
00057                                  END
TOTAL ERRORS 00000--00000
```



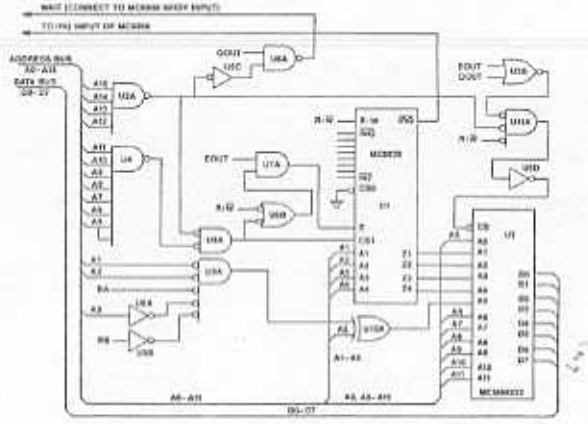
```
PAGE  001  MULRR    SA:0  MULRR

00001                          *   16X16 MULTIPLY-RELOCATABLE-RELASM
00002                          *MULTIPLY TWO 16-BIT POSITIVE VALUES
00003                          *TO GENERATE A 32-BIT PRODUCT.  AT TER-
00004                          *MINATION, BOTH INPUT VALUES AND THE RE-
00005                          *SULT WILL BE IN MEMORY.
00006                          *(V1H,V1L)X(V2H,V2L)=
00007                          *                          HP(V1H X V
00008                          *                        + HP(V1L X V
00009                          *                        +HP(V1H X V2H),LP(V1H X V
00010                          *                        ------------------------
00011                          *                          32-BIT PRODUCT
00012                          *V1=16-BIT FIRST VARIABLE
00013                          *V2=16-BIT SECOND VARIABLE
00014                          *H=HIGH BYTE
00015                          *L=LOW BYTE
00016                          *P=PRODUCT
00017                                  NAM   MULRR
00018                                  OPT   REL
00019P 0000      3344      A MV1       FDB   $3344
00020P 0002      1122      A MV2       FDB   $1122
00021P 0004      0000      A M32P      FDB   0,0
00022P 0006 30   8C F5       SETUP     LEAX  MV1,PCR   *MV1=MEMORY LOCATION O
00023P 0008 31   8C F4                 LEAY  MV2,PCR   *MV2=MEMORY LOCATION O
00024P 000A 33   8C F2                 LEAU  M32P,PCR  *M32P=MEMORY LOCATION

00026P 000C 6F   C4        A           CLR   U
00027P 000E 6F   41        A           CLR   1,U
00028P 0010 A6   01        A           LDA   1,X       *V1L-->A
00029P 0012 E6   21        A           LDB   1,Y       *V2L-->B
00030P 0014 3D                         MUL
00031P 0015 ED   42        A           STD   2,U       *HP(V1L X V2L)-->M32P+
00032P 0017 A6   84        A           LDA   ,X        *V1H-->A
00033P 0019 E6   21        A           LDB   1,Y       *V2L-->B
00034P 001B 3D                         MUL
00035P 001C E3   41        A           ADDD  1,U
00036P 001E ED   41        A           STD   1,U       *HP(V1H X V2L)-->M32P+
00037                                                  *LP(V1H X V2L)+HP(V1L
00038P 0020 24   02        0024        BCC   NEXT1
00039P 0022 6C   C4        A           INC   U         *STORE C=1 IN M32P
00040P 0024 A6   01        A NEXT1     LDA   1,X       *V1L-->A
00041P 0026 E6   84        A           LDB   ,Y        *V2H-->B
00042P 0028 3D                         MUL
00043P 0029 E3   41        A           ADDD  1,U
00044P 002B ED   41        A           STD   1,U       *HP(V1L X V2H)+HP(V1H
00045                                                  *LP(V1L X V2H)+LP(V1H
00046P 002D 24   02        0031        BCC   NEXT2
00047P 002F 6C   C4        A           INC   U         *ADD C=1 TO (M32P)
00048P 0031 A6   84        A NEXT2     LDA   ,X        *V1H-->A
00049P 0033 E6   04        A           LDB   ,Y        *V2H-->B
00050P 0035 3D                         MUL
00051P 0036 E3   C4        A           ADDD  U
00052P 0038 ED   C4        A           STD   U         *HP(V1H X V2H)-->M32P,
00053                                                  *LP(V1H X V2H)+HP(V1L
00054P 003A 39                         RTS
00055                                  END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

```
PAGE  001  MULP     SA:0  MULP

00001                          *   16X16 MULTIPLY-RELOCATABLE IN MACHINE COD
00002                          *MULTIPLY TWO 16-BIT POSITIVE VALUES
00003                          *TO GENERATE A 32-BIT PRODUCT. AT TER-
00004                          *MINATION, BOTH INPUT VALUES AND THE RE-
00005                          *SULT WILL BE IN MEMORY.
00006                          *(V1H,V1L)X(V2H,V2L)=
00007                          +                          HP(V1H X V
00008                          +                          HP(V1L X V
00009                          +                        +HP(V1H X V2H),LP(V1H X V
00010                          +                        ------------------------
00011                          +                          32-BIT PRODUCT
00012                          *V1=16-BIT FIRST VARIABLE
00013                          *V2=16-BIT SECOND VARIABLE
00014                          *H=HIGH BYTE
00015                          *L=LOW BYTE
00016                          *P=PRODUCT
00017                                NAM   MULP
00018A 2000                          ORG   $2000
00019A 2000          2244    A MV1   FDB   $2244
00020A 2002          1122    A MV2   FDB   $1122
00021A 2004          0000    A M32P  FDB   0,0
00022A 2006 30   8C F5         SETUP  LEAX  MV1,PCR   *MV1=MEMORY LOCATION O
00023A 2008 31   8C F4                LEAY  MV2,PCR   *MV2=MEMORY LOCATION O
00024A 200E 33   8C F2                LEAU  M32P,PCR  *M32P=MEMORY LOCATION

00026A 2011 6F   C4          A       CLR   U
00027A 2013 6F   41          A       CLR   1,U
00028A 2015 A6   01          A       LDA   1,X       *V1L-->A
00029A 2017 E6   21          A       LDB   1,Y       *V2L-->B
00030A 2019 3D                       MUL
00031A 201A ED   42          A       STD   2,U       *HP(V1L X V2L)-->M32P+
00032A 201C A6   84          A       LDA   X         *V1H-->A
00033A 201E E6   21          A       LDB   1,Y       *V2L-->B
00034A 2020 3D                       MUL
00035A 2021 E3   41          A       ADDD  1,U
00036A 2023 ED   41          A       STD   1,U       *HP(V1H X V2L)-->M32P+
                                                     *LP(V1H X V2L)+HP(V1L
00037                          +
00038A 2025 24   02    2029          BCC   NEXT1
00039A 2027 6C   C4          A       INC   U         *STORE C=1 IN M32P
00040A 2029 A6   01          A NEXT1 LDA   1,X       *V1L-->A
00041A 202B E6   A4          A       LDB   V         *V2H-->B
00042A 202D 3D                       MUL
00043A 202E E3   41          A       ADDD  1,U
00044A 2030 ED   41          A       STD   1,U       *HP(V1L X V2H)+HP(V1H
                                                     *LP(V1L X V2H)+LP(V1H
00045                          +
00046A 2032 24   02    2036          BCC   NEXT2
00047A 2034 6C   C4          A       INC   U         *ADD C=1 TO (M32P)
00048A 2036 A6   84          A NEXT2 LDA   X         *V1H-->A
00049A 2038 E6   A4          A       LDB   V         *V2H-->B
00050A 203A 3D                       MUL
00051A 203B E3   C4          A       ADDD  U
00052A 203D ED   C4          A       STD   U         *HP(V1H X V2H)-->M32P,
                                                     *LP(V1H X V2H)+HP(V1L
00052                          +
00054A 203F 39                       RTS
00055                                END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

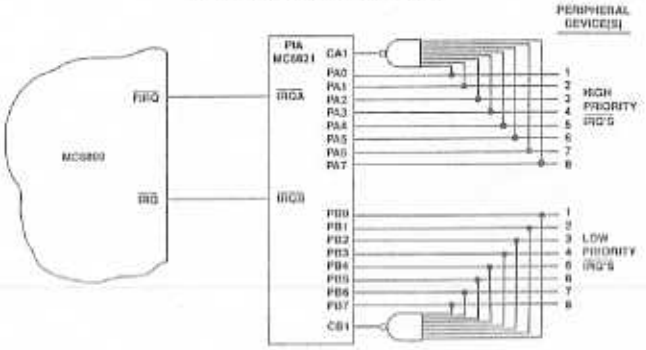

PRIORITIZED INTERRUPTS VIA FIRQ



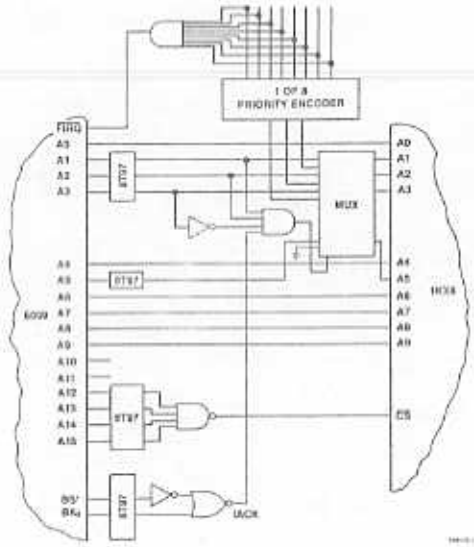
PRIORITIZED INTERRUPTS VIA IRQ



INTERRUPT POLLING VIA PIA



MC6809 VECTORED INTERRUPTS



PAGE 001 POLLF .SA:0 POLLF

```
00001                                       POLLF
00002                                 NAM   REL
00003                                 OPT   POLLF
00004                                 XDEF  FSR1, FSR2, FSR3, FSR4
00005                                 XREF  FSR5, FSR6, FSR7, FSR8
00006                                 XREF  PIAAD
00007P 0000                           PSCT
00008P 0000 B6 0000       A POLLF     LDA   PIAAD
00009P 0003 43                        COMA
00010P 0004 85 0F           A         BITA  #$F
00011P 0006 27 19         0021        BEQ   UPNIB
00012P 0008 85 03           A         BITA  #3
00013P 000A 27 0B         0017        BEQ   UPPAIR
00014P 000C 85 01           A         BITA  #1
00015P 000E 27 04         0014        BEQ   TWO
00016P 0010 7E 0000         A         JMP   FSR1
00017P 0013 3B                        RTI
00018P 0014 7E 0000       A TWO       JMP   FSR2
00019P 0017 85 04         A UPPAIR    BITA  #4
00020P 0019 27 03         001E        BEQ   THREE
00021P 001B 7E 0000         A         JMP   FSR3
00022P 001E 7E 0000         A THREE   JMP   FSR4
00023P 0021 85 30         A UPNIB     BITA  #$30
00024P 0023 27 0A         002F        BEQ   HIPAIR
00025P 0025 85 10           A         BITA  #$10
00026P 0027 27 03         002C        BEQ   FIVE
00027P 0029 7E 0000         A         JMP   FSR5
00028P 002C 7E 0000         A FIVE    JMP   FSR6
00029P 002F 85 40         A HIPAIR    BITA  #$40
00030P 0031 27 03         0035        BEQ   HIBIT
00031P 0033 7E 0000         A         JMP   FSR7
00032P 0036 7E 0000         A HIBIT   JMP   FSR8
00033                                 END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

PAGE 001 POLLI .SA:0 POLLI

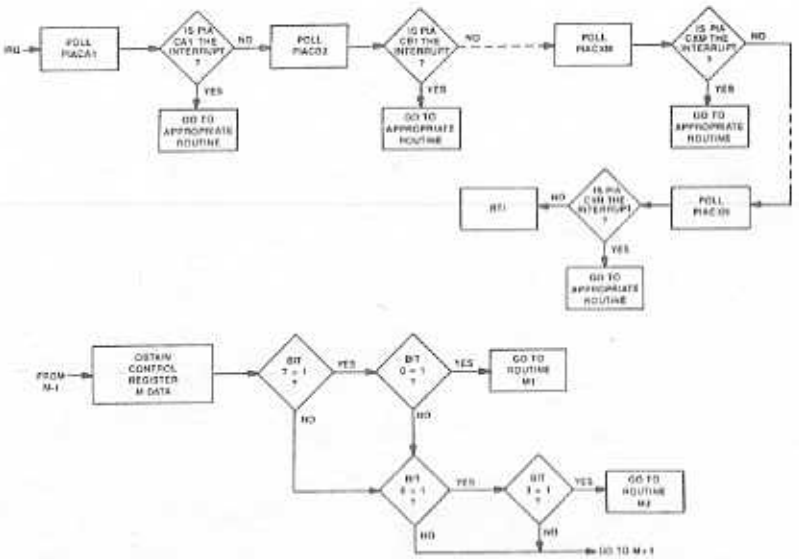
```
00001                                       NAM   POLLI
00002                                       OPT   REL, NOOBJ
00003                                       XREF  ISR1, ISR2, ISR3, ISR4
00004                                       XREF  ISR5, ISR6, ISR7, ISR8
00005                                       XREF  PIABD
00006P 0000                                 PSCT
00007P 0000 C6 10           A POLLI         LDB   #$10
00008P 0002 30 8D 000D                      LEAX  TABLEI, PCR
00009P 0006 96 0000         A               LDA   PIABD
00010P 0009 48                AGAINI        LSLA
00011P 000A 24 05           0011            BCC   JUMPI
00012P 000C C0 02           A               SUBB  #2
00013P 000E 26 F9           0009            BNE   AGAINI
00014P 0010 3B                              RTI
00015P 0011 6E 95             A JUMPI       JMP   [B, X]
00016P 0013       0000        A TABLEI      FDB   ISR1, ISR2, ISR3, ISR4
00017P 001B       0000        A             FDB   ISR5, ISR6, ISR7, ISR8
00018                                       END
```

TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PERIPHERAL POLLING

- STRAIGHT-LINE METHOD RESULTS IN FASTEST POLLING ROUTINE; ALSO NORMALLY USES THE MOST MEMORY.
- MINIMUM INSTRUCTION ROUTINE MEANS MORE TIME REQUIRED FOR POLLING
- DEPENDING UPON USER CONSTRAINTS SUCH AS NUMBER OF PERIPHERALS, INTERRUPT RESPONSE TIME, AVAILABLE MEMORY, ETC., EITHER TYPE OF ROUTINE CAN BE USED, OR PERHAPS A COMBINATION OF THE TWO WILL BE OPTIMUM.

PIA POLLING — METHOD (1)



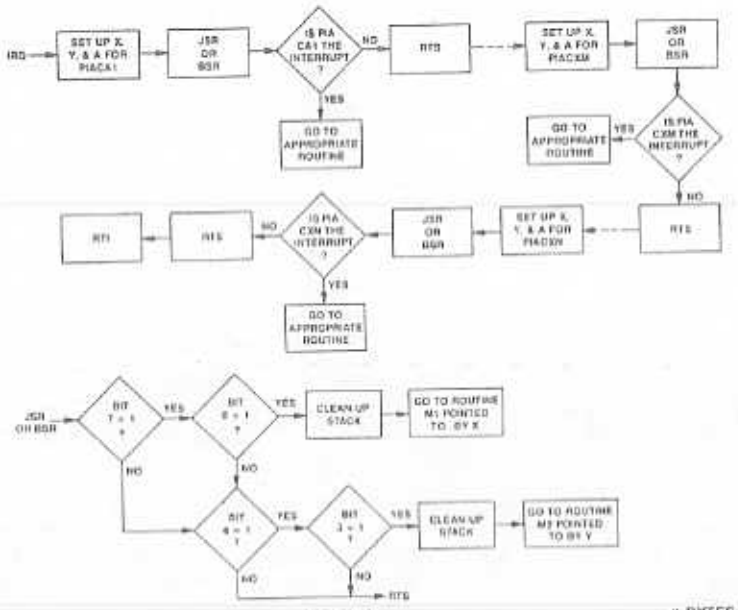
	# BYTES	~
POLLM LDA PIACXM	3	5
BPL CHECK 2	2	3
BIT A #1	2	2
BNE ROUTM1	2	3
CHECK2 ROLA	1	2
BPL POLLM1	2	3
AND A #$10	2	2
BNE ROUTM2	2	3
	16	23
POLLM1		
⋮		
RTI	1	
	23N + 15	

TOTAL # BYTES = 16N + 1

WHERE
N = # CONTROL REGISTERS
2N = # SERVICE ROUTINES

$\frac{N}{2}$ = # OF PIA'S

PIA POLLING — METHOD ②



		# BYTES
POLLM	LDX #ROUTM1	3
	LDY #ROUTM2	4
	LDA PIACXN	3
	JSR CHECK	3
	⋮	
	RTI	1
		13

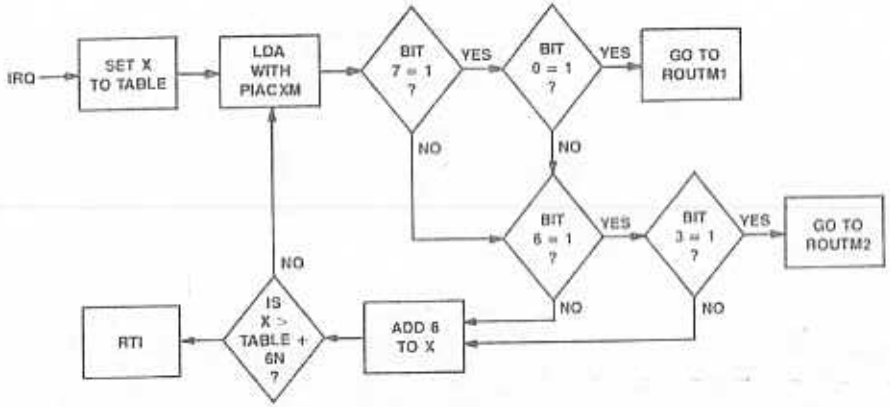
		# BYTES
CHECK	BPL CHECK2	2
	BIT A # 1	2
	BEQ CHECK2	2
	LEA S 2,S	2
	JMP 0,X	2
CHECK2	ROLA	1
	BPL NEXT	2
	AND A # $10	2
	BEQ NEXT	2
	LEAS 2,S	2
	JMP 0,Y	2
NEXT	RTS	1
		22

TOTAL # BYTES = 22 + 1 + 12N
METHOD 2 IS MORE BYTE EFFICIENT THAN METHOD 1 WHEN

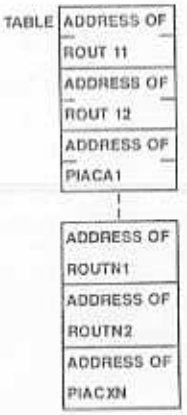
$$22 + 1 + 13N < 16N + 1$$
$$22 < 3N$$
$$7 < N$$

E3037

PIA POLLING — METHOD ③



		# BYTES	~
POLLM	LDX # TABLE	3	3
	LDA [4,X]	2	6
	BPL CHECK2	2	3
	BIT A # 1	2	2
	BEQ CHECK2	2	3
	JMP [0,X]	2	6
CHECK2	ROLA	1	2
	BPL NEXT	2	3
	ANDA # $10	2	2
	BEQ NEXT	2	3
	JMP [2,X]	2	7
NEXT	LEAX 6,X	2	5
	CMPX # TABLE + 6N	3	4
	BNE POLLM + 3	2	3
	RTI	1	15
		30	3 + 61N + 15

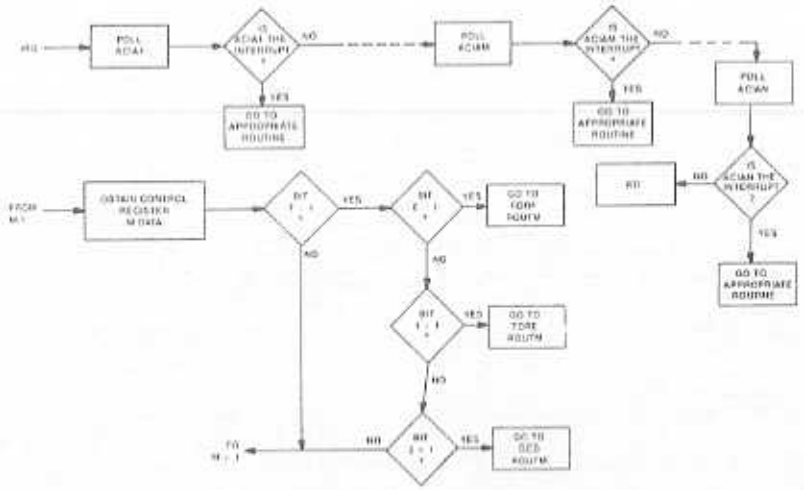


TOTAL # BYTES = 6N + 30
METHOD 3 IS MORE BYTE EFFICIENT THAN METHOD 1 WHEN

$$6N + 30 < 16N + 1$$
$$29 < 10N$$
$$2 < N$$

E3038

ACIA POLLING — METHOD (1)

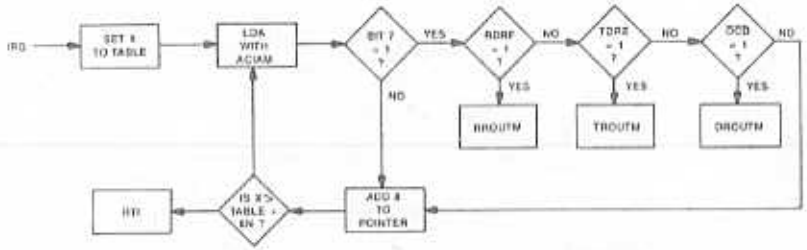


		# BYTES
POLLM	LDAA ACIAM	3
	BPL NEXT	2
	ORAA # $10	2
	TAP	1
	BCS RROUTM	2
	BVS TROUTM	2
	BEQ DROUTM	2
NEXT		14
	RTI	1

TOTAL # BYTES = 14N + 1

N = # CONTROL REGISTERS
= # ACIA'S

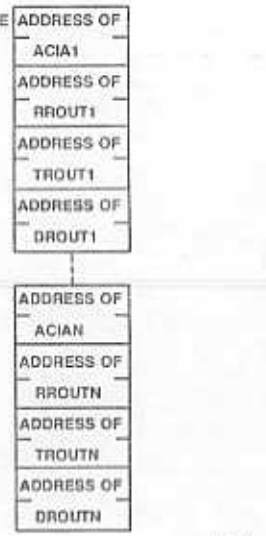
ACIA POLLING — METHOD (2)



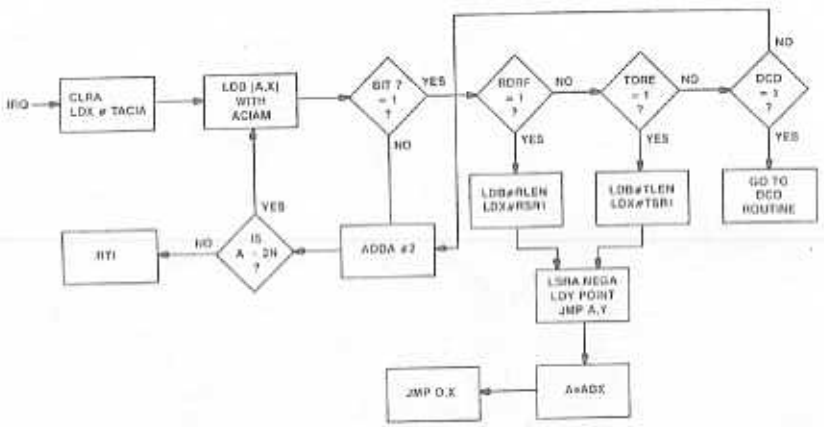
		# BYTES
POLLM	LDX # TABLE	3
LDA	[0,X]	2
BPL	NEXT	2
RORA		1
BCC	OVFL	2
JMP	[2,X]	2
OVFL	RORA	1
BCC	ZERO	2
JMP	[4,X]	2
ZERO	RORA	1
BCC	NEXT	2
JMP	[6,X]	2
NEXT	LEAX 8,X	2
CMPX	# TABLE + 8N	3
BNE	POLLM + 3	2
RTI		1
		30

TOTAL # BYTES = 8N + 30
METHOD 2 BECOMES MORE BYTE EFFICIENT THAN METHOD 1 WHEN

8N + < 14N + 1
29 < 6N
4 < N



ACIA POLLING — METHOD 3




```
                          # BYTES
POLLM   CLR A               1
        LDX #TACIA          3
NEXT    LDB (A,X)           2
        BMI CHECK           2
MOVEON  ADDA #2             2
        CMPA #2N            2
        BNE NEXT            2
        RTI                 1
CHECK   RORB                1
        BCC CHECK1          2
        LDB # RLEN          2
        LDX # RSR1          3
SETUP   LSRA                1
        NEGA                1
        LDY POINT           3
        JMP A,Y             2
CHECK 1 RORB                1
        BCC CHECK2          2
        LDB # TLEN          2
        LDX # TSR1          3
        BRA SETUP           2
CHECK 2 RORB                1
        BCC MOVEON          2
        JMP DCD             3
                           45

        ABX
        ABX
POINT-2 ABX
POINT-1 ABX

POINT   JMP 0,X

                            3 + N - 1
```

TACIA: ADDRESS OF ACIA 1 / ADDRESS OF ACIA 2 / ADDRESS OF ACIA N

RSR1, RLEN, RSR2, RSR3, RSRN

TSR1, TLEN, TSR2, TSR3, TSRN

DCD

$3 + N - 1 + 2N + 45 < 14N + 1$

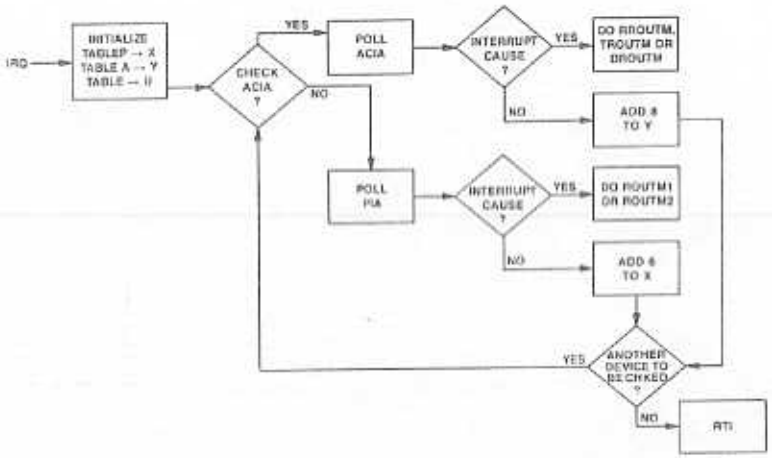
$5N + 48 < 14N + 1$

$47 < 11N$

$4 < N$

TR5175

PIA AND ACIA POLLING



N = TOTAL # CONTROL REGISTERS
N_A = #ACIA CONTROL REGISTERS
N_P = #PIA CONTROL REGISTERS
IN TABLE, 0 MEANS CHECK PIA CONTROL REGISTER
1 MEANS CHECK ACIA CONTROL REGISTER

TABLE

1	0	0	1	0	1	0	1	
0	1	0	0	1	0	1	1	
				0	0	1	1	

$$\text{TOTAL \# BYTES} = 81 + 6N_P + 8N_A + \frac{N}{8}$$

THIS METHOD IS MORE BYTE EFFICIENT THAN COMBINING BOTH METHOD 2 FOR ACIA'S AND METHOD 3 FOR PIA'S WHEN

$$81 + 6N_P + 8N_A + \frac{N}{8} < 16N_P + 14N_A + 1$$

$$80 < 9\tfrac{7}{8}N_P + 5\tfrac{7}{8}N_A$$

N_A \ N_P	2	4	6	8	10	12
1	N	N	N	Y	Y	Y
2	N	N	N	Y	Y	Y
3	N	N	N	Y	Y	Y
4	N	N	Y	Y	Y	Y
5	N	N	Y	Y	Y	Y
6	N	Y	Y	Y	Y	Y

E8242