Pub. No. 4200010A

General Purpose
Controller/Processor
GPC/P

INTRODUCTION TO THE MICROPROGRAM FOR IMPLEMENTATION OF IMP-16 MACROINSTRUCTION SET

Application Note

June 1972

1.0 INTRODUCTION

This document provides an introduction to the microprogram that implements the IMP-16 instruction set. It is assumed that the reader is familiar with the material contained in Pub. No. 4200005, GPC/P Product Description. Additional reference material that would be useful to the reader who needs a more thorough understanding of microprogramming the GPC/P may be found in Pub. No. 4200001, GPC/P Microsymbolic Assembler and Pub. No. 4200007, GPC/P Microcoding Manual.

2.0 FLOWCHART DESCRIPTION

An overview of the microprogram for the IMP-16 instruction set is provided by the flowchart shown in figure 2-1. This figure delineates the major operations performed by the microprogram. A more detailed definition of these operations is presented in Section 3.0 of this publication.

2.1 Fetching a Macroinstruction

Consider first the manner in which a macroinstruction is fetched and executed. The instruction fetch routine consists of two microprogram steps. As a result of the first step, the main memory location whose address is contained in the Program Counter (PC) is read, the nine most-significant bits of the memory data are stored in the CROM Instruction Register (CIR), and the least-significant byte of the memory data (with the sign of this byte extended into the most-significant byte) is stored in the Memory Data Register (MDR). On the second step of the instruction fetch routine, the PC is incremented.

2.2 Instruction Address Formation and Execution

The macroinstruction step that follows the instruction fetch routine depends upon the class of macroinstruction that is to be executed. If the instruction class is a type that does not use the common memory address formation routine (as specified by bit 9 of the Address Control ROM output), then the next microinstruction executed is specified by the macroinstructions op code, and there is a branch to the "entry point" of the microprogram for that macroinstruction. The microinstruction(s) that perform the macroinstruction's function are then executed. Following this, there is jump back to the first step of the instruction fetch routine.

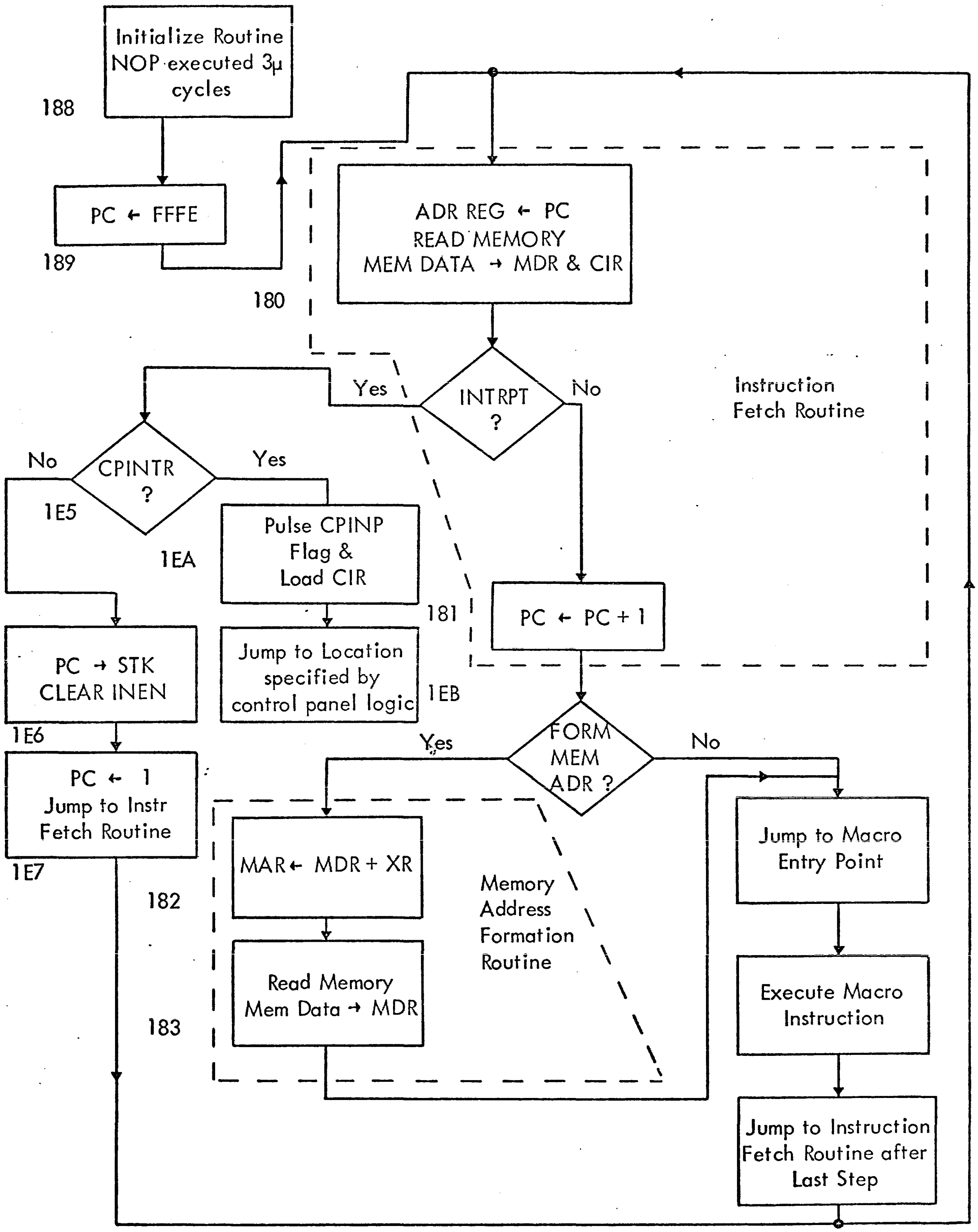
Initialize Routine NOP executed 3μ cycles
188
PC ← FFFE
189
180
ADR REG ← PC
READ MEMORY
MEM DATA → MDR & CIR
Yes
INTRPT ?
No
Instruction Fetch Routine
No
CPINTR ?
Yes
1E5
1EA
Pulse CPINP Flag & Load CIR
181
PC ← PC + 1
PC → STK
CLEAR INEN
Jump to Location specified by control panel logic
1EB
1E6
Yes
FORM MEM ADR ?
No
PC ← 1
Jump to Instr Fetch Routine
MAR ← MDR + XR
Memory Address Formation Routine
Jump to Macro Entry Point
1E7
182
Read Memory
Mem Data → MDR
183
Execute Macro Instruction
Jump to Instruction Fetch Routine after Last Step

Figure 2-1. Flowchart of Basic Microprogram

If there is no interrupt and the instruction class does use the common memory address formation routine (determined by bit 9 of the Address Control ROM output), then two additional microprogram steps are executed before branching to the "macro entry point." The first step consists of forming the memory address and storing it in the MAR. In the second step, memory is read and stored in the MDR. Then, there is the jump to the macro entry point.

2.3 Interrupt Processing

Consider now the response of the microprogram to an interrupt. On the first step of the instruction fetch routine, the condition of the INTRPT input to the jump condition multiplexer is tested. If the line is active, the microprogram jumps to the interrupt routine (INTR). On the first step of this routine, the CPINT input to the jump condition multiplexer is tested; when a control panel interrupt is requested, both the interrupt (INTRPT) and control panel interrupt (CPINT) inputs to the multiplexer are active. See figure 2-2 for the logic associated with the interrupt. Assume first that INTRPT is active as a result of INT REQ (an interrupt request from any I/O device controller), and that CP INTR is not active (see figure 2-2). Then, the next microprogram step executed turns off the interrupt enable flip-flop (INT EN becomes a "0," thus inhibiting further interrupts) and the program counter (PC) is saved on the stack. On the following step, the PC is set to a value of 1, and the microprogram branches to the instruction fetch routine at location 1. This causes the execution of the macroinstruction stored in location 1 of the main memory, and thus provides entry to the macro-level interrupt service routine.

Now, assume that INTRPT is active as a result of the CPINTR line being active. Then, following the first step of the interrupt routine, the microprogram branches to the control panel service routine. The first step of this routine causes the CPINP flag to be pulsed, and loads the CROM Instruction Register (CIR) and Memory Data Register (MDR) with the

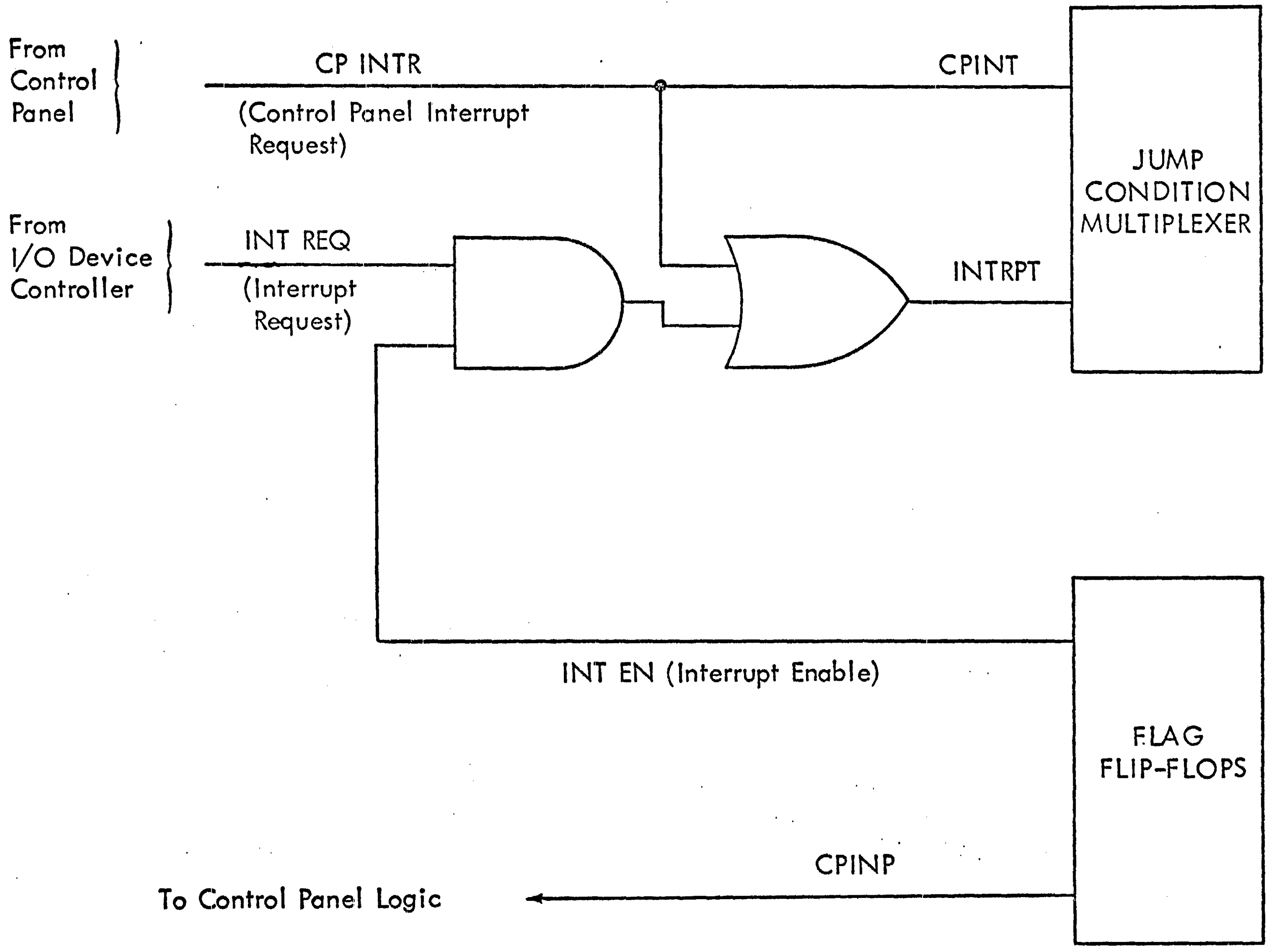
From Control Panel
CP INTR
(Control Panel Interrupt Request)
CPINT
JUMP CONDITION MULTIPLEXER
From I/O Device Controller
INT REQ
(Interrupt Request)
INTRPT
INT EN (Interrupt Enable)
FLAG FLIP-FLOPS
CPINP
To Control Panel Logic

Figure 2-2. Interrupt Logic Circuitry

3.0 SYMBOLIC DESCRIPTION OF MICROPROGRAM

A detailed description of the microprogram is provided by the listing of the symbolic code given in table 3-1. In order to fully understand the program, the reader should be familiar with the material presented in the manuals referred to in 1.0, Introduction.

A few of the major features of the listing are explained below.

- The first two pages of the listing define mnemonics used by the assembler (EQU statements).

- The first two fields of data on the left side of the page give the address and content in hexadecimal code. The 23-bit content is represented by 6 hexadecimal characters with the data right-hand justified: ROM bit 22 corresponds to the least-significant bit of the right-most hexadecimal character.

- A leading asterisk in the label field of the symbolic code denotes a comment statement.

- The prefix X' before a string of characters indicates that the characters are interpreted as a value in hexadecimal code.

- Arithmetic instructions have the following format (brackets indicate an optional field):

  [label]   operation, A-Bus, B-Bus, R-Bus   auxiliary operations   comments

- Branch instructions have the following format:

  [label]   operation   [, condition] address   auxiliary operations   comments

- Flag operations have a format similar to the format for arithmetic instructions except the operation field is replaced by a flag operation field followed by a flag designation.

- The ORG statement is a directive to the assembler and is used to control the ROM address where the microinstructions are located.

The first word of the instruction fetch routine is at location 180. (Note that all references to addresses in the following text are expressed in hexadecimal code.) The first word is not expressed in symbolic code, because the coding of this ROM word has been chosen to provide multiple functions for many of the ROM bits and thus make the instruction fetch routine efficient.

The ensuing operations performed are indicated in the comment statement that follows this step.

Note that the flag address of the Read Memory Flag (RDFF) is chosen to be the same as the jump condition multiplexer address for the interrupt line (INTRPT).

Coding of the ROM word is chosen so the microprogram jumps to location 1E5 if an interrupt is active.

Table 3-1. Symbolic Code Listing

```
000   000000   *****                                                              *****
000   000000   *****                                                              *****
000   000000   *****    DEFINE A SET OF MNEMONICS FOR USE BY PROGRAM              *****
000   000000   *****                                                              *****
000   000000   *****                                                              *****
000   000000   *
000   000000   *        DEFINE ARITHMETIC INSTRUCTIONS
000   000000   *
000   000100   XOR      EQU      X'100,1      EXCLUSIVE OR
000   000200   OR       EQU      X'200,1      INCLUSIVE OR
000   000000   AND      EQU      0,1          LOGICAL AND
000   000300   ADD      EQU      X'300,1      ARITHMETIC ADD
000   000000   *
000   000000   *        DEFINE TRANSFER INSTRUCTIONS
000   000000   *
000   200004   JMP      EQU      X'200004,2     JUMP (CONDITION OPTIONAL)
000   200004   B        EQU      JMP,2          BRANCH
000   600004   JSR      EQU      X'600004,2     JUMP TO SUBROUTINE
000   600004   BSR      EQU      JSR,2          BRANCH TO SUBROUTINE
000   600000   RET      EQU      X'600000,4     RETURN (UNCOND)
000   000008   JFETCH   EQU      8,4            JUMP TO FETCH (UNCOND)
000   400000   JINST    EQU      X'400000,4     JUMP TO INST (CIR TO RAR)
000   000000   *        DEFINE JUMP CONDITIONS
000   000000   INTRPT   EQU      0   INTERRUPT REQUEST TRUE
000   000001   REQ0     EQU      1   R BUS EQUAL TO ZERO (PREV INST)
000   000002   PSIGN    EQU      2   R BUS BIT IS FALSE (PREV INST)
000   000003   BIT0     EQU      3   R BUS BIT 0 TRUE (PREV INST)
000   000004   BIT1     EQU      4   R BUS BIT 1 TRUE (PREV INST)
000   000005   NREQ0    EQU      5   R BUS NOT ZERO (PREV INST)
000   000006   CPINT    EQU      6   CONTROL PANEL INTERRUPT REQUEST TRUE
000   000007   START    EQU      7   START BUTTON DEPRESSED
000   000008   STKFUL   EQU      8   STACK FULL
000   000009   INEN     EQU      9   INTERRUPT ENABLE FLAG TRUE
000   00000A   CYOV     EQU      10  CYOV1 OUTPUT FROM RALU TRUE
000   00000B   RLTEQ0   EQU      11  R BUS LESS THAN OR EQUAL TO ZERO (PREV INST)
000   00000E   FAILC    EQU      14  DIAGNOSTIC FAIL JUMP COND
000   000010   GJC      EQU      16  GATED JUMP COND (CIR BITS 0-3)
000   000000   *                     GJC ALSO CAUSES A ROM ENABLE
000   000000   *
000   000000   *        DEFINE FLAG INSTRUCTIONS
000   000000   *
000   000030   PFLG     EQU      X'30,3  PULSE FLAG
000   000010   RFLG     EQU      X'10,3  RESET FLAG
000   000020   SFLG     EQU      X'20,3  SET FLAG
000   000000   *        DEFINE FLAGS
000   000000   RDFF     EQU      0   READ MEMORY
000   000001   WRFF     EQU      1   WRITE MEMORY
000   000002   IDFF     EQU      2   INPUT DATA FROM PERIPHERAL
000   000003   ODFF     EQU      3   OUTPUT DATA TO PERIPHERAL
000   000004   CPINP    EQU      4   CONTROL PANEL COMMAND INPUT REQUEST
000   000005   SVRSTF   EQU      5   SAVE OR RESTORE STATUS FLAGS
000   000006   LDARFF   EQU      6   LOAD ADDRESS REGISTER
000   000007   HALTFF   EQU      7   SET BY HALT INSTRUCTION
000   000008   FAILFF   EQU      8   SET BY DIAGNOSTICS ON A FAILURE
000   000009   INENFF   EQU      9   ENABLE INTERRUPTS
000   00000A   SELFF    EQU      10  SELECT (USED BY STATUS FLAGS)
000   000010   GFLG     EQU      16  GATED FLAG ADDRESS (CIR BITS 0-3)
000   000000   *                     GFLG ALSO CAUSES A ROM ENABLE
000   000000   *
```

Table 3-1. Symbolic Code Listing (Continued)

```
000  000000  *        DEFINE MISCELLANEOUS INSTRUCTIONS
000  000000  *
000  000000  DATA     EQU      0,4   USED TO ENTER SPECIAL INSTRUCTIONS
000  000000  NOP      EQU      0,4   ALTERNATE MNEMONIC TO DATA
000  000000  *
000  000000  *        DEFINE REGISTER MNEMONICS
000  000000  *
000  000001  R1       EQU      1
000  000002  R2       EQU      2
000  000003  R3       EQU      3
000  000004  R4       EQU      4
000  000005  R5       EQU      5
000  000006  R6       EQU      6
000  000007  R7       EQU      7
000  000001  PC       EQU      R1      *
000  000002  MDR      EQU      R2      *
000  000003  MAR      EQU      R3      *
000  000004  AC0      EQU      R4      * ALTERNATES TO R1-R7
000  000005  AC1      EQU      R5      *
000  000006  AC2      EQU      R6      *
000  000007  AC3      EQU      R7      *
000  000008  GS       EQU      11      GATED SOURCE (FROM CIR3,2) - B OR R BUS ONLY
000  000009  GD       EQU      9       GATED DESTINATION (FROM CIR1,0) - A OR R BUS ONLY
000  00000A  GXR      EQU      10      GATED INDEX REGISTER (FROM CIR1,0) - A BUS ONLY
000  000008  STK      EQU      8       STACK, USE ON A OR R BUS ONLY
000  000000  *                            - REQUIRES ALT R FNS ON B BUS
000  000000  FLGS     EQU      0       RALU STATUS FLAGS (ONLY TO BE USED
000  000000  *                           IN CYCLE FOLLOWING 'PFLG,SVPSTF') - A AND R BUS
000  000000  *
000  000000  *        DEFINE OPTIONS FOR OPERAND FIELD (VALUE LIST B)
000  000000  *
000  300000  ROR      EQU      X'300000 ROTATE RIGHT (ALT A FNS ONLY)
000  300020  ROL      EQU      X'300020 ROTATE LEFT  (ALT A FNS ONLY)
000  300010  SHR      EQU      X'300010 SHIFT RIGHT  (ALT A FNS ONLY)
000  300030  SHL      EQU      X'300030 SHIFT LEFT   (ALT A FNS ONLY)
000  000040  CIN      EQU      X'40     SET LOW ORDER CARRY IN TRUE
000  000080  CMPA     EQU      X'80     COMPLEMENT THE A BUS
000  000002  BLB      EQU      2  BLANK LEFT BYTE OF B BUS (DON'T USE ADD)
000  000002  ENCOV    EQU      2  ENABLE CARRY AND OVERFLOW (ADD ONLY)
000  200002  RDI      EQU      X'200002 READ INSTRUCTION INTO CIR AND MDR
000  000004  DATAIN   EQU      4        LOAD THE R BUS FROM THE DATA BUS
000  000001  RE       EQU      1        ROM ENABLE
000  000008  IF       EQU      JFETCH   INSTRUCTION FETCH (ALT TO JFETCH)
000  000002  SPR      EQU      2        SIGN PROP(WITH DATAIN ONLY)
000  000000  *****                                                                        *****
000  000000  *****                                                                        *****
000  000000  *****    END OF MNEMONIC DEFINITIONS                                         *****
000  000000  *****                                                                        *****
000  000000  *****                                                                        *****
```

Table 3-1. Symbolic Code Listing (Continued)

```
000  000000  *
000  000000  *          INITIALIZE BRANCH TO IFETCH
189  000000             ORG   X'189
189  000000  *  NOTE THE 1ST STEP OF INITIALIZE ROUTINE IS EXECUTED 3 TIMES(LOC 188)
189  000000  *  SET PC TO FFFF
189  300688             OR,0,0,PC    CMPA,IF, SHL
18A  000000  * COMMON SKIP ROUTINE
18A  002748  SKIP       ADD,PC,,PC        CIN,IF
18B  000000  *
1E3  000000  * CONTROL PANEL INTERRUPT SERVICE ROUTINE
1EA  000000             ORG   X'1EA
1EA  200132  CPINTR     PFLG,CPINP,,,0    RDI
1EB  400200             OR,0,0,0    JINST
1EC  000000  *  CPINP FLAG LETS CONTROL PANEL LOGIC KNOW THAT CONTROL PANEL INTERRUPT IS
1EC  000000  *  BEING SERVICED. SERVICE ROUTINE MAY BE EITHER MACRO PROGRAM OR
1EC  000000  *  MICROPROGRAM STORED IN ANOTHER CROM.
1EC  000000  *
1EC  000000  *          INSTR FETCH ROUTINE
180  000000             ORG   X'180
180  2F2836  IFETCH     DATA  X'2F2836
181  000000  *  JMP TO INTR IF INTRPT,PULSE RDFF, R-BUS TO MDR, READI
181  402740             ADD,PC,,PC        CIN,JINST        AUTO JMP TO MACRO ENTRY PT IF ACR-9= 0
182  000000  *    NOTE RE IS NEEDED ON STEP FOLLOWING LCAR
182  122F01             ADD,GXR,MDR,MAR RE                 MEMORY ADDRESS FORMATION
183  000000  *    NOTE RE IS NEEDED, ALL MEM REF INSTRUCTIONS MUST BE IN MASTER CROM
183  406834             PFLG,RDFF, MAR,,MDR  DATAIN,JINST  READ MEMORY
184  000000  *   AUTOMATIC JUMP TO ENTRY POINT OF MACROINSTRUCTION HERE
1E5  000000             ORG   X'1E5
1E5  000000  *          INTERRUPT ROUTINE FOLLOWS
1E5  2F5184  INTR       B,CPINT    CPINTR     TEST FOR CONTROL PANEL INTERRUPT
1E6  092250             RFLG,INENFF,PC,0,STK  0        TURN OFF INTERRUPT ENABLE AND SAVE PC
1E7  000000  *  NOTE INENFF HAS ADDRESS WHICH CAUSES 'OR'
1E7  000748             ADD,0,0,PC  CIN,IF             EXECUTE INSTR IN LOC 1 OF MAIN MEMORY
1E8  000000  * LOAD
080  000000             ORG   X'80
080  0A4209             OR,MDR,,GS        IF,RE            MOVE MDR TO GATED REGISTER
081  000000  * LOAD INDIRECT (SINGLE LEVEL)
090  000000             ORG   X'90
090  0A403D             PFLG,RDFF,MDR,,GS    DATAIN,IF,RE   RD MEM, LOAD INTO SOURCE REG
091  000000  * STORE
0A0  000000             ORG   X'A0
0A0  000000  *          TWO FOLLOWING LOCS COULD BE SAVED IF COMMON MEM ADR FORM USED
0A0  122F01             ADD,GXR,MDR,MAR  RE   MEMORY ADDRESS
0A1  0061B0             PFLG,LDARFF,MAR,,0  0
0A2  0C0A00  STOR       OR,0,GS,MDR  0
0A3  004078             PFLG,WRFF,MDR,,0     IF         WRITE MEMORY WITH (GS),
0A4  000000  * STORE INDIRECT (SINGLE LEVEL)
0B0  000000             ORG   X'B0
0B0  004181             PFLG,LDARFF,MDR,,0  RE         LOAD AR WITH ADDRESS
0B1  251404             B   STOR
0B2  000000  * ADD
0C0  000000             ORG   X'C0
0C0  0E430B             ADD,MDR,GS,GS            IF,RE,ENCOV
0C1  000000  * SUB
0D0  000000             ORG   X'D0
0D0  0E43CB             ADD,MDR,GS,GS        CMPA,CIN,IF,RE,ENCOV    2'S COMPLEMENT OF MDR + SR
0D1  000000  *  SKG
0E0  000000             ORG  X'E0
0E0  0C4581             ADD,MDR,GS,MDR  RE,CMPA     SUB (MDR+1) FROM SR
0E1  2C508C             B,PSIGN  SKIP,IF
0E2  000000  *  SKNE
0F0  000000             ORG  X'F0
0F0  0C4901             XOR,MDR,GS,MDR  RE
0F1  2C514C             B,NREQU  SKIP,IF
0F2  000000  * ANDI
```

[illegible]le 3-1. Symbolic Code Listing (Continued)

```
                   X'60
           [illegible]DR,GS,GS        IF,RE

                   X'68
           [illegible]DR,ACO,ACO      -IF,RE

                   X'6C
           [illegible]DR,AC1,AC1  IF,RE

      * [illegible]
                   X'70
           AND,MDR,GS,MDR   RE
           [illegible]ERO       SKIP,IF
      * ISZ
                   X'78
           [illegible]DR,,MDR    CIN,RE          INCREMENT
           [illegible]WREF,MDR,MDR,MDR   0       RESTORE IN MEMORY
[illegible]     *  NOTE [illegible] MUST HAVE MDR CONTENTS FOR PROPER TEST, 2 MS BITS OF WREFF
[illegible]     *  MUST [illegible]ROS (AND FUNCTION).
[illegible]         [illegible]ERO      SKIP,IF            SKIP IF ZERO, ELSE IFETCH
[illegible]000  *  DSZ
[illegible]000             X'7C
[illegible]01      [illegible]D,MDR,MDR    CMPA,RE    SUB ONE
[illegible]04         X'79    GO TO ISZ+1
[illegible]000  * PUSH
[illegible]000             X'40
[illegible]209     [illegible]GD,,STK       RE,IF
[illegible]000  *
[illegible]000  *     NOTE: [illegible]OCATION 41 USED BY REG-REG INSTRUCTIONS
[illegible]000  *     ([illegible] CHANGE THIS SINCE R-BUS ADDRESS CAN BE ARBITRARY)
[illegible]000  * PULL
[illegible]000             X'44
[illegible]209     [illegible]STK,,GD       RE,IF
[illegible]000  * AIS[illegible]   [illegible]D N, SKIP IF ZERO
[illegible]000             X'48
[illegible]303     [illegible]GD,MDR,GD    RE,ENCOV
[illegible]504C    [illegible]ERO      SKIP,IF
[illegible]000  * LI
[illegible]000             X'4C
[illegible]209     [illegible],MDR,GD     RE,IF
[illegible]000  * CAI      [illegible]D N TO 1'S COMPLEMENT OF D (2'S COMPLEMENT FOR N=1)
[illegible]000             X'50
[illegible]289     [illegible]GD,MDR,GD    RE,CMPA,IF
[illegible]000  *
[illegible]000  *      NOTE [illegible] ROM OF CROM, ARN(8) IS DECODED AS 'DON'T CARE' FOR CAI
[illegible]000  *
[illegible]000           X'150
[illegible]389     [illegible]GD,MDR,GD    RE,CMPA,IF
000000  *
000000  *
000000  * SHL      [illegible]IFT LEFT OPEN N PLACES)
000000             X'56
30C230     [illegible]D,,GD        SHL
000000  *  EN[illegible] [illegible]T FOLLOWS
020081     [illegible]D,MDR,MDR    CMPA,RE   SUBTRACT  1 FROM N  (ENTRY POINT)
22088C     [illegible]SIGN     $-2,IF        IF RESULT NON-NEGATIVE GO TO SHIFT, ELSE IF
000000  * ROL      [illegible]TE LEFT (SHIFT CIRCULAR)
000000             X'57
30C220     [illegible]D,,GD        ROL
000000  * ENT[illegible] [illegible]T FOLLOWS
020081     [illegible]D,MDR,MDR    CMPA,RE
22088C     [illegible]SIGN     $-2,IF
000000  * SHR      [illegible]FT RIGHT
000000             X'153
30C210     [illegible]D,,GD        SHR
000000  *      NOTE [illegible] IS INITIALLY A NEGATIVE NO.
000000  * EN[illegible]Y POIN[illegible] FOLLOWS
```

Table 3-1. Symbolic Code Listing (Continued)

```
060  000000          ORG   X'60
060  0E4009          AND,MDR,GS,GS        IF,RE
061  000000  * OR0
068  000000          ORG   X'68
068  045209          OR,MDR,AC0,AC0      -IF,RE
069  000000  * OR1
06C  000000          ORG  X'6C
06C  055609          OR,MDR,AC1,AC1  IF,RE
06D  000000  * SKAZI
070  000000          ORG  X'70
070  0C4601             AND,MDR,GS,MDR   RE
071  2C504C          B,REG0        SKIP,IF
072  000000  * ISZ
078  000000          ORG   X'78
078  004641          ADD,MDR,,MDR    CIN,RE          INCREMENT
079  024870          FFLG,WRFF,MDR,MDR,MDR   0      RESTORE IN MEMORY
07A  000000  *  NOTE F-BUS MUST HAVE MDR CONTENTS FOR PROPER TEST, 2 MS BITS OF WRFF
07A  000000  *  MUST BE ZEROS (AND FUNCTION).
07A  2C504C          B,REG0      SKIP,IF                SKIP IF ZERO, ELSE IFETCH
07B  000000  *  DSZ
07C  000000          ORG  X'7C
07C  020861          ADD,0,MDR,MDR   CMPA,RE    SUB ONE
07D  23CC04          B   X'79   GO TO ISZ+1
07E  000000  * PUSH
040  000000          ORG   X'40
040  194209          OR,GD,,STK        RE,IF
041  000000  *
041  000000  *      NOTE LOCATION 41 USED BY REG-REG INSTRUCTIONS
041  000000  *     (COULD CHANGE THIS SINCE R-BUS ADDRESS CAN BE ARBITRARY)
041  000000  * PULL
044  000000          ORG   X'44
044  198209          OR,STK,,GD        RE,IF
045  000000  * AISZ     ADD N, SKIP IF ZERO
048  000000          ORG   X'48
048  12C303          ADD,GD,MDR,GD   RE,ENCOV
049  2C504C          B,REG0      SKIP,IF
04A  000000  * LI
04C  000000          ORG   X'4C
04C  128209          OR,0,MDR,GD       RE,IF
04D  000000  * CAI      ADD N TO 1'S COMPLEMENT OF D (2'S COMPLEMENT FOR N=1)
050  000000          ORG   X'50
050  12C389          ADD,GD,MDR,GD   RE,CMPA,IF
051  000000  *
051  000000  *      NOTE IN ROM OF CROM, ARM(8) IS DECODED AS 'DON'T CARE' FOR CAI
051  000000  *
150  000000          ORG X'150
150  12C389          ADD,GD,MDR,GD   RE,CMPA,IF
151  000000  *
151  000000  *
151  000000  * SHL     (SHIFT LEFT OPEN N PLACES)
05B  000000          ORG   X'5B
05B  30C230          OR,GD,,GD          SHL
05C  000000  *  ENTRY POINT FOLLOWS
05C  020B81          ADD,0,MDR,MDR   CMPA,RE   SUPTRACT  1 FROM N  (ENTRY POINT)
05D  22D88C          B,PSIGN     $-2,IF          IF RESULT NON-NEGATIVE GO TO SHIFT, ELSE IF
05E  000000  * ROL     ROTATE LEFT (SHIFT CIRCULAR)
057  000000          ORG   X'57
057  30C220          OR,GD,,GD          ROL
058  000000  * ENTRY POINT FOLLOWS
058  020BA1          ADD,0,MDR,MDR   CMPA,RE
059  22B68C          B,PSIGN    $-2,IF
05A  000000  * SHR     SHIFT RIGHT
15B  000000          ORG  X'15B
15B  30C210          OR,GD,,GD          SHR
15C  000000  *     NOTE N IS INITIALLY A NEGATIVE NO.
15C  000000  * ENTRY POINT FOLLOWS
```

Table 3-1. Symbolic Code Listing (Continued)

```
15C  020541            ADD,0,MDR,MDR   CIN,RE
15D  2ADACC            B,PLTEGD S-2,IF
15E  000000  * ROR     ROTATE RIGHT
157  000000            ORG   X'157
157  30C200            OR,GD,,GD           ROR
158  000000  * ENTRY POINT FOLLOWS
158  020541            ADD,0,MDR,MDR   CIN,RE
159  2ADACC            B,PLTEGD S-2,IF
15A  000000  *  XCHRS EXCHANGE REGISTER AND STACK
054  000000            ORG   X'54
054  C90E01            CR,STK,,MAR    RE   (PULL STACK TO WIPE OUT OLD DATA)
055  194200            OR,GD,,STK     0    (PUSH NEW DATA ONTO STACK)
056  138208            OR,0,MAR,GL    IF
057  000000  *
057  000000  * REG-REG INSTRUCTIONS
057  000000  * RADL
030  000000            ORG    X'030
030  1CC308            ADD,GD,GS,GD          IF,RE,ENCOV
031  000000  *  RAND MICROPROGRAM BRANCHES TO THIS POINT AFTER AUGMENT DECODE
031  1CC008  RANDL     AND,GD,GS,GD    IF
032  000000  *  RXOR MICROPROGRAM BRANCHES TO THIS POINT AFTER AUGMENT DECODE
032  1CC108  RXORL     XOR,GD,GS,GD    IF
033  000000  *
033  000000  *  ENTRY POINT FOR RXCH,RCPY,RXOR,RAND
033  000000  *  DECODE AUGMENT FIELD (REG-REG INSTR'S)
130  000000            ORG    X'130
130  024A01            OR,MDR,MDR,MDR    RE  TEST BIT 0
131  2208C4            B,BIT0  X'41            GO TO TEST RCPY OR RAND IF TRUE
132  000000  *      NOTE THE ABOVE INSTRUCTION HAS THE EFFECT OF GATING MDR ONTO THE
132  000000  *      R BUS TO PERMIT TEST FOR BIT 1.
132  000000  *     (SAME AS AND,0,MDR,MDR    CMPA)
132  219104            B,BIT1  RXORL
133  000000  *  RXCH
133  104A01            OR,GD,0,MDR    RE  TEMP STORE
134  1C8200            OR,0,GS,GD     0   LOAD GD
135  0A4208            OR,MDR,0,GS    IF  LOAD GS
136  000000  * .TEST FOR RCPY OR RAND
041  000000            ORG X'41
041  218904            B,BIT1  RANDL
042  000000  *  RCPY
042  1C8208            OR,0,GS,GD      IF
043  000000  *   JMP
020  000000            ORG    X'20
020  122709            ADD,GXR,MDR,PC  IF,RE  FORM ADDRESS STORE IN PC
021  000000  *   JMP INDIRECT
024  000000            ORG  X'24
024  004609            OR,MDR,,PC  IF,RE  MOVE MDR TO PC
025  000000  *   JSR (JUMP TO SUBROUTINE)
028  000000            ORG   X'28
028  092201            OR,PC,,STK        RE                PUSH PC ONTO STACK
029  122708            ADD,GXR,MDR,PC  IF    FORM ADDRESS,STORE IN PC
02A  000000  *   JSR INDIRECT
02C  000000            ORG  X'2C
02C  092201            OR,PC,,STK  RE  PUSH PC ONTO STACK
02D  004608            OR,MDR,,PC   IF
02E  000000  * SET FLAG
008  000000            ORG   X'008
008  004181            PFLG,LDAREF,MDR,,0   RE         LOAD AR WITH DEVICE ADDR
009  200029            SFLG,GFLG            IF         SET GATED FLAG
```

Table 3-1. Symbolic Code Listing (Continued)

```
00A  000000  * PULSE FLAG

108  000000             ORG   X'108
108  0041C1             PFLG,LDARFF,MDR,,0   RE
109  200639             PFLG,GFLG            IF
10A  000000  *  BOC
010  000000             ORG   X'10
010  009201             OR,AC0,,AC0      RE
011  209000             B,GJC       $+1,IF                 GO TO IFETCH IF CONDITION NOT SATISFIED
012  014708             ADD,MDR,PC,PC    IF                OTHERWISE ADD DISPLACEMENT TO PC
013  000000  *   HALT
1FE  000000             ORG X'1FE
1FE  2FF10C  CKREL      B,START     $,IF,PFLG  CLEARS HALT FF
1FF  000000  *  NOTE THAT ROM ADDRESS ''WRAPS AROUND'' FROM 1FF
1FF  2FF1E4  CKST       B,START     CKREL,SFLG  SETS HALTFF
000  000000           . ORG X'00
000  000000  * NOTE RE CAUSES GATED JC (OK SINCE HALT HAS OP CODE OF ALL 0'S)
000  2FFC05             B    CKST,RE
001  000000  *  RTI  (RETURN FROM INTERRUPT)
001  000000             ORG  X'01
001  000261             SFLG,INENFF  RE
002  000000  *RTS (RETURN FROM SR)
002  000000             ORG   X'02
002  090601  RTSUB      OR,STK,,PC         RE            POP STACK
003  022708             ADD,PC,MDR,PC    IF          ADD DISPL
004  000000  * PUSHF    PUSH FLAGS ONTO STACK
100  000000             ORG  X'100
100  000171             PFLG,SVRSTF       RE      PULSE SVRSTF
101  090208             OR,0,0,STK     IF      PUSH FLAGS ONTO STACK
102  000000  * PULLF     PULL STACK STORE IN FLAGS
102  000000             ORG  X'102
102  090971             PFLG,SVRSTF,STK,0,MDR  RE  PULL STACK INTO MDR(SVRST ADDRESS MUST BE 5)
103  004208             OR,MDR,0,0    IF  PUT MDR INTO FLAGS
104  000000  *
104  000000  * RIN
004  000000             ORG   X'04
004  6CC405             BSR   X'198,RE       LOADS ADDR REG
005  0010BC             PFLG,IDFF,0,0,AC0    DATAIN,IF   DATA IN TO AC0
006  000000  * ROUT
006  000000             ORG   X'06
006  3CC405             BSR   X'198,RE       LOADS ADDR REG
007  0060F8             PFLG,ODFF,AC0,0,0     IF          DATA OUT
008  000000  * LOAD ADDR REG SUBROUTINE
198  000000             ORG   X'198
198  000000  * BLANK MOST SIGNIFICANT BYTE
198  024A02             OR,MDR,MDR,MDR   BLB
199  074C00             ADD,MDR,AC3,MDR    0   ADD DISPL TO AC3
19A  604160             PFLG,LDARFF,MDR,0,0  RET   LOAD AR
19B  000000             END
```

4.0 ADDRESS CONTROL ROM PROGRAMMING

The programming of the Address Control ROM (ACR) is shown in table 4-1. The address programming establishes the classes of instructions that result in the various masks generated by the ACR output. From 1 to 12 different masks are possible. An X denotes a "don't care" term.

The reader is referred to the GPC/P Microcoding Manual for a more complete discussion of the function of the Address Control ROM.

Table 4-1. Address Control ROM Programming

ADDRESS 876543210	DATA 0123456789	
X1XXXXXXX	0000111101	ARITH-REG XFER 12-15,FORM MEMADR
X1010XXXX	0000111100	STORE DOES NOT USE COMMON MEM ADR FORMATION
X011X1XXX	0011111101	ORI,ISZ,DSZ XFER 10-15,FORM MEMADR
X011X0XXX	0000111101	ANDI,SKAZI XFER 12-15
X0100XXXX	0011111100	SINGLE-REG XFER 10-15
X0101XXXX	0011111110	SHIFTS,CAI,AND STACK COPY XFER 10-15,7
X0011XXXX	0000111110	REG-REG XFER 12-15,7
X0010X0XX	0011111100	JMP,JSR XFER 10-15,(NO COMMON MEM ADR FORM)
X0010X1XX	0011111101	JMP*JSR* XFER 10-15,FORM MEM ADR
X00001XXX	0001111110	FLAG XFER 11-15,7
X0001XXXX	0000111100	COND JMP XFER 12-15
X00000XXX	1111111110	I/O,MISC XFER 8-15,7