gri-909

eir devices

GRI Computer Corporation
320 NEEDHAM STREET, NEWTON, MASSACHUSETTS 02164

Price $3.75

GRI-909

EIR DEVICES MANUAL

GRI Computer Corporation, 320 Needham Street, Newton, Massachusetts 02164

10-40-002-B
0471-500

TABLE OF CONTENTS

1	EIR MODE PROCESSING	
1.0	Introduction	1-1
1.1	GRI-909 Instruction Cycle	1-2
1.2	The EIR Channel	1-3
2	INSTRUCTIONS	
2.0	Fetching Micro-Instructions	2-1
2.1	Executing EIR Instructions	2-1
2.2	Terminating the EI State	2-1
2.3	Major State Control	2-2
2.4	Permissible Instructions	2-4
2.5	Conditional Jumping in the EIR Mode	2-4
3	THE EXTENDED ARITHMETIC OPERATOR	
3.0	Introduction	3-1
3.1	Installation	3-1
3.2	Preliminary Checkout	3-1
4	THE MEDIUM SPEED MULTIPLY OPERATOR (MPO)	
4.0	Medium Speed Multiply (MPO)	4-1
4.1	MPO Micro Program	4-2
4.2	MPO Address State Flow and Timing	4-2
4.3	Utilization of MPO with $SXP	4-3
5	THE MEDIUM SPEED DIVIDE OPERATOR (DVO)	
5.0	Medium Speed Divide (DVO)	5-1
5.1	Divide Check	5-2

5.2 DVO Micro Program 5-2

5.3 DVO Address State Flow and Timing 5-3

5.4 Utilization of DVO with $SDV 5-5

6 THE ARITHMETIC RIGHT SHIFT (ARS)

6.0 Arithmetic Right Shift (ARS) 6-1

6.1 ARS Micro Program 6-1

6.2 ARS Address State Flow and Timing 6-2

6.3 Programming 6-4

7 THE NORMALIZE OPERATOR (NORM)

7.0 Normalize (NORM) 7-1

7.1 NORM Micro Program 7-2

7.2 NORM Address State Flow and Timing 7-3

7.3 Programming 7-3

CHAPTER ONE

EIR MODE PROCESSING

1.0 Introduction:

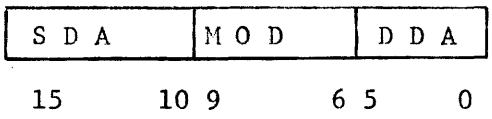
GRI External Instruction Request devices are devices that operate in the EIR mode. In the EIR mode of processing, main memory-reference processing is temporarily suspended while a device in the system other than main memory presents 16-bit words directly to the Instruction Register to be executed as instructions. EIR mode processing takes control priority over the GRI 909's Program and Program-Interrupt modes of processing, but the EIR mode itself can be overridden by the 909's DMA mode of processing.

In the EIR mode, instructions are executed one after the other in groups of two, with each instruction taking only 880 nanoseconds which is half the normal instruction cycle time. The EIR device "borrows" registers from other devices to use as part of itself and thereby performs certain hybrid operations more economically than could be done by completely self-contained hardware modules.

Examples of EIR devices are the Medium Speed Multiply Operator (MPO), the Medium Speed Divide Operator (DVO), the Arithmetic Right Shift Operator (ARS), and the Normalize Operator (NORM), which make use of the Arithmetic Operator, the Six General Purpose Registers option, and the Extended Arithmetic Operator, to produce inexpensive hardware devices for performing hardware multiplication and division, double-precision shift and normalize, and firmware double-precision and floating point arithmetic. These optional operators are standard features of the GRI-909, Model 40.

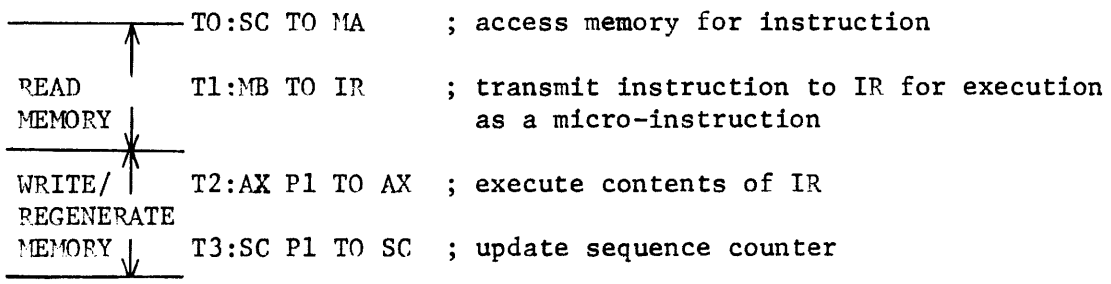
1.1 GRI-909 Instruction Cycle:

The GRI-909 processor utilizes a basic register reference instruction as a micro-instruction. Groups of four of these micro instructions make up the basic macro-instruction cycle. All of the net results of each major state in the GRI-909 may be expressed in terms of these register reference micro-instructions. Each micro cycle requires 440 nanoseconds for execution. The format of the register reference macro-instruction, consisting of 16 bits is:



Groups of such micro-instructions also accomplish memory referencing in the GRI-909. This occurs when the first micro-instruction of a set of four places a data word (used as an address) in the memory address register. Thus, a SC TO MA micro-instruction causes the memory system to start on a memory reference.

A typical stream of micro-instructions that occurs for a register reference macro-instruction, such as AX P1 TO AX (RS AX, P1) is:



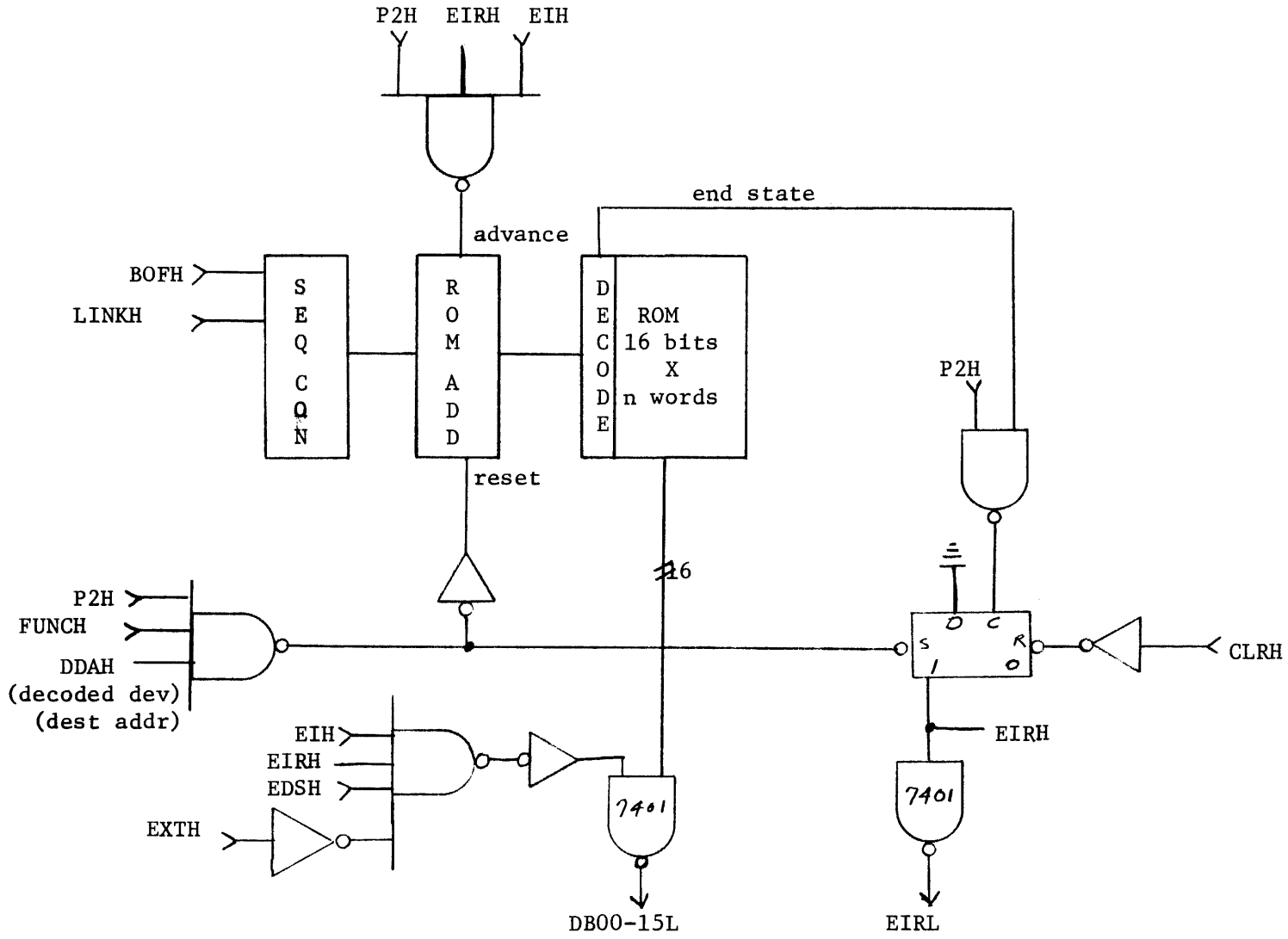
Thus, all macro-instructions are implemented by executing streams of micro-instructions whose net effect at the end of the instruction

is the execution of that instruction. The streams of micro-instructions are retrieved from a ROM located on PC-1. This ROM is addressed by the micro-time slots T0,T1,T2, and T3 on one axis, and the major states FI, FA, FO, FD, etc. on the other axis. Each juncture of a micro-time slot and a major state causes another micro-instruction to be executed from ROM.

1.2 The EIR Channel:

The GRI-909's EIR channel permits the ROM used for controlling the processor to effectively be extended. To simplify the use of the channel, the EIR device is restricted to register-reference micros only. These micros are actually executed out of the instruction register rather than directly out of the ROM added in the EIR device. The EIR device is generally activated by the execution of a function output instruction such as STRT TO MP0 (F0 STRT, MP0). The F0 control pulse causes the EIR flip-flop to set in the EIR device. The EIR flip-flop is gated onto the EIRL bus, causing an EI request to be present when the processor completes the execution of the F0 instruction. This request is granted at the completion of the F0 instruction, and the processor enters the EI major state. The timing of the processor now switches over to a double time cycle that causes a fetch-execute pair of micro-cycles every 880 nanoseconds.

Figure 1-1 shows typical EIR device logic.



TYPICAL EIR DEVICE LOGIC

Figure 1-1

CHAPTER TWO

EIR INSTRUCTIONS

2.0 Fetching Micro-Instructions:

Every other micro-cycle during the EI state, the processor sets up and executes the micro-instruction

EDSH* TO IR

*generic address (15_8) for EXTERNAL DATA.

The call for external data causes the activated EIR device to gate the next instruction from its ROM onto the DB lines. The instruction word is then transferred to the IR.

2.1 Executing EIR Instructions:

After the instruction has been fetched from the EIR device's ROM into the IR, it is now executed out of the IR during the next 440 nanosecond period. Thus, a single machine cycle (1.76 microseconds) now consists of a pair of EI fetch and executes, e.g:

```
T0:  EDSH TO IR    ; fetch micro-instruction
T1:  (EXECUTE C(IR))
T2:  EDSH TO IR    ; fetch micro-instruction
T3:  (EXECUTE C(IR))
```

2.2 Terminating the EI State:

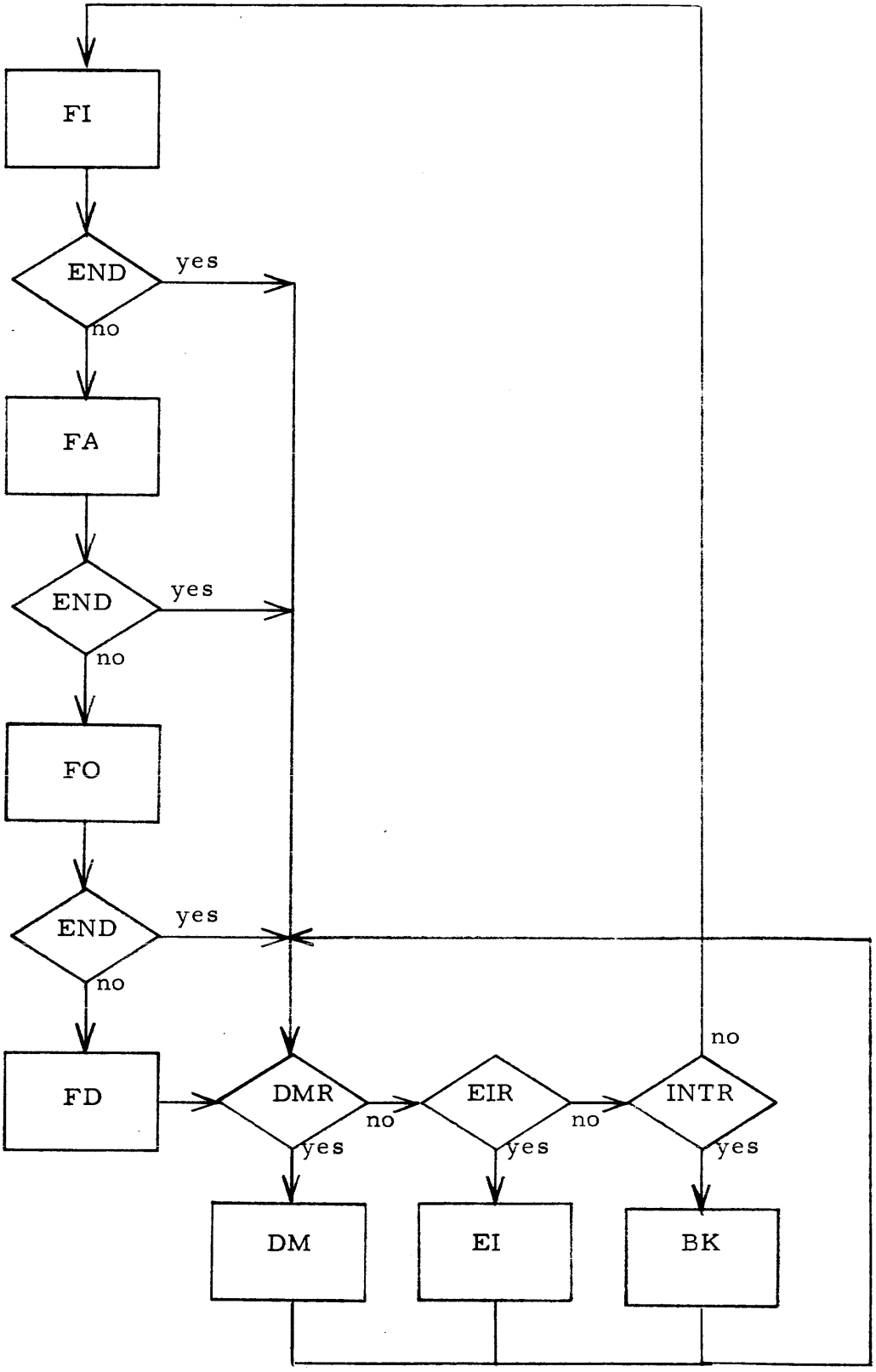
When the EIR device has completed its last ROM access (T0 or T2), it has simply to reset its EIR flip flop. With the EI request gone, the processor will now terminate the EI state at the end of T3 and enter the next major state (usually the FI state).

2.3 Major State Control:

Fig. 2-1 describes all possible major states in the GRI-909 as well as state flow when terminating each major state. Note that entering the EI state effectively prolongs the instruction that caused entering the EI mode. If the EI instruction were built into the processor's internal ROM, it would execute at a higher rate, since no fetch cycle is required; but it would still appear as an n cycle instruction. The net effect (i.e. prolonged instruction) is the same. Thus, if an interrupt request appears during the EI state, it cannot be processed until the effectively prolonged instruction that caused the EI request has terminated. No other EIR device can interrupt a current EIR device because macro-instructions to activate it cannot be executed during EI processing.

CAUTION: One EIR device should not activate another EIR device, unless its ROM is desensitized during the second device's execution. This can be built into the EIR logic so that EIR devices can call other EIR devices much the same as subroutines.

Direct Memory Access requests (DMRs) will be processed within at most one cycle after the DMR request is generated. This is because the memory is not being used during EI execution and is available at the end of any T3 state. The DM state does not effect machine states so that a return to the EI state after terminating the DM state allows EI processing to resume where it was interrupted.



GRI 909 MAJOR STATE FLOW

Figure 2-1

2.4 Permissible Instructions:

Any register reference instruction is permissible.

Any function output instruction is permissible (if the FO activates another EIR device, see section 1.6 for CAUTION).

Memory reference instructions (MB (06)=SDA or DDA) are not permissible.

*Jump instructions (TRP (03)=DDA) are not permissible.

*Skip instructions (02=DDA) are not permissible.

*These operations refer to manipulation of the sequence counter which is not in use during execution of our EI string. The effective sequence counter for the EIR micro string is the local address register used to address the ROM in the EIR device.

2.5 Conditional Jumping in the EIR Mode:

A conditional jump or skip in the EIR mode consists of changing the contents of the local address register that addresses the EIR ROM as a function of some logic condition.

EXAMPLE:

ROM ADDR6 = (ROM ADDR3) · (LINKH)

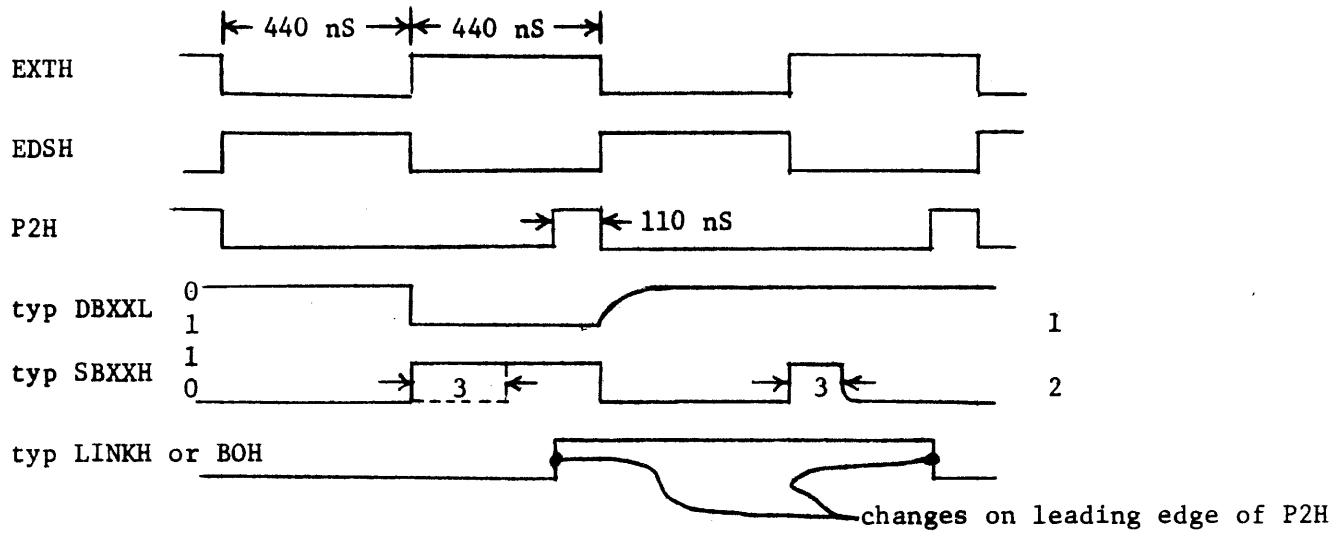
ROM ADDR15 = (ROM ADDR 14) · (BOFL)

etc.

The testing of data in the EIR mode is done as the data is in transit from one device to another. The timing chart (Figure 2-3) gives some timing criteria for the examination of link or bus overflow lines and SB or DB data lines. Typical tricks that combine data testing operations with the normal movement of data in a register operation are given in Figure 2-4.

In Example 1, the sign of AX before the shift operation is required for later use in the ROM program to cause a conditional jump in the ROM address register. During the left shift operation, the data on SB15H is actually what was in AX14 and the data on the LINKH line is what was in AX15, but only after the leading edge of P2H. Therefore, the clock to FF AXLTZ is an inverted P2H to guarantee that the LINK has assumed the proper value before capturing it in AXLTZ. The signal AnH is an address state of the ROM address register so that the test FF (AXLTZ) is locked at the correct instruction in the ROM.

In Example 2, the sign of AY is required after the shift operation is performed. Here again, the data from AO14 will be shifted left onto SB15 and become AX15 with the leading edge of P2H. We therefore, simultaneously clock SB15H into the control FF AYLTZ on the leading edge of P2H and the ROM address state AnH.

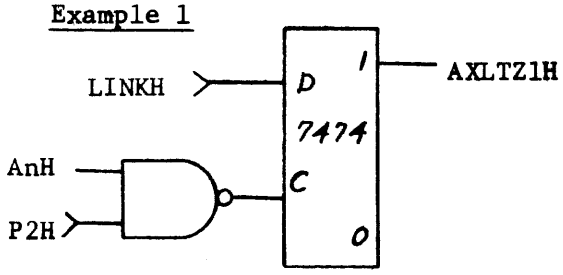


1. source device gates data on DB lines
2. data appears on SB lines for destination device
3. bus modifier delay dependent on path chosen

EIR TIMING

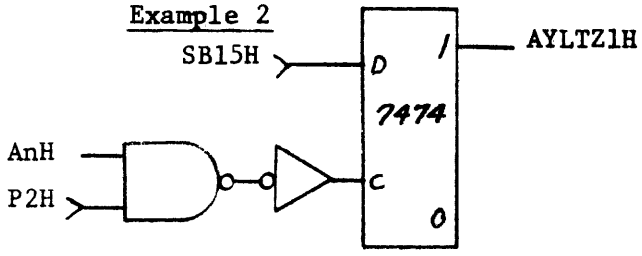
Figure 2-3

Example 1




```
RS AX,L1 ;check sign of AX at same time
         ;for later use in ROM program
```

Example 2




```
RR AO,L1,AY ;check sign of AY after
            ;operation for later use
            ;in ROM program
```

COMBINING OPERATIONS

Figure 2-4

CHAPTER THREE

THE EXTENDED ARITHMETIC OPERATOR

3.0 Introduction:

The Extended Arithmetic Operator is a standard feature of the GRI-909 model 40. It may, however, be purchased separately and plugged into any model 909 provided that model also has an Arithmetic Operator and Six General Purpose Registers installed in it.

3.1 Installation:

The EAO is contained on one of the standard large operator boards and is plugged into either of the two option slots in the front of the machine. (The two slots to the right of the AO.) Make sure that any interrupt chain jumpers are removed before inserting the board.

3.2 Preliminary Checkout:

After inserting the card, the diagnostic program DEMO should be run. This will require a teletype, CRT terminal, or some other "teletype like" output device with the address 77. There are so many numbers to examine should a failure occur that the various arguments and register contents are printed by the diagnostic.

CHAPTER FOUR

THE MEDIUM SPEED MULTIPLY OPERATOR (MPO)

4.0 Medium Speed Multiply (MPO):

The MPO option is an EIR device that executes an iterative string of micro-instructions which manipulate the AO and GR1 to accomplish a single precision, unsigned multiply.

The operator expects the following conditions prior to starting:

1. C (AX) = multiplicand (positive magnitude)
2. C (AY) = multiplier (positive magnitude)
3. C (LINK) = sign of result

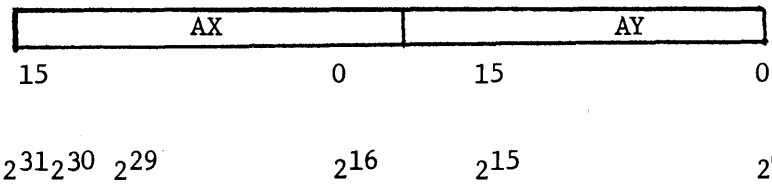
The operator is started by the command

```
FO 1,14  02 0001 14
```

The operator completes its execution leaving the following states:

1. C (AX) = MSH product (bits 0 - 13)
2. C (AY) = multiplier
3. C (GR1) = LSH product (bits 0-15)
4. C (LINK) = sign of result
5. AO STATE = ADD

The AX and AY are considered a double precision word as follows:



4.1 MPO Micro Program:

MPO ROM ADDR	CONTENTS	
0	FOA ADD	
1	RR AX, GR1	
2	ZR AX	
3	RS 30, R1	;LOOP
4	RR MSR, R1, 0	
5	RR AO, AX	
6	RS AX, R1	
7	RS 30, R1	

4.2 MPO Address State Flow and Timing:

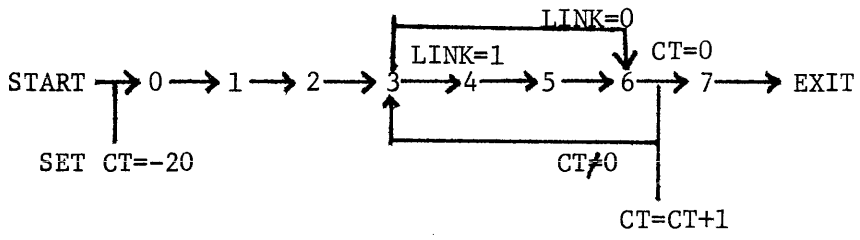
Figure 4-1 is a state flow chart which shows the sequence control of the MPO ROM address. The timing becomes a function of the number of "1" bits in the multiplier (C(GR1)). Each time a "1" is shifted off the right end of GR1, the inner loop time is 3.52 microseconds (4x.880). Each time a "0" is shifted off, the inner loop time is 1.76 microseconds.

The maximum loop time occurs if the C(GR1) is a -1 (177777), causing an inner loop time of 16 x 3.52 = 56.32 microseconds. The lead in code to the loop (steps 0-2) and the exit code (step 7) require another 4 x 880 = 3.52us . The maximum possible execution time then is 59.84 microseconds. The minimum execution occurs for

a 0 multiplier. The inner loop time is 16 x 1.76 = 28.16 microseconds. With the lead in and exit time, the total minimum time is 31.68 microseconds. The average execution time is 45.76 microseconds.

MPO Timing Summary:

MIN. TIME	31.68 microseconds
AVG. TIME	45.76 microseconds
MAX. TIME	59.84 microseconds



MPO ADDRESS STATE DIAGRAM

Figure 4-1

4.3 Utilization of MPO with $SXP:

The single precision, fixed point multiply routine may be changed to utilize the MPO as shown by the listing on the following page (Fig. 3-2). Times involved are:

MIN. TIME	52.8 microseconds	
AVG. TIME	71.28 microseconds	
MAX. TIME	89.76 microseconds	(positive multiplicand max. negative multiplier)

These times include sign adjustment on entry and exit, the latter using the double precision complement subroutine $SCT.

```
001                         ; $SMY ($SCT)  USES MPO
002                         ; 74-43-001L
003                         ; MULTIPLY
004                         MPO=14
005 00000 03 0010 06 $SMY:  RMI   TRP,0
    00001 000000
006 00002 02 0001 00        FOM   CLL          ;SIGN FLAG
007 00003 11 1010 03        JC    AX,GEZ,.+4
    00004 000007
008 00005 11 0110 11        RSC   AX,P1        ;ARG1 NEG
009 00006 02 0011 00        FOM   CML
010 00007 12 1010 03        JC    AY,GEZ,.+4
    00010 000013
011 00011 12 0110 12        RSC   AY,P1        ;ARG2 NEG
012 00012 02 0011 00        FOM   CML
013 00013 02 0001 14        FO    STRT,MPO
014 00014 00 0101 02        SFM   NOT LNK      ;CHECK RESULT SIGN
015 00015 00 0100 03        JU    $SCT
    00016 000021
016 00017 00 0101 03        JUD   $SMY+1
    00020 000001
001                         ; $SCT
002                         ; 74-43-005L
003                         ; COMPLEMENT TWICE - PRECISION
004 00021 11 0010 11 $SCT:  RSC   AX
005 00022 12 0110 12        RSC   30,P1
006 00023 00 0011 02        SFM   NOT BOV
007 00024 11 0100 11        RS    AX,P1
008 00025 00 0000 00        NOP
009 00026 03 0000 07        RR    TRP,SC
001                         END
```

UTILIZATION OF MPO WITH $SXP

Figure 4-2

CHAPTER FIVE

THE MEDIUM SPEED DIVIDE OPERATOR (DVO)

5.0 Medium Speed Divide (DVO):

The DVO option is an EIR device that executes an iterative string of micro-instructions which manipulate the AO and GR1 to accomplish a single precision, unsigned divide.

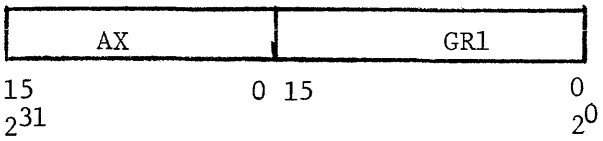
The operator expects the following conditions prior to starting:

1) C (AX) = MSH Dividend (bits 0-13)

2) C (GR1) = LSH Dividend (bits 0-15)

3) C (AY) = Divisor

The AX and GR1 are considered a double precision dividend as follows:



The operator is started by the command

```
FO 2,14    02 0010 14
```

The operator completes execution having the following states:

1) C (GR1) = Quotient

2) C (AX) = Remainder

3) C (LINK) = 1: divide check occurred
= 0: no divide check occurred

4) AO STATE = ADD

5) C (AY) = 2's complement of divisor

5.1 Divide Check:

Divide check is an alarm condition that occurs when an illegal divide is attempted. It signals the user that the results of the divide (quotient and remainder) are meaningless. Divide check can occur in two ways:

1) A divide by 0 is attempted.
2) The MSH of twice dividend is greater than or equal to the divisor.

On exiting from the divide operator, the link must be checked for a divide check alarm, or the validity of the quotient and remainder cannot be guaranteed.

5.2 DVO Micro Program:

DVO ROM ADDR	CONTENTS	
0	FOA ADD	
1	RSC AY,P1	
2	RRC AO,L1,0	;LOOP
3	RS GR1,L1	
4	RR AO,AX	
5	RS AX,L1	
6	RS AX,R1	
7	FOM STL	

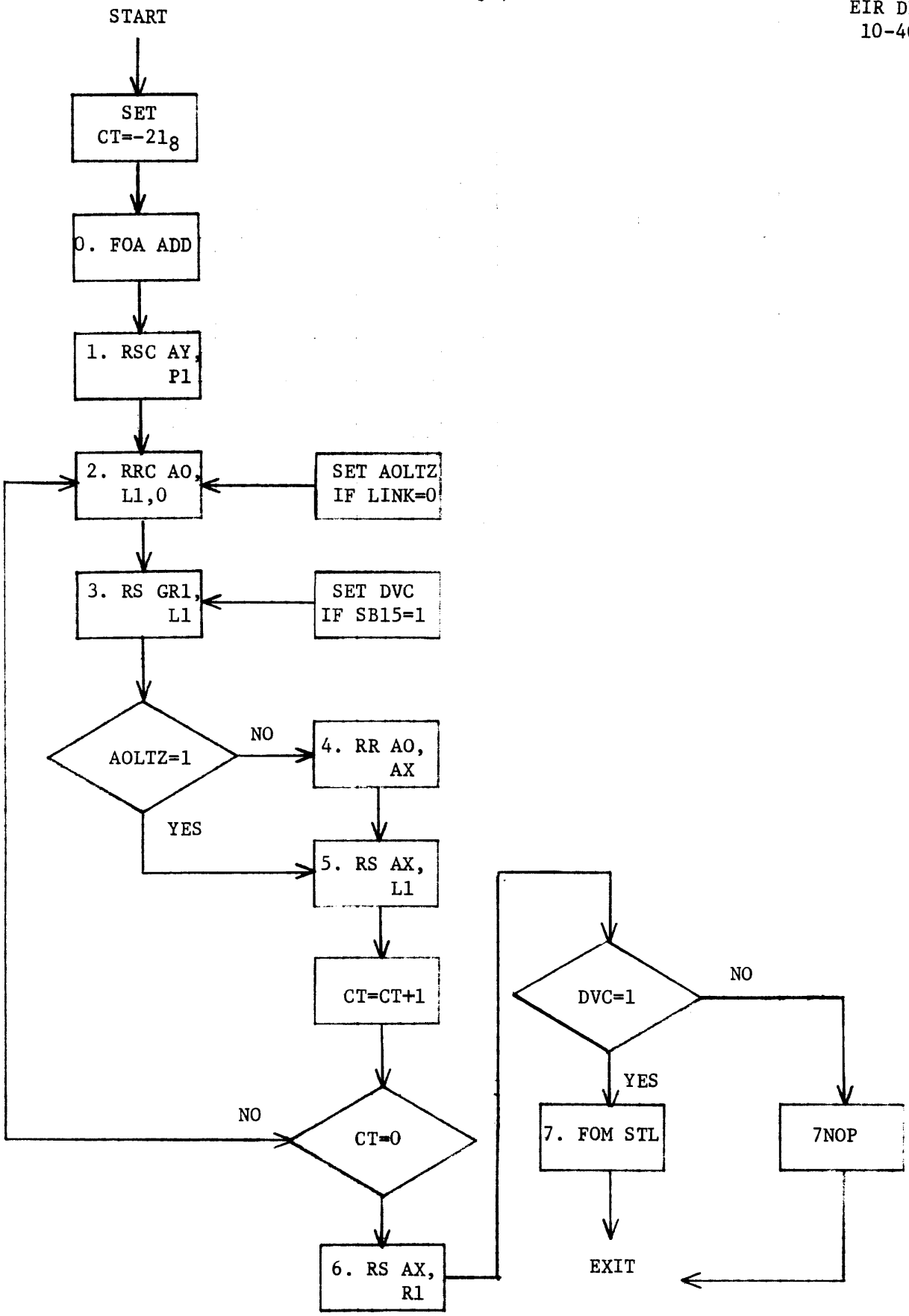
5.3 DVO Address State Flow and Timing:

Figure 5-1 is a state flow chart which shows the control sequence of the DVO ROM address register. The timing is a function of the number of 1's shifted off the left end of A0 in step 2. Each time a 1 is shifted, the inner loop time is 4 micro cycles. Each time a 0 is shifted off the left end of A0 in step 2, the inner loop time is 3 micro cycles. The maximum loop time occurs when 16 1's are shifted off the A0. This can only occur when the sum of AX and the 2's complement of AY is 0 for all 17 shifts of AX. Thus, the worse case loop time is 68 micro cycles. The lead in and exit code requires 4 micro cycles.

Thus, the worse case operator time is 72 micro cycles. The minimum execution occurs if 1's are continually shifted off the A0 in step 2. This leads to 51 micro cycles. With lead in an exit code (with no divide check) time of 4 micro cycles, we have a minimum execution of 55 micro cycles. Since these micro cycles occur in pairs, the exit from the EIR mode will not occur until 56 micro cycles have occurred.

DVO Timing Summary:

	Micro Cycles	Time (microseconds)
MIN. TIME	56	49.28
AVG. TIME	64	56.32
MAX. TIME	72	63.36



DVO STATE FLOW CHART

Figure 5-1

5.4 Utilization of DVO With $SDV:

The single precision fixed point divide route $SDV may be changed to incorporate the DVO as follows:

```
001                         ; $SDV   USES DVO
002                         ; 74-43-002L
003                         ; DIVIDE
004                         DVO=14
005 000000 03 0010 06 $SDV: RMI   TRP,0
    000001 000000
006 000002 11 0000 06       RM    AX,DIVD
    000003 000031
007 000004 02 1000 13       FOA   XOR          ;SET SIGN OF RESULT
008 000005 13 1000 00       RR    A0,L.1,0
009 000006 00 1010 06 SIGN: ZMI   L.1,0
    000007 000000
010 000010 11 1010 03       JC    AX,GEZ,.+7 ;FORCE EVERYTHING POS.
    000011 000017
011 000012 30 0110 30       RSC   30,P1        ;FOR DVO
012 000013 00 0011 02       SFM   NOT ROV
013 000014 11 0110 11       RSC   AX,P1
014 000015 07 0100 07       RS    SC,P1
015 000016 11 0010 11       RSC   AX
016 000017 12 1010 03       JC    AY,GEZ,.+3
    000020 000022
017 000021 12 0110 12       RSC   AY,P1
018 000022 02 0010 14       FO    2,DVO        ;DIVIDE
019 000023 06 0000 03       MR    SIGN+1,TRP
    000024 000007
020 000025 03 0100 03       JC    TRP,EIZ,.+3;SET QUOTIENT SIGN
    000026 000030
021 000027 30 0110 30       RSC   30,P1
022 000030 06 0010 03       MRI   0,TRP        ;USE SIGN OF
    000031 000000
023                         DIVD=.-1           ;DIVIDEND TO
024 000032 03 1010 03       JC    TRP,GEZ,.+3;SET REMAINDER SIGN
    000033 000035
025 000034 11 0110 11       RSC   AX,P1
026 000035 02 0000 13       FOA   ADD
027 000036 00 0101 03       JUD   $SDV+1       ;DIVIDE CHK IF LNK=1
    000037 000001
028                         END
```

UTILIZATION OF DVO WITH $SDV

Figure 5-2

The routine calls $SCT (double precision two's complement) if the dividend is negative. The divisor is also complemented to a positive number if it is negative.

The timing for the complete routine is a complex function of the signs of the operands, the signs of the quotient and remainder, and the absolute value of the operands. We therefore offer the following table as a guide for timing estimates.

Sign	Dividend	Divisor	Quotient	Remainder	Div Check
+	2	1	4	5	
-	11	5	5	5	
					4

FIXED TIME: 27

DVO TIME: MIN 28
AVG 32
MAX 36

1. All times are given in cycles. Multiply the result by 1.76 to find time in usec.
2. If divide check occurs, the times for quotient and remainder correction need not be considered since the routine exits without considering them.

Example:

+ Dividend = 2
- Divisor = 5
- Quotient = 5
Remainder = 5
Fixed Time = 27
DVO Avg Time = 32
Total = 76 cycles = 131.76 u seconds

- Dividend	= 11	
- Divisor	= 5	
+ Quotient	= 4	
Remainder	= 5	
Fixed Time	= 27	
DVO Max Time	= 36	
Total	= 88	cycles = 154.88 u seconds

CHAPTER SIX

THE ARITHMETIC RIGHT SHIFT OPERATOR (ARS)

6.0 Arithmetic Right Shift (ARS):

The ARS option is an EIR device that executes a string of micro-instructions which manipulate AX, AY, and GR1 to perform an n bit, arithmetic shift of AX and AY (considered a double precision register) according to the count in GR1.

The operator expects the following conditions prior to starting:

1. C (AX) = MSH operand
2. C (AY) = LSH operand
3. C (GR1) = negative bit shift count

The operator is started by the command

FO 3,14 02 0011 14

The operator completes its execution, leaving the following states:

1. C (AX) = MSH operand/n
2. C (AY) = LSH operand/n
3. C (GR1) = 0 if $-32_{10} \leq n \leq 1$, or n + 32 otherwise
4. C (LINK) = sign of AX
5. BOV = 1

The AX and AY are considered a double precision word as shown in 4.0 of the MPO description.

6.1 ARS Micro Program:

An arithmetic right shift requires that the sign bit (AX15) be unchanged and its value copied into AX14 for each bit shift performed.

Thus, consider the following examples:

a) before shift

GRI = -7 AX=0101101001011010 AY=1101110000000000

after shift

GRI = 0 AX=0000000010110100 AY=1011010110111000

b) before shift

GRI = -5 AX=1110001010110110 AY=0101101011100101

after shift

GRI = 1 AX=1111111100010101 AY=1011001011010111

The micro-program for ARS is:

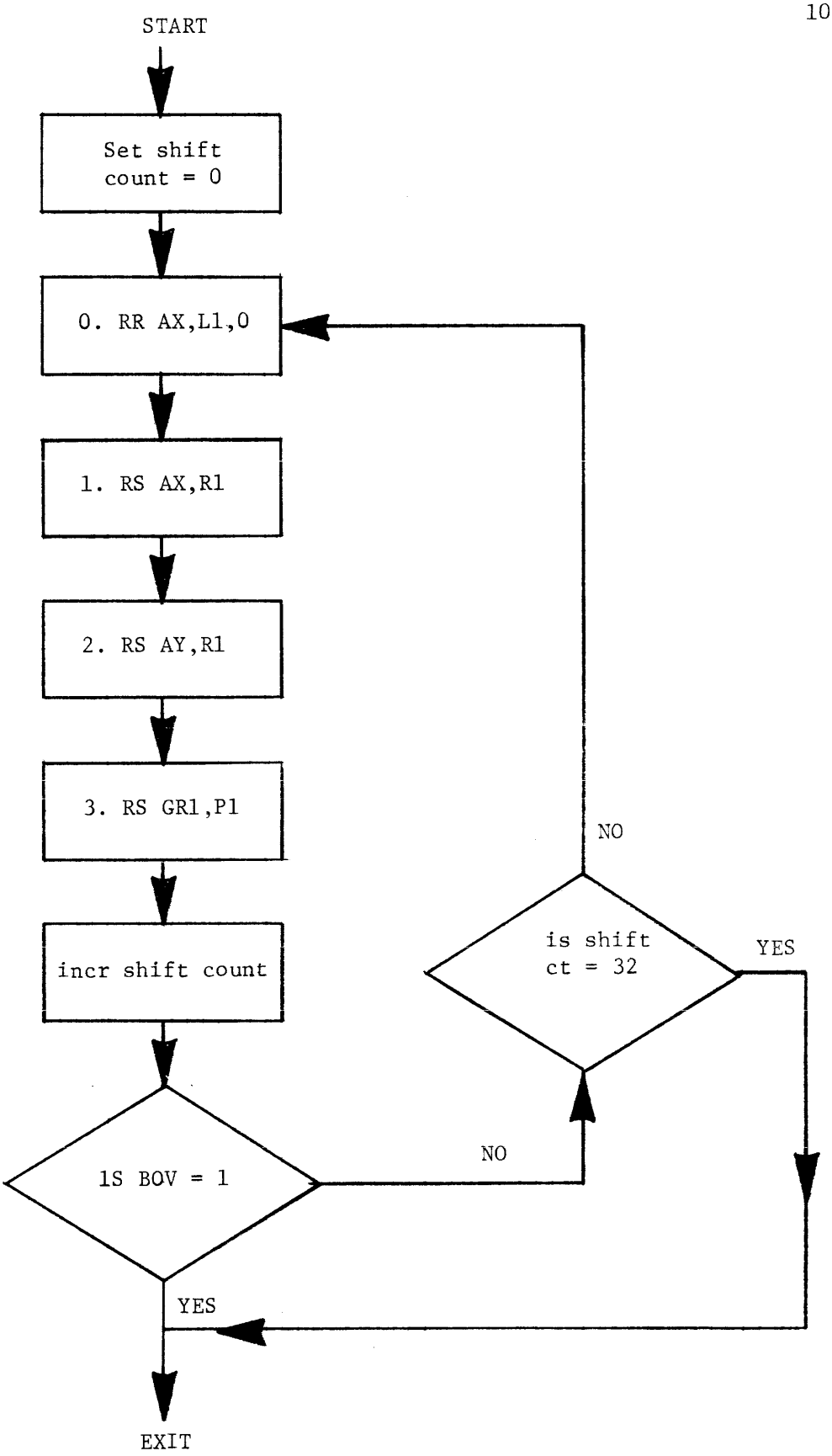
```
0  RR AX,L1,0  ┐
1  RS AX,R1    │ ;LOOP
2  RS AY,R1    │
3  RS 30,P1    ┘
```

6.2 ARS Address State Flow and Timing:

Figure 6-1 is a state flow chart which shows the sequence control of the ARS ROM address. The timing is a function of either the count in GR1 or the number of times shifted. The loop terminates when the count in GR1 reaches 0 or when 32 shifts have been performed. Thus, the execution timing may be expressed as:

$$t = 4n\ (.880)\ \text{micro-seconds}$$

where n = the absolute value of the count in GR1. ($n \leq 32$)



ARS STATE FLOW

Figure 6-1

Thus, a 32 bit shift requires 112.64 micro-seconds excluding the FO instruction which calls ARS. Including the time (1 cycle), the worse case timing becomes 114.4 micro-seconds.

6.3 Programming:

An arithmetic right shift is used in scaling operations on single, double precision, or floating point numbers. The number loaded into GR1 represents a scale factor which is a power of 2 by which the D.P. number in AX and AY is divided. Thus, with GR1 = -2, AX and AY will be divided by 2^2 or 4. The scaling operation is analogous to moving the decimal point to the left for each power of 10 in standard scientific notation, e.g. $2.5 \times 10^{-3} = 0.0025$.

The operator, of course, assumes that bit 15 of AX is the sign of the D.P. number in AX and AY.

CHAPTER SEVEN

THE NORMALIZE OPERATOR (NORM)

7.0 Normalize (NORM):

The NORM option is an EIR device that executes a string of micro-instructions which manipulate AX, AY, and GR1 to perform a normalize operation on AX and AY (considered a double precision register). Each shift required to normalize the DP value in AX and AY causes the contents of GR1 to be incremented.

The operator expects the following conditions prior to starting:

1. C (AX) = MSH operand
2. C (AY) = LSH operand
3. C (GR1) = V (any no.)

The operator is started by the command

FO 4,14 02 0100 14

The operator completes its execution leaving the following states:

1. C (AX) = MSH normalized operand
2. C (AY) = LSH normalized operand
3. C (GR1) = V + n where n=no. shifts required to normalize AX,AY
4. AO state = ADD

The AX and AY are considered a double precision word as shown in 4.0 of the MPO description. The word is filled with trailing 0's, which are entered through bit 0 of AY.

7.1 NORM Micro Program:

A normalize operation is used to make a binary number as large as it can be made and record the number of shifts required. Bit 15 of AX is the sign of the number, and the algorithm shifts the D.P. number in AX and AY until a 1 appears in AX14 for positive numbers or a 0 appears in AX14 for negative numbers. The algorithm used adds 11 to bits 15 and 14 of AX until a 0 appears as the 15th sum bit. The following table demonstrates detection of normalization.

AX 15,14	$\Sigma_{15,14}$ (+1 1)		
0 0	(1)	1	;not normalized
0 1	(0)	0	;normalized (positive number)
1 0	(0)	1	;normalized (negative number)
1 1	(1)	0	;not normalized

The micro-program for NORM is:

```
0 FOA OR
1 RR AO,0       ;check for AX and AY=0
2 RR AX,0       ;check for normalization  ┐
3 FOM CLL       ;not normalized           │
4 RS AY,L1      ;shift AX,AY left         │  ;Loop
5 RS AX,L1                                │
6 RS 30,P1      ;increment count          ┘
7 FOA ADD
```

7.2 NORM Address State Flow and Timing:

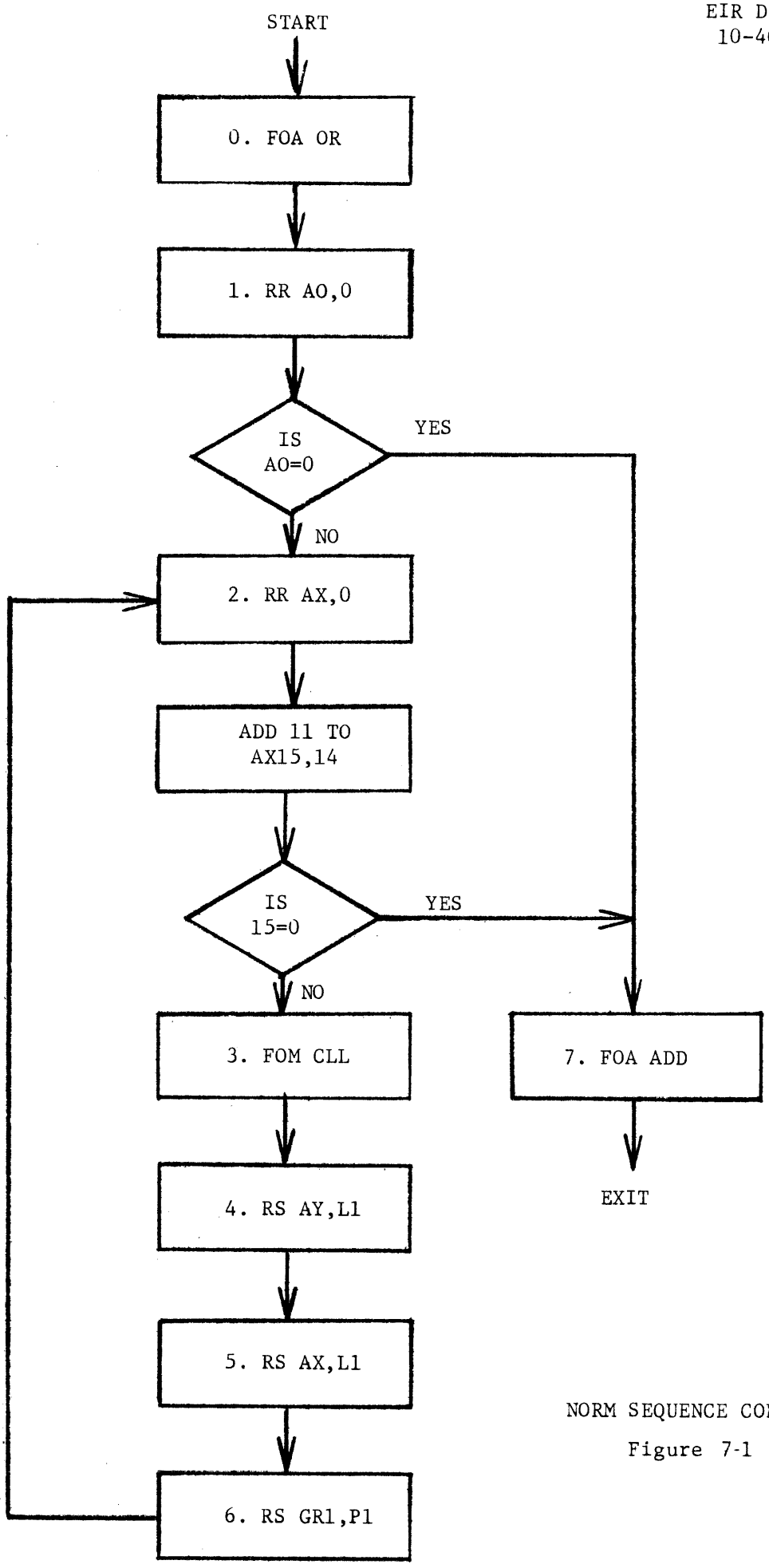
Figure 7-1 is a state flow chart which shows the sequence control of the NORM ROM address. The timing is a function of the value in AX and AY. For each bit of normalization that occurs, the loop timing is 5 micro-cycles. The lead in and exit code require 3 micro-cycles. The maximum time occurs for normalizing the D.P. number -1, which requires 32 shifts. The minimum time occurs when the D.P. number 0 is normalized. The following table is a timing summary for the NORM operator.

	Micro-cycles	u sec*
MIN time	4	3.52
AVG time	84	146.08**
MAX time	164	288.64

*The execution time of the FO instruction (1.76us) which invokes the NORM operator is not included. **In actual use with random numbers, the average time is 8 micro-cycles or 7.04 us.

7.3 Programming;

The normalize operation is the counterpart of the arithmetic right shift. It is, in effect, an unscaling operation. It is analogous to moving the decimal point to the right in order to put a number into standard scientific notation with a power of 10. e.g. $0.0025=0.25X10^{-2}$.



NORM SEQUENCE CONTROL

Figure 7-1

If GR1 is 0 at the start of a normalize, the number in GR1 after normalizing is the positive number of shifts required to normalize AX and AY. If GR1 is 2's complemented and an ARS is performed, the initial number is re-established in AX and AY.

GRI Computer Corporation

320 NEEDHAM STREET, NEWTON, MASSACHUSETTS 02164

TEL: (617) 969-0800