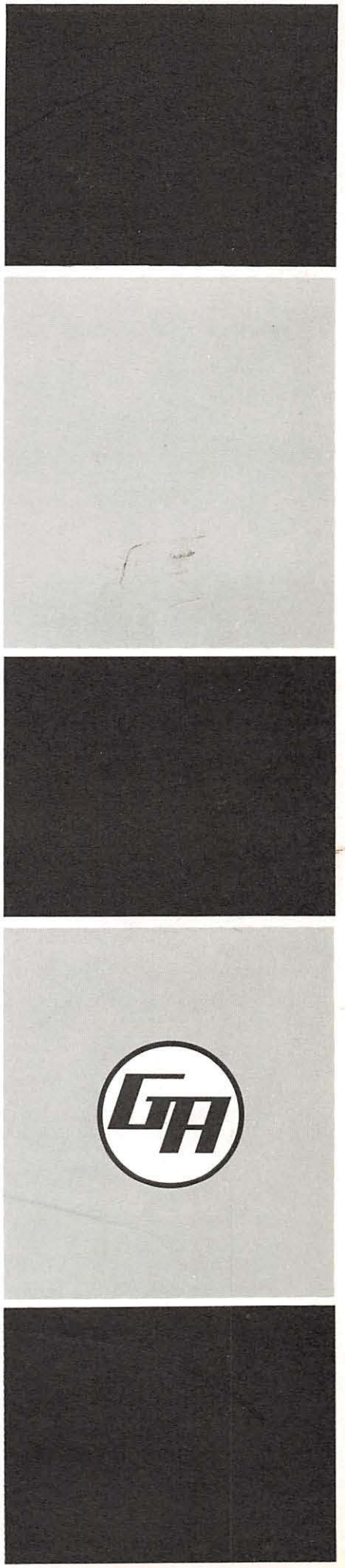
TECHNICAL MANUAL

SPC-12
PROGRAM
DESCRIPTION



GENERAL AUTOMATION, INC.

GENERAL AUTOMATION, INC.

PRICE $10.00

88A00106A-B

SPC-12 PROGRAM DESCRIPTION

TECHNICAL MANUAL

GENERAL AUTOMATION, INC.
Automation Products Division
1055 East Street, Anaheim, California 92805 (714) 778-4800

REVISION

Issue	Description	Approval	Date
A	Original Issue	George Rugg	Feb 70
B	Production Release	George Rugg	Apr 70

CONTENTS

CONTENTS (continued)

SECTION 1
INTRODUCTION

1.1 GENERAL

The SPC-12 Program Description Manual is to be used in conjunction with the SPC-12 Reference and the SPC-12 Programming Manual. The SPC-12 Program Description Manual consists of four sections.

Section one very briefly describes the contents of the manual. Section two contains the various coding techniques in the form of example problems and solutions. Section three describes the group of subroutines which make up the SPC-12 arithmetic library. Also included in this section are: the use of the library, memory requirements and program applications. Section four contains the teletype listings of the subroutines in the arithmetic package. The appendices include listings for the following:

a. PGS bootstrap loader

b. Serial teletype I/O

c. PGS paper tape loader/puncher

d. PGS basic utility system II (BUS II)

e. GAI recommended shared command bytes

f. PGS CAS II source tape preparation routine (TAPREP)

SECTION 2
CODING TECHNIQUES

2.1 GENERAL

This Section is a collection of coding techniques that have been used, at one time or another, in the generation of programs on the SPC-12.

Most examples aid in the understanding of instruction execution and should aid in acquainting an individual in coding programs on the SPC-12.

This Section may also be used as a scrapbook by the user to accumulate techniques that are worth saving and referencing at a later date.

2.2 CONVENTIONAL TECHNIQUES

The following paragraphs contain example problems and solutions using conventional coding techniques. The following example problems are included:

a. Zero B register.

b. Eight-bit complement A register.

c. Complementing a 12-bit or eight-bit value (2's complement).

d. Testing for a 12-bit zero.

e. 16-bit comparison.

f. Skipping in the forward or backward direction relative to the P counter.

g. Loading the B register and any other register with the same value.

h. Subroutine linkage convention.

i. Flags.

j. Moving 100_8 bytes from area 1 to area 2 *

k. Comparing a character to a number of characters arising in line code.

2.2.1 ZERO B REGISTER

Solution:

```
TBB     Shift upper 4 bits to B
TBB     Shift zero to B
```

2.2.2 EIGHT-BIT COMPLEMENT A REGISTER

Solution:

```
*       8 bit complement
        RDC   B,B
        AXR   A.B
```

2.2.3 TWO'S COMPLEMENT OF 12-BIT OR EIGHT-BIT VALUE

Solution:

Technique	Value in A Register
RTR A,B	Put A in B
AZE A	Zero A
ASU A,B	A - B to A

2.2.4 TEST FOR 12-BIT ZERO

Solution:

Following technique used for Y or Z Registers

```
RDC Y,Y      Y - 1 to Y        Load Zero F/F
SKZ 2        Is (Y) = 0?
JMP —        No
RTR YY,YY    YES (YY) = 0      Load Zero F/F
SKN 2
JMP Zero Exit  Yes  Exit
ETC.
```

Alternate technique for A or X Registers

```
RIC   A,A
SKZ   2              Is (A) = 0?
JMP   —              No
RTR   A,B            Yes
TBB                  Shift B(8-11) to B(0-3)
SKN   2              Is B = 0?
JMP   Zero Exit      Yes
ETC.
```

2.2.5 16-BIT CONVERSION

Solution:

```
* A,Y        16 bit Minuend
* X,Z        16 bit Subtrahend
  RTR Z,B    Low Order 8 bits to B
  ASU Y,B    Subtract low order 8 bits
  RLK A,A    Carry to high order
  RTR X,B    Hi order to B
  ASU A,B    Subtract high order
  RDC A,A    Adjust to 1's complement arith-
               metic for high order only.
  SKP 2
  JMP Minus
  SKZ 2
  JMP Plus
  JMP Zero
```

2.2.6 FORWARD OR BACKWARD SKIP

To skip in the forward or backward direction relative to the P counter, proceed as shown below:

Solution:

To skip in the forward direction, the instruction

```
AAD   P,n-1
```

applies where n is the number of cells required to skip to the destination address, relative to the location of the skip constant.

To skip in the backward direction, the instruction

```
ASU   P,n+1
```

2.2.7 LOAD B REGISTER AND OTHER REGISTERS WITH SAME VALUE

To load the B register and any other register with the same value.

Solution:

The following instruction will load the A register with the constant 123.

```
ALD   A,123
```

The value is also put into the B register.

2.2.8 SUBROUTINE LINKAGE CONVENTION

Subroutine linkage convention.

Solution:

```
Ω     RIC P,B*   Ω+1 to B register
Ω+1   JMP SU     Go to subroutine
Ω+2
Ω+3   Return here
```

```
SU   STB  TE      Save return link
     TBB
     STB  TE+1
     .
     .
     .
     .
     .
     .
     ELB  TE      Restore return link
     RIC  B,P*    Return to calling program at Ω+3

                  (B) = Ω+1
                  RIC B,P = Ω+2 to P register
                  CPU executes P+1 to P at the end
                    of instruction execution cycle.
```

*Any register not used in subroutine can be substituted for B.

2.2.9 FLAGS

Solution:

To turn a flag on (off), the following instruction is useful:

```
RDC  B,B
```

This sets B to all ones (negative).
A test for negative would indicate an on (off) condition.

To turn a flag off (on) the following instruction is useful:

```
TBB
TBB
```

A test for zero would indicate an off (on) condition. Since the flag is in the B register, it can easily be stored with

```
STB  FLAG
```

or when testing, the flag is easily loaded with

```
LDB  FLAG
```

2.2.10 MOVE 100_8 BYTES FROM AREA ONE TO AREA TWO

Solution 1:

```
M1   AZE  X       Initialize Index
M2   LDB  A1,X    Get from A1
     STB  A2,X    Put into A2
     RTR  X,A
     ASU  A,077
     SKZ  4       All Done?
     RIC  X,X     No - Increment Index
     JMP  M2
     JMP  EX      Exit
A1   ZR   100     Area 1
A2   ZR   100     Area 2
```

Solution 2:

```
M1   ALD  X,077   Initialize Index
M2   LDB  A1,X    Get from A1
     STB  A2,X    Put into A2
     RDC  X,X     Decrement Index
     SKM  2       Is Index Negative?
     JMP  M2      No
     JMP  EX      Exit
A1   ZR   100     Area 1
A2   ZR   100     Area 2
```

Solution 3:

```
M1   ELB  MA      Initialize Pointer Index
     RTR  B,X
     AZE  Y
M2   GOL  A,X,I   Get from A1
     G2S  A,X,I   Put into A2
     RTR  Y,A
     ASU  A,077
     SKZ  4       All Done?
```

```
        RIC  Y,Y      No - Increment Index
        JMP  M2
        JMP  EX       Exit
MA      ZA   MB       Pointer Address
MB      ZA   A1,Y     Pointer to Area 1
        ZA   A2,Y     Pointer to Area 2
A1      ZR   100      Area 1
A2      ZR   100      Area 2
```

2.2.11 COMPARING CHARACTERS

To compare a character to a number of characters arising in line code, proceed as follows:

Solution:

Compare the character G against the character C, E, A, G, J.

Arrange the characters in ascending order with respect to their numerical value and compute the differential.

A = 301
→ 2
C = 303
→ 2
E = 305
→ 2
G = 307
→ 3
J = 312

The comparison can now be made without reloading the G character each time:

```
ALD   A,307     Load G into A register
ASU   A,301     Subtract A value
SKN   2
JMP   A         Equal to A
ASU   A,2       Subtract A-C
SKN   2
JMP   C         Equal to C
ASU   A,2       Subtract C-E
SKN   2
JMP   E         Equal to E
ASU   A,2       Subtract E-G
SKN   2
JMP   G         Equal to G
ASU   A,3       Subtract G-J
SKN   2
JMP   J         Equal to J
```

2.3 ADVANCED TECHNIQUES

The following paragraphs contain example problems and solutions using advanced coding techniques. The following example problems are included:

2.3.1 CHANGE STATUS OF MEMORY WORD BITS

Change the status of certain bits within a word of memory by using the following:

Given:

Ones in a mask word indicate which bits are to be changed.

Solution:

Mask out the bits to be changed, complement them and restore to the word. The operation is defined in terms of the following logic equation.

1. $S = (S \cdot \bar{A}) \oplus ((S \cdot A) \cdot A)$

2. $S = \bar{S} \cdot A \oplus S \cdot \bar{A}$ (Reduced form of 1.)

Where: S = word in memory to be changed
A = mask indicating bits to be operated upon

```
LDB   S
RTR   B,A
```

```
LDB   A
AND   A,B
AXR   A,377   S̄ · Ā
LDB   A
AND   A,B     (S̄ · Ā) · A
LDB   A
RTR   B,Y
AXR   Y,377   Ā
LDB   S
AND   Y,B     S · Ā
RTR   Y,B
AOR   A,B     (S · Ā) ⊕ (S̄ · Ā) · A
RTR   A,B
STB   S
```

Equation 1. Coding

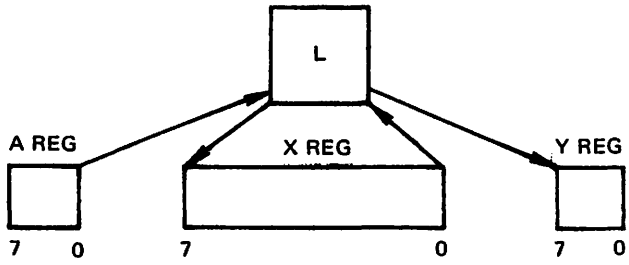
```
LDB   S
RTR   B,A
RTR   A,X
AXR   A,377   S̄
LDB   A
RTR   B,Y
AND   A,B     S̄ · A
AXR   Y,377   Ā
RTR   X,B
AND   Y,B     Ā · S
RTR   Y,B
AOR   A,B
RTR   A,B
STB   S
```

Equation 2. Coding

2.3.2 SHIFT RIGHT n BITS

Solution:

Information in A, X and Y are shifted right n bits. The link is used to transfer bit 0 data of one register to bit 7 of the next.

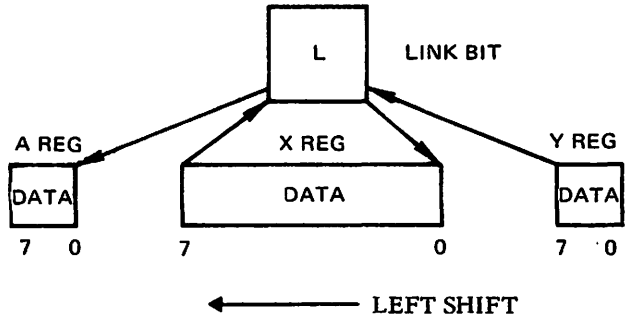


RIGHT SHIFT ⟶

```
      ALD Z,n   Load Z with shift count
LB    PLR       Reset Link
      SHL A     Shift A register right with bit 0 into
                  link
      SHL X     Shift link into X and bit 0 into link
      SHL Y     Shift link into Y
      RDC Z,Z   Decrement count
      SKZ 2     If Zero, exit
      JMP LB    Loop
```

2.3.3 SHIFT LEFT n BITS

Solution:



⟵ LEFT SHIFT

Information in A, X, and Y registers is shifted left n bits. The link bit is used to hold the carry out of bit position 7.

```
      ALD Z,n   Load Z register with shift count, Ω
LB    RTR A,B
      AAD A,B   Shift A 1 place
      RTR X,B
```

```
AAD X,B   Shift X 1 place
RLK A,A   Add Link to A
RTR Y,B
AAD Y,B   Shift Y 1 place
RLK X,X   Add Link to X
RDC Z,Z   Decrement Count
SKZ 2     If Zero, exit
JMP LB    Loop
```

Note

This is an open shift. The bit shifted from bit seven of the A register is lost and a zero is shifted into bit zero of the Y register.

2.3.4 UPDATE SYSTEM TIMER

Update system-timer consisting of time in units of milliseconds, seconds, minutes, and hours.

Given:

1. An interrupt; triggered every Δt milliseconds.
2. Four unit-of-time counters
   a. Milliseconds (max value = 250)
   b. Seconds (max value = 60.00)
   c. Minutes (max value = 60)
   d. Hours (max value = 24)

Solution:

When an interrupt occurs:

1. Increment the millisecond unit-of-time counter by ΔT and save.
2. Determine if the time increment has exceeded the range of the counter, that is, if an overflow has occurred. This is done by adding an overflow constant. (In the case of the millisecond counter, since $250_{10} = 372_8$, the overflow constant is $377_8 - 372_8$ or 5_8.)
   a. Increment the result of step 1 by the overflow constant.
   b. Replace the counter value with the residue obtained from step 2a if overflow occurs.
3. If any counter overflows, the residue of the overflow is saved in that counter and the next unit-of-time counter is incremented. Then this unit-of-time counter is tested for overflow in the same manner.

```
BE  LDB   PT        LOCATION OF COUNTER TABLE
    RTR   B,Y
    ALD   X,4       QTY OF "UNIT-OF-TIME" COUNTERS
LP  GØL   A,Y
    LDB   DT,X      INCREMENT OF UNIT-TIME
    AAD   A,B
    GØS   A,Y       UPDATED "UNIT-OF-TIME" COUNTER
    G4A   A,Y       OVERFLOW CONSTANT
    SKZ   2
    JMP   @         COUNTER DID NOT REACH MAX VAL - OUT
    GØS   A,Y       OVERFLOW - SAVE RESIDUE
    RDC   X,X
    SKN   2
    JMP   @         ALL COUNTERS UPDATED - OUT
    RIC   Y,Y       LOCATION OF NEXT COUNTER
    JMP   LP        TRANSFER BACK INTO LOOP



PT  ZC    Ø         MILLISECOND TIME COUNTER- MAX OF 25Ø
    ZC    Ø         SECONDS     TIME COUNTER- MAX OF 6Ø.ØØ
    ZC    Ø         MINUTES     TIME COUNTER- MAX OF 6Ø
    ZC    Ø         HOURS       TIME COUNTER- MAX OF 24
    ZC    6         MILLISECOND OVERFLOW CONSTANT
    ZC    2Ø        SECONDS     OVERFLOW CONSTANT
    ZC    3Ø4       MINUTES     OVERFLOW CONSTANT
    ZC    35Ø       HOURS       OVERFLOW CONSTANT
DT  ZC    Ø         FILLER
    ZC    Ø         HOURS       INCREMENT OF UNIT-TIME
    ZC    Ø         MINUTES     INCREMENT OF UNIT-TIME
    ZC    Ø         SECONDS     INCREMENT OF UNIT-TIME
    ZC    Ø         MILLISECOND INCREMENT OF UNIT-TIME
```

2.3.5 MODE LINKAGE VIA MAIN MEMORY

Solution:

Normal subroutine linkage may be effected from mode #N (N ≠ 0) to main memory.

```
MODE #N EXIT RIC  P,B
            JMP  Q M  Link in Main Memory

RETURN IN MODE #N  _____
                   _____
                   _____

MAIN MEMORY LINKAGE

      QM    STB  LM     Save Return
            TBB           In Link
            STB  LM+1     Table
            LDB  FN     Pick up Current Run-
                        ning Function
                        Number
            RTR  B,X
            LDB  M,X    Pick up FN's Normal
            STB  LM+2   Mode and Save In
            LDB  LM+3     Link Table
            TBE         Shift to New
            RIC  P,B    Memory Mode and
            JMP  GQ     Xfer to Desired Routine


      RETURN
      FROM GQ LDB  FN
              RTR  B,X
              LDB  LM+2   Restore Mode In
              STB  M,X      FN Mode Table
              TBE         Shift Mode to
              ELB  LM     Calling Prog Mode NR.
              RIC  B,P    Return to Calling Prog.

LINKAGE
TABLE LM      ZA          0
      LM+2    ZC          Mode of Running FN
      LM+3    ZC          Mode of Subroutine
```

2.3.6 TABLE SEARCH FOR COMPARISON

Table search for comparison (using augmented memory address instructions).

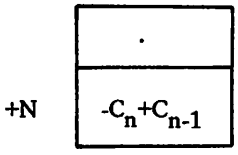
Solution:

Assumption: Character (unknown) is in A register. Bank zero pointer refers to the origin of the table. Contents of X are zero.

```
Ω    GOA     A,I   Pointer O  ZA  TB-1,X,I
     SKN     2
     JMP     Ω1
     ASU     X,N   Is it end of table
     SKP     6
     AAD     X,N   No  Restore X Register
     JMP     Ω     Loop
     SKF     O     Filler (No operation)
     ERROR         Error
Ω1   FOUND         Comparison
                   Successful
```

TABLE

TB	-CO
	$-C_1+C_0$
	$-C_2+C_1$
	.
	.
	.
	.
	.



Where C_i is the character for comparison

2.3.7 PARITY CHECK

To check parity (odd or even).

Z = 8-bit character (including parity).

Solution:

```
        AZE     A       Clear A
        ALD     Y,10    8 stage counter
LL      SHC     Z       Bit 0 to Link & Bit 7
        RLK     A,A
        RDC     Y,Y     Decrement counter by 1
        SKZ     2
        JMP     LL      Loop back
        SHR     A
        SKS     2
        JMP     EVEN    EVEN Parity OK
        JMP     ODD     ODD Parity OK
```

2.3.8 PARITY GENERATION

Generate Parity (odd or even).

Z = 8-bit character (including parity position).

Solution:

```
        AZE     A       Zero to A
        ALD     Y,7
LL      SHC     Z       Shift circular.,
                        Bit 0 ⟶ Bit 7 & Link.
        RLK     A,A     Add Link to A register
                        Count logic "1" 's
        RDC     Y,Y
        SKZ     2       All 7 bits counted
        JMP     LL      No: Loop back
        SHR     A       Parity to Link Bit
        SHL     Z       Shift Link to Bit 7
```

* (Z) = Character w/even parity (Parity to Bit 7)

```
        AXR     Z,200.....  Toggle parity bit for odd
                            parity.
```

2.3.9 INDEXED JUMP

Indexed jump through a jump vector table at TA.

Solution:

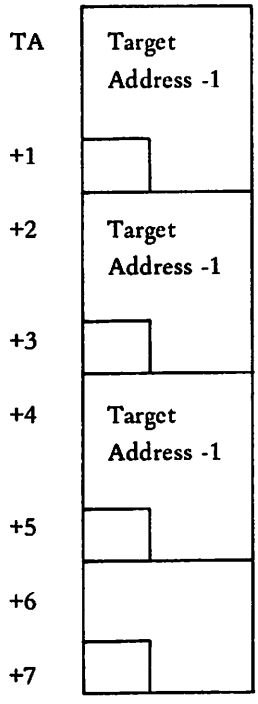
```
        GØL     A,I
```

* B contains the effective operand address viz the target address -1.

```
        RTR     B,P     Execute jump
```

Location 0 = ZA TA,X

X = Index



2.3.10 RANDOM MAPPING FUNCTION

Input table mapped into output table with a random mapping function.

Solution:

```
*       Pointers 0, 2, 4 contain the
*       references to Input, Output, and the
*       translator tables respectively.

*       0    IN-1,Y,I
*       2    OU,Z
*       4    TR-1,Y,I

*       Translator table is in 1 to 1
*       correspondence with the input table:
*       The translator table contains the
*       indexes to the output table.

        AZE  Y

LP      GØL  A,I     Fetch input character
        G4L  Z,I     Fetch index to OU table.
                       (Mapping function)
        G2S  A,I     Store in the appropriate
                       entry in OU table.
        ASU  Y,C     Test for end of table
        SKP  6

        AAD  Y,C
        JMP  LP      Loop Back
        SKF  0       No operation
        JMP  EXIT    EXIT
```

Note

C = Table Length

2.3.11 HOLLERITH CODE TO ASCII CODE CONVERSION

To convert Hollerith code (card input) to ASCII (printer output), proceed as follows:

Given:

A 12-bit code, corresponding to rows 12 through 9 of a card column, is received for translation to a printer-compatible 8-bit code.

Solution:

The high-order (12-11-0) punch positions are isolated for use as a major index on a list of pointers to groups of ASCII characters. The low-order (1-9) punch positions are isolated for use as a minor index pointing to a character within a group. Multiple punches on low-order positions require a special technique in generating the minor index.

Each bit found set causes a weighting factor to be included in a summation counter. The weighting factors are assigned by relative bit position within the field and correspond to the equivalent card row.

Using the two indices a memory location is addressed to complete the translation from Hollerith to ASCII.

Refer to the following program listing.

CONVERT HOLLERITH CODE TO ASCII

```
     ALD   X,K       LOCATN OF TBL CONTAING MAJOR INDEX OPERATOR
     RTR   B,A       12-BIT HOLLERITH CODE
     TBB             ROWS 12-1 OF CARD ARE RT-JUSTIFIED IN B-REG
     AAD   X,B       LOCATN OF PARTICULAR ASCII CHARACTER GRP
     RTR   B,Z
     AND   X,376     FORCE ROW-1 BIT  TO ZERO
     SHR   Z         IF ROW-1 BIT POSITION
     AZE   Z           CONTAINS A PUNCH THE ROW
     RLK   Z,Z         WEIGHTING FACTOR IS PLACED INTO SUM CNTR
     ALD   Y,11      ROW 9 AND ITS CORRESPONDING WEIGHTING FACTOR
LP   SHR   A         SHIFT ONE BIT TO LINK FOR TEST
     SKR   4
     RTR   Y,B       IF BIT Y (ROW Y)  IS SET ADD ITS
     AAD   Z,B         ROW POSITION TO SUM CNTR
     RDC   Y,Y
     SKZ   2
     JMP   LP        ALL BITS NOT CHKD  -  GO BACK ON LOOP
     GØL   A,X,I     PICK UP ASCII(X,Z)

K    ZA  T1,Z        NO PUNCH IN ROWS 12-1
     ZA  T2,Z        PUNCH IN ROW Ø
     ZA  T3,Z        PUNCH IN ROW 11
     ZA  Ø           FILLER
     ZA  T4,Z        PUNCH IN ROW 12



T4   ZC  253         "+"     12
     ZC  3Ø1         "A"     12-1
     ZC  3Ø2         "B"     12-2
     ZC  3Ø3         "C"     12-3
     ZC  3Ø4         "D"     12-4
     ZC  3Ø5         "E"     12-5
     ZC  3Ø6         "F"     12-6
     ZC  3Ø7         "G"     12-7
     ZC  31Ø         "H"     12-8
     ZC  311         "I"     12-9
     ZC  277         "?"     12-2-8
     ZC  256         "."     12-3-8
     ZC  251         ")"     12-4-8
     ZC  333         "["     12-5-8
     ZC  274         "<"     12-6-8
     ZC  246         "ε"     12-7-8
```

T3	ZC	255	"_"	11
	ZC	312	"J"	11-1
	ZC	313	"K"	11-2
	ZC	314	"L"	11-3
	ZC	315	"M"	11-4
	ZC	316	"N"	11-5
	ZC	317	"O"	11-6
	ZC	320	"P"	11-7
	ZC	321	"Q"	11-8
	ZC	322	"R"	11-9
	ZC	241	"!"	11-2-8
	ZC	244	"$"	11-3-8
	ZC	252	"*"	11-4-8
	ZC	335	"]"	11-5-8
	ZC	273	";"	11-6-8
	ZC	245	"%"	11-7-8
T2	ZC	260	"Ø"	Ø
	ZC	257	"/"	Ø-1
	ZC	323	"S"	Ø-2
	ZC	324	"T"	Ø-3
	ZC	325	"U"	Ø-4
	ZC	326	"V"	Ø-5
	ZC	327	"W"	Ø-6
	ZC	330	"X"	Ø-7
	ZC	331	"Y"	Ø-8
	ZC	332	"Z"	Ø-9
	ZC	300	"@"	Ø-2-8
	ZC	254	","	Ø-3-8
	ZC	250	"("	Ø-4-8
	ZC	242	"""	Ø-5-8
	ZC	334	"\"	Ø-6-8
	ZC	243	"#"	Ø-7-8
T1	ZC	240	"SP"	-----
	ZC	261	"1"	--1
	ZC	262	"2"	--2
	ZC	263	"3"	--3
	ZC	264	"4"	--4
	ZC	265	"5"	--5
	ZC	266	"6"	--6
	ZC	267	"7"	--7
	ZC	270	"8"	--8
	ZC	271	"9"	--9
	ZC	337	"_"	--2-8
	ZC	275	"="	--3-8
	ZC	247	"'"	--4-8
	ZC	272	":"	--5-8
	ZC	276	">"	--6-8
	ZC	336	"+"	--7-8

SECTION 3
SPC-12 ARITHMETIC LIBRARY

3.1 GENERAL

The arithmetic library consists of a group of subroutines which perform double and triple precision fixed point, signed arithmetic using the A, Z, and sometimes Y registers as one operand and the PSEUDO-A registers as the other operand. Its contents are as follows:

Arithmetic Library Working Storage.

Save Registers Subroutine (SR).

Restore Registers Subroutine (RR).

Double Precision Add (AD).

Double Precision Subtract (SD).

Double Precision Multiply (MS).

Double Precision Divide (DD).

Triple Precision Add (AT).

Triple Precision Subtract (ST).

Triple Precision Multiply (MT).

Triple Precision Divide (DT).

3.2 USING THE ARITHMETIC LIBRARY

In using the arithmetic library, the following comments will be helpful:

a. The Arithmetic Library Working Storage is used by the different routines as indicated in the comments field of the Arithmetic Library Working Storage listing. For space requirements, insertion or omission of storage may be made per user requirement.

b. The Save Registers and Restore Registers subroutines are included as part of the total SPC-12 Arithmetic Library Package. These subroutines have been included for the case of the addition of the more complex user supplied arithmetic routines (such as Sin (X)/Cos (X), e^{-X}, etc.); they are not needed for the arithmetic routines in this arithmetic library. Omission or insertion is left to the user per system requirement.

c. The Double Precision Add and Double Precision Subtract subroutines exit through common coding contained within the Double Precision Add subroutine.**

d. The Triple Precision Add and Triple Precision Subtract subroutines exit through common coding contained within the Triple Precision Add subroutine.**

e. The Double Precision Multiply and Double Precision Divide subroutines exit through common coding contained within the Double Precision Multiply subroutine.**

f. The Arithmetic Library subroutines are supplied in source tape form. To go from source to object, use GA Conversational Assembly System program (CAS II) 88Z00003A. GA

**Should only one subroutine be desired and not the other for a particular system, the common coding must be incorporated. The addresses shown in the program listings in Section 4 are relative and for example purposes only. The user selects the actual addresses desired during the assembly process.

recommended shared command bytes. GA RCMD CCB, 88Y00015A, may be wed. The program listings shown in section 4 uses the Recommended Command Code Bytes.

g. Register contents not obviously used by the subroutine, as indicated in calling sequence are either unused or preserved and restored at the end of the subroutine.

h. For purposes of planning, listings of the tapes may be made by listing the source paper tape with the teletype ASR in the LOCAL mode.

3.3 MEMORY REQUIREMENT

The programmer may assemble the Arithmetic Library using the GA Recommended Shared Commands in order to conserve memory locations. The subroutine length in table 3-1 show the Arithmetic Library as assembled without use of shared commands.

Table 3-1. Arithmetic Library Subroutine Length

Name	Length (Unshared)
Arithmetic Library Working Storage	43_8
Save Registers Subroutine	52_8
Restore Registers Subroutine	32_8
Double Precision Add	34_8
Double Precision Subtract	33_8*
Double Precision Multiply	274_8
Double Precision Divide	330_8*
Triple Precision Add	50_8
Triple Precision Subtract	47_8*
Triple Precision Multiply	457
Triple Precision Divide	543

*These subroutines are interdependent and utilize common coding. Since the arithmetic package is considered an entity, the lengths shown do not include the common coding used.

3.4 PROGRAM APPLICATION DOCUMENT

3.4.1 SAVE REGISTERS SUBROUTINE (SR)

Purpose:

The contents of the A, X, Y, and Z registers are preserved in the arithmetic library storage, intended for use with complex math routines.

Input:

N/A (Not Applicable).

Output:

N/A

Tables and Formats:

N/A

System Parameters:

N/A

Linkage:

RIC P, B
JMP SR

Execution Time:

Approximately 0.117 ms.

Core Requirements:

52 octal locations (unshared).

Method:

Remarks:

References:

W: Arithmetic Library Storage.

3.4.2 RESTORE REGISTER SUBROUTINE (RR)

Purpose:

The contents of the A, X, Y, and Z registers are restored from the arithmetic library storage. Contents are originally stored by the save registers subroutine.

Input:

N/A (Not Applicable).

Output:

N/A

Tables and Formats:

N/A

System Parameters:

N/A

Linkage:

```
RIC  P,B
JMP  RR
```

Execution Time:

0.082 ms.

Core Requirements:

32 octal locations (unshared).

Method:

Remarks:

References:

W: Arithmetic Library Storage.

3.4.3 ADD ARITHMETIC SUBROUTINE (AD) · DOUBLE PRECISION 88Y00004A REV B

Purpose:

Add operand specified in Pseudo-A Register to contents of A and Z registers and store results in Pseudo-A register.

Input:

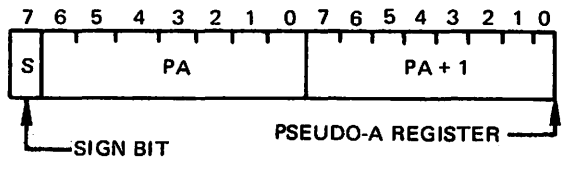
Registers A and Z contain the augend and Pseudo-A contains the addend.

Output:

Results in Pseudo-A register.

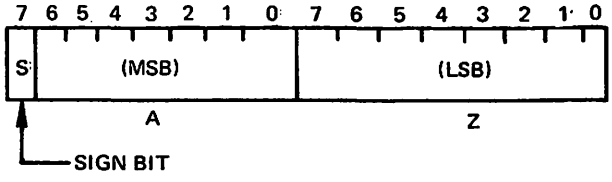
Tables and Formats:

Addend



System Parameters:

Augend in physical registers A and Z.



Linkage:

RIC P, B
JMP AD

Execution Time:

Approx. 0.078 ms.

Core Requirements:

34 octal locations (unshared).

Method:

The least significant byte of Pseudo-A (PA+1) is added to contents of Z register. The link bit is allowed to propagate into the contents of the A register. The more significant byte of Pseudo-A (PA) is added to contents of the A register. The result is stored in the Pseudo-A register.

Remarks:

The results of the addition will be in the A and Z registers as well as in Pseudo-A. Registers X and Y are not used by this subroutine.

References:

PA Arithmetic Library working storage.

3.4.4 SUBTRACT ARITHMETIC SUBROUTINE (SD) · DOUBLE PRECISION 88Y00006A REV B

Purpose:

Subtract operand specified in A and Z registers from Pseudo-A register and store results in Pseudo-A register.

Input:

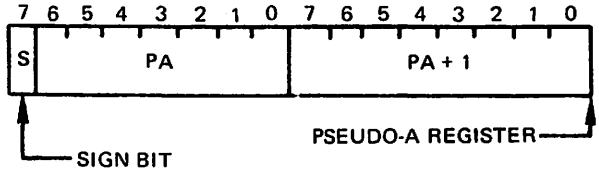
Registers A and Z contain the subtrahend.

Output:

Results in Pseudo-A register.

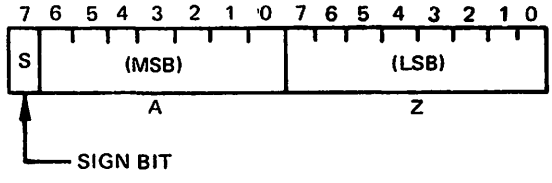
Tables and Formats:

Minuend in Pseudo-A register.



System Parameters:

Subtrahend in Registers A and Z.



Linkage:

RIC P, B
JMP SD

Execution Time:

Approx. 0.1 ms.

Core Requirements:

33 octal locations (unshared).

Method:

The contents of the A register is subtracted from the more significant byte of Pseudo-A register (PA). The contents of the Z Register are subtracted from the least significant byte of Pseudo-A register (PA+1). If the link bit is set, the contents of the A register is decremented by one.

Remarks:

Results of subtract is in A and Z registers as well as in Pseudo-A register. Registers X, and Y are not used by this subroutine.

References:

PA Arithmetic Library working storage
AD Double Precision Add.

3.4.5 MULTIPLY ARITHMETIC SUBROUTINE (MS) DOUBLE PRECISION 88Y00008A REV B

Purpose:

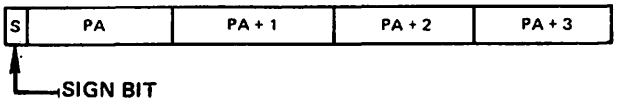
To multiply double precision number in Pseudo-A register with contents of A and Z registers and store results in Pseudo-A registers.

Input:

The Pseudo-A register contains the multiplicand and registers A and Z contain the multiplier.

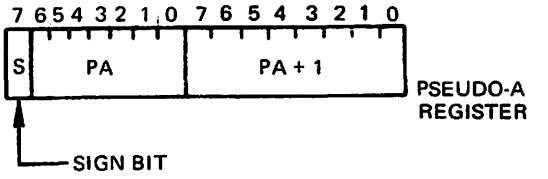
Output:

Results of multiplication is in the Pseudo-A registers.



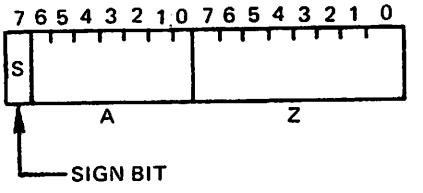
Tables and Formats:

Pseudo A
Multiplicand.



System Parameters:

Multiplier
contents of A and Z
register.



Linkage:

RIC P, B
JMP MS

Execution Time:

Approx. 0.80 ms min.
Approx. 1.28 ms max.

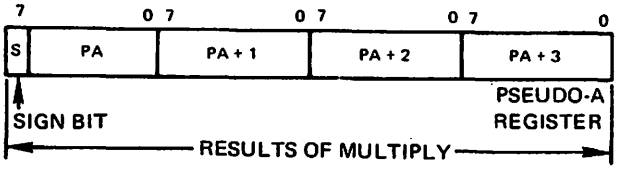
Core Requirements:

274 octal locations.

Method:

The product is formed by successive adds and shifts. Successive additions of the multiplicand are done for each multiply indicated by a bit shifting out of the multiplier. Positive numbers are used during the process and an end correction is made for negative multipliers.

Remarks:



Algorithm for binary point:

$(N1 @ B1) \times (N2 @ B2) = N1 \cdot N2 @ B1 + B2 + 1)$

Where:

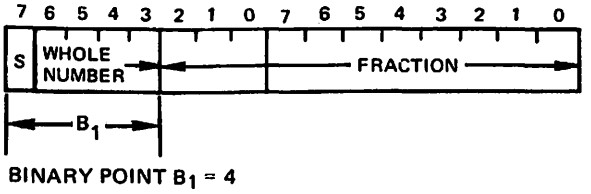
N1 and N2 are any binary numbers

and:

B1 and B2 are the binary points for these numbers

The binary point is the position which separates the integer from the fractional portion of a number. Reference is made, counting from the left, from a fixed position between the sign bit and the next bit position to the right (binary scale = BO).

Example:



BINARY POINT $B_1 = 4$

For double precision right justified integers, B = 15. The resulting product will also be right justified, B = 31 (15 + 15 + 1). It is the user's task to maintain the binary point.

At exit, the contents of the A and Z registers are undefined.

References:

PA Arithmetic Library Storage.

3.4.6 DIVIDE ARITHMETIC SUBROUTINE (DD) · DOUBLE PRECISION 88Y00010A REV B

Purpose:

Divide contents of Pseudo-A registers by the divisor in registers A and Z. Store quotient and remainder in Pseudo-A.

Input:

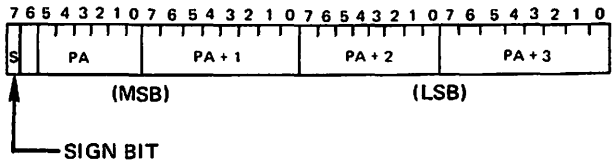
Registers A and Z contain the divisor. Pseudo-A contains the dividend.

Output:

Quotient in least significant bytes of Pseudo-A and remainder in most significant bytes of Pseudo-A.

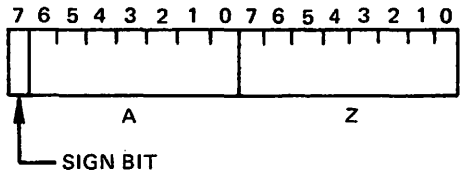
Tables and Formats:

Pseudo-A
Dividend.



System Parameters:

Registers A, Z
Divisor.



Linkage:

RIC P, B
JMP DD

Execution Time:

Approx. 2.3 ms min.
Approx. 2.9 ms max.

Core Requirements:

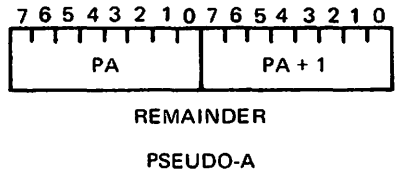
330 octal locations.

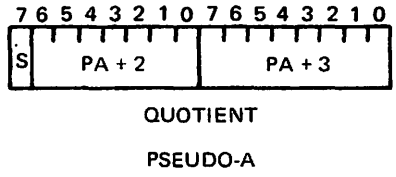
Method:

Division is accomplished by subtracting the divisor from the numerator until the subtraction yields a negative result. The numerator is loaded into the X, Y, A and Z registers. A double precision subtract is performed between the divisor and the contents of X and Y. If the result is negative the divisor is added back to X and Y and a bit is zeroed.

If the result is positive (the divisor goes into the numerator) a bit is set. The contents of X, Y, A and Z are shifted and the above is repeated until all of the numerator has had the divisor subtracted from it. The accumulation of the setting and resetting of a bit as the subtraction occurs forms the quotient. The remainder is stored in the two most significant bytes of the Pseudo-A. The quotient is stored in the least significant two bytes of Pseudo-A.

Remarks:





Common Rule: The divisor must be greater than the most significant half of the dividend. Algorithm for binary point is reverse that of the double precision multiply.

NOTE

For double precision right justified integer (B=31), the resulting quotient will also be right justified (B=15).

References:

PA Arithmetic Library Storage.
MS (Common Exit) Double Precision Multiply.

3.4.7 ADD ARITHMETIC SUBROUTINE (AT) · TRIPLE PRECISION 88Y00005A REV B

Purpose:

Add operand specified in Pseudo-A register to contents of A, Y, and Z registers, and store results in Pseudo-A register.

Input:

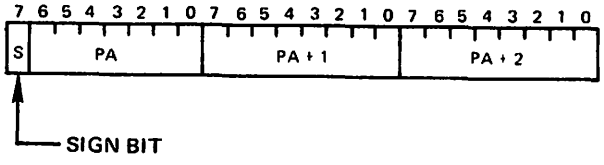
Registers A, Y, and Z contain augend.

Output:

Results in Pseudo-A.

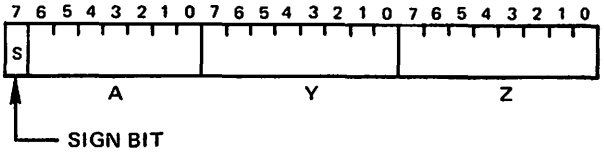
Tables and Formats:

Addend
Pseudo-A register.



System Parameters:

Augend in registers
A, Y, and Z.



Linkage:

RIC P, B
JMP AT

Execution Time:

Approx. 0.108 ms.

Core Requirements:

50 Octal Locations.

Method:

The least significant byte of Pseudo-A is added to the Z register. The link bit is allowed to propagate into the Y and A registers. The middle byte is added to the Y register and the link bit is allowed to propagate into the A register. The most significant byte of Pseudo-A is added to the A register and the contents of registers A, Y, and Z are stored in Pseudo-A.

Remarks:

References:

PA Arithmetic Library Storage.

3.4.8 SUBTRACT ARITHMETIC SUBROUTINE (ST) · TRIPLE PRECISION 88Y00007A REV B

Purpose:

Subtract operand in A, Y, and Z from the Pseudo-A register and store results in the Pseudo-A register.

Input:

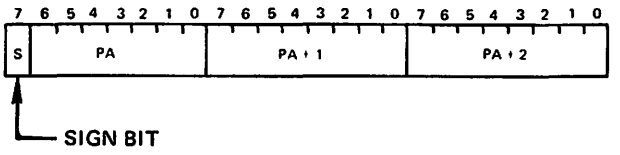
Registers A, Y, and Z contain the subtrahend and Pseudo-A contains the minuend.

Output:

Results in Pseudo-A register.

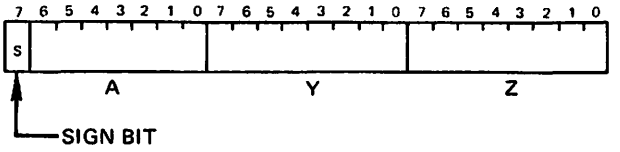
Tables and Formats:

Pseudo-A register
minuend.



System Parameters:

Subtrahend
registers A, Y, Z.



Linkage:

RIC P, B
JMP ST

Execution Time:

Approx. 0.140 ms.

Core Requirements:

47 octal locations.

Method:

The contents of A, Y, and Z registers is subtracted from the Pseudo-A register. Subtraction is performed on a byte by byte basis and the Link is used to control borrowing from the next more significant byte.

Remarks:

References:

PA Arithmetic Library Storage.
AT (Common Exit) Triple Precision Add.

3.4.9 MULTIPLY ARITHMETIC SUBROUTINE (MT) · TRIPLE PRECISION 88Y00009A REV B

Purpose:

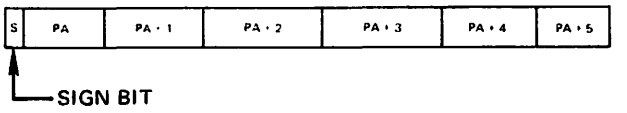
To multiply the Pseudo-A register with the contents of A, Y, and Z registers and store the results in Pseudo-A.

Input:

The Pseudo-A register contains the multiplicand, and registers A, Y, and Z contain the multiplier.

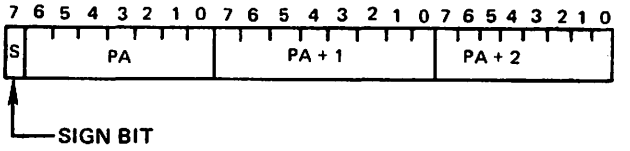
Output:

The results of the multiplication is in the Pseudo-A registers.



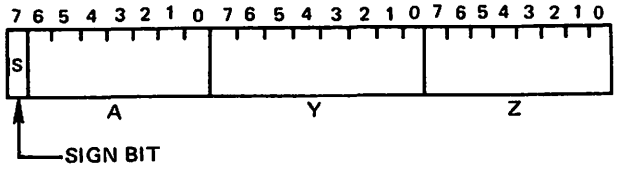
Tables and Formats:

Pseudo-A
multiplicand.



System Parameters:

Multiplier
contents of A, Y, and Z.



Linkage:

RIC P, B
JMP MT

Execution Time:

2.9 ms min.
5.6 ms max.

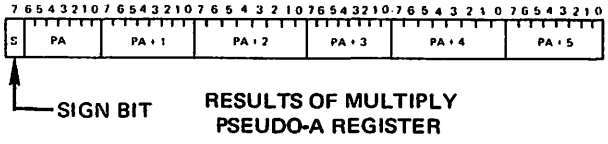
Core Requirements:

457 octal locations.

Method:

The product is formed by successive adds and shifts. Successive adds of the multiplicand is done for each multiply, indicated by a bit shifting out of the multiplier. Positive numbers are used during the process and an end correction is made for negative multipliers.

Remarks:



Algorithm for binary point:

(N1 @ B1) x (N2 @ B2) = (N1N2 @ B1 + B2 + 1
(See Algorithm for Multiply Double Precision)

References:

PA Arithmetic Library Storage.

3.4.10 DIVIDE ARITHMETIC SUBROUTINES (DT) · TRIPLE PRECISION 88Y00011A REV B

Purpose:

Divide the contents of Pseudo-A registers by the divisor in registers A, Y, and Z. Store the quotient in least significant bytes of Pseudo-A and the remainder in most significant bytes of Pseudo-A.

Input:

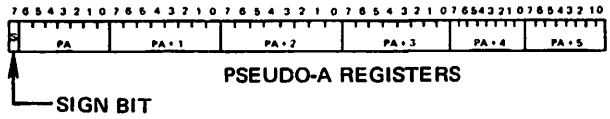
Registers A, Y, and Z contain the divisor. Pseudo-A register contains the dividend.

Output:

Remainder in most significant bytes of Pseudo-A and quotient in least significant bytes of Pseudo-A.

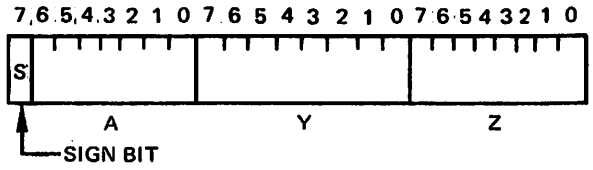
Tables and Formats:

Dividend.



System Parameters:

Registers A, Y, and Z
divisor.



Linkage:

RIC P, B
JMP DT

Execution Time:

3.8 ms min.
5.6 ms max.

Core Requirements:

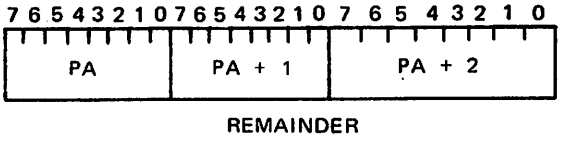
543 octal locations.

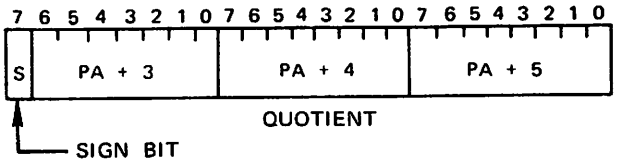
Method:

Division is accomplished by subtracting the divisor from the numerator until the subtraction yields a negative result. A triple precision subtract is performed between the divisor and the numerator.

Each time the divisor goes into the numerator a bit is set. The numerator is shifted left after each triple precision subtraction until all 46 bits of the numerator have been used. The quotient is the accumulation of the bits set and is stored in the least significant bytes of Pseudo-A. The remainder is what is left over after the shifting and it is stored in the most significant bytes of Pseudo-A.

Remarks:





References:

PA Arithmetic Library Storage.

SECTION 4
ARITHMETIC PACKAGE

4.1 GENERAL

The listings associated with the arithmetic library subroutines are included in this section. They are presented in the following order:

a. Arithmetic Library Working Storage

b. Save Registers

c. Restore Registers

d. Add Arithmetic Subroutine (Double Precision)

e. Subtract Arithmetic Subroutine (Double Precision)

f. Multiply (Double Precision)

g. Divide Arithmetic Subroutine (Double Precision)

h. Add Arithmetic Subroutine (Triple Precision)

i. Subtract Arithmetic Subroutine (Triple Precision)

j. Multiply (Triple Precision)

k. Divide Arithmetic Subroutine (Triple Precision)

4.2 ARITHMETIC LIBRARY WORKING STORAGE

```
*       ARITHMETIC LIBRARY WORKING STORAGE
*    88Y000016A     REV B      05-07-69
*
* * * * * *** *** * * * * *** * * * * * * * * * * * * * * * ***  ***
*  *
* *                          * USE AS REQUIRED PER COMMENTS  *    *
*  *
* * * * * *** *** * * * * *** * * * * * * * * * * * * * * * ***  ***
PA        ZC          0         PSUEDO-A
          ZC          0           +1
          ZC          0           +2
          ZC          0           +3
          ZC          0           +4
          ZC          0           +5
:Z        ZR          2         USERS RETRN ADDR-AD,SD,MS    ,DD,AT,
* * * * * * * * * *   * *** *** *** *** *** * * *** * * *** * * *** ***
:Y        ZR          2         NESTED SUBR RETURN ADDRESS (SPARE-USAGE)
* * * * * * * * * *   * *** *** *** *** *** * * *** * * *** * * *** ***
:X        ZR          5         TEMP.STG.    (MS USES 5 LOCS)
*                                            (MT USES 3 LOCS)
*                                           (DD USES 4 LOCS)
*                                           (DD USES 5 LOCS)
:W        ZR          12        TEMP. STG. (USED BY SR,RR ONLY)
:V        ZR          6         TEMP.STG.  (DT)
* * * * * * * * * *   * *** *** *** *** *** * * *** * * *** * * *** ***
:U        ZR          4         TEMP.STG.(SPARE USAGE)
* * * * * * * * * *   * *** *** *** *** *** * * *** * * *** * * *** ***
P
```

4.3 SAVE REGISTERS

```
*
*
*                              CALLING SEQ RIC P,B
*     SAVE REGISTERS                        JMP SR
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                              USE PER SYSTEM REQUIREMENT;
*                    MAY BE OMITTED IF NOT DESIRED BY SYSTEM
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
SR      STB     :W
        TBB
        STB     :W+1
        RTR     A,B         SAVE A REG
        STB     :W+2
        TBB
        STB     :W+3
        RTR     X,B         SAVE  X REG
        STB     :W+4
        TBB
        STB     :W+5
        RTR     Y,B         SAVE  Y REG
        STB     :W+6
        TBB
        STB     :W+7
        RTR     Z,B         SAVE  Z REG
        STB     :W+1Ø
        TBB
        STB     :W+11
        ELB     :W          RETURN TO CALLER
        RIC     B,P
```

4.4 RESTORE REGISTERS

```
*
*
*                                CALLING SEQ  RIC P,B
*     RESTORE RESISTERS                       JMP RR
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                USE PER SYSTEM REQUIREMENT;
*                    MAY BE OMITTED IF NOT DESIRED BY SYSTEM
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
RR       STB        :W
         TBB
         STB        :W+1
         ELB        :W+2          RESTORE A
         RTR        B,A
         ELB        :W+4          RESTORE X
         RTR        B,X
         ELB        :W+6          RESTORE Y
         RTR        B,Y
         ELB        :W+1Ø
         RTR        B,Z           RESTORE Z
         ELB        :W
         RIC        B,P
```

4.5 ADD ARITHMETIC SUBROUTINE (DOUBLE PRECISION)

```
*****************          ***************** *                    ****************************
*
*       ADD ARITHMETIC SUBROUTINE (DOUBLE PRECISION)
* *      88Y00004A     REV B     05-07-69
*
*  *    CALLING SEQUENCE -          RIC  P, B
* *  *                              JMP  AD
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*       ENTRY              PA,PA+1       (ADDEND)
*                        + A,  Z         (AUGEND)
*
*       EXIT               PA, PA+1       (ANSWER)
*****************          ***************** *                    ****************************
AD      STB       :Z              SAVE RETURN
        TBB
        STB       :Z+1
        LDB       PA+1            ADD LEAST TO LEAST
        AAD       Z,B
        RLK       A,A           ADD LINK TO A
        LDB       PA
        AAD       A,B             ADD MOST TO MOST
<1      RTR       A,B
        STB       PA              STORE MOST
        RTR       Z,B
        STB       PA+1            STORE LEAST
        ELB       :Z
        RIC       B,P             RETURN
```

4.6 SUBTRACT ARITHMETIC SUBROUTINE (DOUBLE PRECISION)

```
*
*****************************************************************************
*
*       SUBTRACT ARITHMETIC SUBROUTINE  (DOUBLE PRECISION)
*  *      88Y00006A   REV B    05-07-69
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  *    CALLING  SEQUENCE -              RIC  P, B
* * *                                    JMP SD
*
*       ENTRY              PA,PA+1       (MINUEND)
*                        - A,  Z         (SUBTRAHEND)
*
*       EXIT               PA, PA+1      (ANSWER)
*****************************************************************************
SD      STB        :Z            SAVE RETURN
        TBB
        STB        :Z+1
        AXR        A,377         TWO,S COMPLEMENT OF SUBTRAHEND
        AXR        Z,B
        RIC        Z,Z
        RLK        A,A
        LDB        PA            SUBTRACT MOST FROM MOST
        AAD        A,B
        LDB        PA+1          SUBTRACT LEAST FROM LEAST
        AAD        Z,B
        RLK        A,A
        JMP        <1
*
*
```

4.7 DOUBLE PRECISION MULTIPLY

```
*
*     D O U B L E       P R E C I S I O N       M U L T I P L Y
*  *     88Y0000 8A      REV B     07-22-69
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  *    CALLING SEQUENCE -                RIC  P, B
* *  *                                    JMP  MS
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  ENTRY-                PA, PA+1      (MULTIPLICAND)
*                      *  A, Z         (MULTIPLIER)
* *
*  EXIT-              PA,+1,+2,+3        (ANSWER)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
* * *          USES   SR, RR, AD, SD, MS, AND ARITH. LIB.STG. S
*
*       (X,Y REGISTERS ARE SAVED UPON ENTRY AND RESTORED UPON
*     (N1 @ B1)  TIMES  (N2  @ B2)
*
*
*     (N1 TIMES N2)  @  (B1 + B2 + 1)
*     N = NUMBER (16 BITS IN COMPLEMENT FORM)
*     B = BINARY POINT
*     BØ IS BETWEEN THE TWO MOST SIGNIFICANT BITS
*
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*
*
MS      STB       :Z            SAVE RETURN
        TBB
        STB       :Z+1
        RTR       X,B
        STB       :X
        TBB
        STB       :X+1
        RTR       Y,B           SAVE X,Y  (XX,YY INCL)
        STB       :X+2
        TBB
        STB       :X+3
        AND       A,377         INSURE ONLY 8 BITS IN A AND Z EACH
        AND       Z,B
        RTR       A,X
        LDB       PA
        AXR       X,B
```

```
        RTR     X,B
        STB     :X+4            BIT 7, =Ø POS  ;  =1 NEG.  (RESUUS)
        SKM     2
        JMP     <F              RESULTS POS.; CK MULTIPLIER IF NEG.;
<A      LDB     PA              CK MULTIPLICAND; IF NEG. , MAKE POS
        SKM     2
        JMP     <B              MAKE MULTIPLIER POS.
        RTR     B,X
        LDB     PA+1
        RTR     B,Y
        AXR     X,377
        AXR     Y,B
        RIC     Y,Y
        RLK     X,X
        RTR     Y,B
        STB     PA+1
        RTR     X,B
        STB     PA
        JMP     <C
<B      AXR     A,377           MAKE MULTIPLIER POS.
        AXR     Z,B
        RIC     Z,Z
        RLK     A,A
<C      AZE     X
        AZE     Y
        SHL     A               SHIFT MULTIPLIER
        SHL     Z
        ALD     ZZ,17
        SKS     2               DO WE MULTIPLY?
        JMP     <E               NO
<D      LDB     PA+1            YES  (MULTIPLY)
        AAD     Y,B
        RLK     X,X
        LDB     PA
        AAD     X,B
*       * *      * * * * * * * * * * * * * * * * * * * * *
<E      RDC     ZZ,B
        SHR     X               SHIFT PARTIAL PROD.  (X,Y,A,Z)
        SHL     Y
        SHL     A
        SHL     Z
*       * *      * * * * * * * * * * * * * * * * * * * * *
        RTR     B,ZZ            DECR SHIFT CTR
        SKM     6               LAST SHIFT ?
        SKS     2                NO,  (DO WE MULTIPLY?)
        JMP     <E                    ((NO)
        JMP     <D
        SKF     Ø               (NOP)
        LDB     :X+4            IS RESULTS TO BE NEG. ?
        SKM     2               YES
        JMP     <3              STORE RESULTS (X,Y,A,Z)
```

```
         AXR     Z,377
         AXR     A,B
         AXR     Y,B
         AXR     X,B
         RIC     Z,Z
         RLK     A,A
         RLK     Y,Y
         RLK     X,X
*
<3       RTR     X,B          STORE RESULTS IN PSEUDO A, AND REST
         STB     PA
         RTR     Y,B
         STB     PA+1
         RTR     A,B
         STB     PA+2
         RTR     Z,B
         STB     PA+3
         ELB     :X
         RTR     B,X
         ELB     :X+2
         RTR     B,Y
         ELB     :Z
         RIC     B,P
*                                                          *
<F       RTR     A,B          IS MULTIPLIER NEG. ?
         SKM     2            YES
         JMP     <C           NO  (BOTH POS.)
         AXR     A,377
         AXR     Z,B
         RIC     Z,Z
         RLK     A,A
         JMP     <A           GO MAKE MULTIPLICAND POS.
*                                                          *
**********    * *     * *     *    *   *   *  * * * * * *
```

4.8 DIVIDE ARITHMETIC SUBROUTINE (DOUBLE PRECISION)

```
*********   * *   *  *    *    *   *   *  * * * * * *
*   DIVIDE ARITHMETIC SUBROUTINE (DOUBLE PRECISION)
*   88Y000-10A  REV B  5/07/69
*  *     CALLING  SEQUENCE -              RIC  P, B
* * *                                     JMP  DD
*
*        ENTRY              PA, PA+1, PA+2, PA+3
*                            A,  Z       (DIVISOR)
*
*        EXIT               PA, PA+1,  PA+2, PA+3       (DIVIDEND)
*                           REMAINDER, QUOTIENT
*********************************************************************************
DD       STB       :Z              SAVE RETURN
         TBB
         STB       :Z+1
         RTR       X,B             STORE X AND Y REGS.
         STB       :X
         TBB
         STB       :X+1
         RTR       Y,B
         STB       :X+2
         TBB
         STB       :X+3
         RTR       A,B             IS DIVISOR NEGATIVE
         SKM       2
         JMP       <G              NO
         AXR       Z,377           YES
         AXR       A,B
         RIC       Z,Z
         RLK       A,A
         LDB       :Z+1            TRIGGER SIGN FLAG
         RTR       B,Y
         AXR       Y,200
         RTR       Y,B
         STB       :Z+1
<G       RTR       A,B
         STB       :1
         RTR       Z,B             STORE DIVISOR
         STB       :2
         LDB       PA+3            PUT DIVIDEND IN X,Y,A AND Z
         RTR       B,Z
         LDB       PA+2
         RTR       B,A
         LDB       PA+1
         RTR       B,Y
         LDB       PA
         RTR       B,X
```

```
        SKM     2           IS DIVIDEND NEGATIVE
        JMP     <H          NO
        AXR     Z,377       NEGATE
        AXR     A,B
        AXR     Y,B
        AXR     X,B
        RIC     Z,Z
        RLK     A,A
        RLK     Y,Y
        RLK     X,X
        RTR     Y,B
        STB     :3          (TEMP.STORE)
        LDB     :Z+1        TRIGGER SIGN FLAG
        RTR     B,Y
        AXR     Y,2ØØ
        RTR     Y,B
        STB     :Z+1
        ALD     Y,Ø         (RESTORE Y)
:3      ZL      %-1
<H      ALD     B,357
        STB     :3          STORE COUNTER
<I      ASU     X,Ø
:1      ZL      %-1
        ASU     Y,Ø
:2      ZL      %-1
        SKS     2
        RDC     X,X
        RTR     X,X         IS RESULT NEGATIVE
        SKM     6
        ALD     B,ØØ1
        JMP     <J          NO
        ZC      1ØØ         (NOP)
        LDB     :2          YES
        AAD     Y,B
        SKR     2
        RIC     X,X
        LDB     :1
        AAD     X,B
        TBB                 ZERO B REG, RESET QUOTIENT BIT
        TBB
<J      STB     :4
        LDB     :3          16 TIMES ?
        RIC     B,B
        SKR     2
        JMP     <K
        STB     :3
        RTR     X,B         SHIFT X,Y,A,Z LEFT ONE BIT
        AAD     X,B
        RTR     Y,B
        AAD     Y,B
```

```
        RLK     X,X
        RTR     A,B
        AAD     A,B
        RLK     Y,Y
        RTR     Z,B
        AAD     Z,B
        RLK     A,A
        AAD     Z,Ø
:4      ZL      %-1
        JMP     <I
%
<K      RTR     A,B             SHIFT A AND Z
        AAD     A,B
        RTR     Z,B
        AAD     Z,B
        RLK     A,A
        LDB     :4              ADD QUOTIENT BIT TO Z
        AAD     Z,B
        LDB     :Z+1             IS ANSWER NEGATIVE?
        SKM     2
        JMP     <L               NO
        RDC     B,B
        AXR     Z,B
        AXR     A,B
        RIC     Z,Z
        RLK     A,A
<L      JMP     <3              GO TO RESTOR5
%%%%%%%%%%%%%%%%%%%%%%%%%       %%%%%%%%%%%%%%%%%       %%%%%%%%%%
```

4.9 ADD ARITHMETIC SUBROUTINE (TRIPLE PRECISION)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*   ADD ARITHMETIC SUBROUTINE  (TRIPLE PRECISION)
*   88YØØØØ5A   REV.B    5/Ø7/69
*
*  *    CALLING  SEQUENCE -              RIC  P, B
* *  *                                   JMP  AT
*
*       ENTRY              PA, PA+1, PA+2      (ADDEND)
*                        + A,    Y,    Z       (AUGEND)
*
*       EXIT               PA, PA+1, PA+2         (ANSWER)
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
AT      STB      :Z            SAVE RETURN ADDRESS
        TBB
        STB      :Z+1
        LDB      PA+2          ADD LEAST TO LEAST
        AAD      Z,B
        RLK      Y,Y
        RLK      A,A
        LDB      PA+1          ADD MIDDLE TO MIDDLE
        AAD      Y,B
        RLK      A,A
        LDB      PA            ADD MOST TO MOST
        AAD      A,B
<2      RTR      A,B
        STB      PA            STORE MOST
        RTR      Y,B
        STB      PA+1          STORE MIDDLE
        RTR      Z,B
        STB      PA+2          STORE LEAST
        ELB      :Z
        RIC      B,P           RETURN
```

4.10 SUBTRACT ARITHMETIC SUBROUTINE (TRIPLE PRECISION)

```
*
*
*   SUBTRACT ARITHMETIC SUBROUTINE (TRIPLE PRECISION)
*   88Y00007A   REV. B   5/07/69
*
*  *    CALLING  SEQUENCE -              RIC  P, B
* *  *                                   JMP  ST
*
*       ENTRY              PA, PA+1, PA+2      (MINUEND)
*                        - A,   Y,   Z       ,(SUBTRAHEND)
*
*       EXIT               PA, PA+1, PA+2          (ANSWER)
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ST      STB        :Z            SAVE RETURN
        TBB
        STB        :Z+1
        AXR        A,377         GET TWO'S COMPLEMENT OF SUBTRAHEND
        AXR        Y,B
        AXR        Z,B
        RIC        Z,Z
        RLK        Y,Y
        RLK        A,A
        LDB        PA            SUB MOST FROM MOST
        AAD        A,B
        LDB        PA+1          SUB MIDDLE FROM MIDDLE
        AAD        Y,B
        RLK        A,A
        LDB        PA+2          SUB LEAST FROM LEAST
        AAD        Z,B
        RLK        Y,Y
        RLK        A,A
        JMP        <2
*
```

4.11 ARITHMETIC MULTIPLY SUBROUTINE (TRIPLE PRECISION)

```
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  *    ARITHMETIC MULTIPLY SUBROUTINE  (TRIPLE PRECISION)
* *   88YØØØØ9A    REV B  Ø5-Ø7-69
*  *
*  *   CALLING  SEQUENCE -             RIC  P, B
* *  *                                 JMP  MT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * *  * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * *     ENTRY-          PA, +1, +2        (MULTIPLICAND)
*                       *  A,  Y,  Z        (MULTIPLIER)
*  *
* *       EXIT   -        PA, +1, +2, +3, +4 +5       (ANSWER)
* *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
MT      STB       :Z          SAVE RETURN
        TBB
        STB       :Z+1
        RTR       X,B
        STB       :X          SAVE X
        TBB
        STB       :X+1
        AND       A,377       INSURE MULTIPLIER  8-BITS EA. BYTE
        AND       Y,B
        AND       Z,B
        TBB                   ZERO (PARTIAL) PRODUCT ACCUMULATORS
        STB       <5
        STB       <6
        STB       <7
        RTR       A,X         MULTIPLIER TO X-REG.
        LDB       PA
        AXR       X,B
        RTR       X,B
        STB       :X+2        BIT7; =Ø POS.,  =1 NEG.  (RESULTS)
        SKM       2
        JMP       <R          RESULTS POS.; CK MULTIPLIER IF NEG.
<M      LDB       PA          CK MULTIPLICAND; IF NEG.MAKE POS.
        SKM       2
        JMP       <N          GO MAKE MULTIPLIER POS.
        LDB       PA+2        (MAKE MULTIPLICAND POS.)
        RTR       B,X
        AXR       X,377
        RIC       X,B
        STB       PA+2
        LDB       PA+1
```

```
        RTR     B,X
        AXR     X,377
        RLK     X,B
        STB     PA+1
        LDB     PA
        RTR     B,X
        AXR     X,377
        RLK     X,B
        STB     PA
        JMP     <0
<N      AXR     A,377       MAKE MULTIPLIER POS.
        AXR     Y,B
        AXR     Z,B
        RIC     Z,Z
        RLK     Y,Y
        RLK     A,A
<0      SHL     A
        SHL     Y
        SHL     Z
        ALD     B,27        SET SHIFT CTR
        STB     <8
        SKS     2           DO WE MULTIPLY?
        JMP     <0          NO
<P      LDB     PA+2        YES,(MULTIPLY),ADD M-CAND TO MSB, P
        RTR     B,X
        LDB     <7
        AAD     X,B
        RTR     X,B
        STB     <7
        LDB     PA+1
        RLK     B,X
        RLK     YY,YY       SET AS LINK
        LDB     <6
        AAD     X,B
        RTR     X,B
        RLK     YY,YY       SET AS LINK
        STB     <6
        RTR     YY,X        XFR POSSIBLE LINK BIT TO X
        LDB     PA
        AAD     X,B
        LDB     <5
        AAD     X,B
        RTR     X,B
        STB     <5
<Q      ALD     X,Ø         DECREMENT SHIFT CTR
<8      ZL      *-1
        RDC     X,B
        STB     <8
        ALD     X,Ø
<5      ZL      *-1
```

```
        SHR     X           NO -  SHIFT 48 PART. PROD. THRU
        RTR     X,B
        STB     <5
        ALD     X,Ø
<6      ZL      %-1
        SHL     X
        RTR     X,B
        STB     <6
        ALD     X,Ø
<7      ZL      %-1
        SHL     X
        RTR     X,B
        STB     <7
        SHL     A
        SHL     Y
        SHL     Z
        LDB     <8          FETCH DECREMENTED SHIFT CTR COUNT
        SKM     6           LAST SHIFT ?
        SKS     2            NO, (DO WE MULTIPLY?)
        JMP     <Q                    (NO)
        JMP     <P                 (YES MULT)
        SKF     Ø           (NOP)
        LDB     :X+2        IS RESULTS TO BE NEG.?
        SKM     2           YES
        JMP     <4          NO, STORE RESULTS
        AXR     Z,377       YES, NEG. RESULTS   (:V+Ø,+1,+2,A,Y
        AXR     Y,B
        AXR     A,B
        RIC     Z,Z
        RLK     Y,Y
        RLK     A,A
        LDB      7
        RTR     B,X
        AXR     X,377
        RLK     X,B
        STB     <7
        LDB     <6
        RTR     B,X
        AXR     X,377
        RLK     X,B
        STB     <6
        LDB     <5
        RTR     B,X
        AXR     X,377
        RLK     X,B
        STB     <5
<4      LDB     <5          STORE RESULTS IN PSEUDO A, AND REST
        STB     PA
        LDB     <6
        STB     PA+1
```

```
        LDB     <7
        STB     PA+2
        RTR     A,B
        STB     PA+3
        RTR     Y,B
        STB     PA+4
        RTR     Z,B
        STB     PA+5
        ELB     :X
        RTR     B,X
        ELB     :Z
        RIC     B,P             RETURN
**** ****  ****  *  *       * * * * * *
<R      RTR     A,B
        SKM     2               MULTIPLIER NEG?
        JMP     <O              NO  (BOTH POS.)
        AXR     A,377           YES
        AXR     Y,B
        AXR     Z,B
        RIC     Z,Z
        RLK     Y,Y
        RLK     A,A
        JMP     <M              GO MAKE MULTIPLICAND POS.
```

4.12 DIVIDE ARITHMETIC SUBROUTINE (TRIPLE PRECISION)

```
*
*     DIVIDE ARITHMETIC SUBROUTINE (TRIPLE PRECISION)
*     88YØØØ11A  REV C   5/Ø7/69
*
*  *     CALLING  SEQUENCE -              RIC  P, B
* *  *                                     JMP  DT
*
*        ENTRY              PA,PA+1,PA+2,PA+3,PA+4,PA+5   (DIVIDEN
*                            A,   Y,   Z     (DIVISOR)
*
*        EXIT               PA,PA+1,PA+2  PA+3,PA+4,PA+5
*                          (REMAINDER)    (QUOTIENT)
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
DT       STB       :Z            SAVE RETURN
         TBB
         STB       :Z+1
         RTR       X,B
         STB       :X            SAVE X-REG.
         TBB
         STB       :X+1
         LDB       :Z+1
         RTR       B,X           X HOLDS SIGN FLAG
         RTR       A,B
         SKM       2             IS DIVISOR NEGATIVE?
         JMP       :A            NO
         AXR       Z,377         NEGATE DIVISOR
         AXR       Y,B
         AXR       A,B
         RIC       Z,Z
         RLK       Y,Y
         RLK       A,A
*
         AXR       X,2ØØ         TRIGGER SIGN FLAG
         RTR       X,B
         STB       :Z+1
         RTR       A,B           STORE DIVISOR
:A       STB       :5
         RTR       Y,B
         STB       :6
         RTR       Z,B
         STB       :7
*
```

```
        LDB     PA+5        MAKE NUMERATOR CONTIGUOUS
        STB     :V+5
        LDB     PA+4
        STB     :V+4
        LDB     PA+3
        STB     :V+3
        LDB     PA+2
        STB     :V+2
        LDB     PA+1
        STB     :V+1
        LDB     PA
        STB     :V
        SKM     2           IS NUMERATOR NEGATIVE
        JMP     :C          NO
        AXR     X,2ØØ       YES, TRIGGER SIGN FLAG
        RTR     X,B
        STB     :Z+1
        ALD     X,ØØ5
        ALD     Z,ØØ1
:B      RTR     Z,Y
        AZE     Z
        ALD     A,377
        LDB     :V,X
        AXR     A,B
        RTR     Y,B
        AAD     A,B
        RLK     Z,Z
        RTR     A,B
        STB     :V,X
        RDC     X,X
        SKM     2
        JMP     :B
:C      LDB     :V          LOAD PART OF NUMERATOR INTO REGS.
        RTR     B,X
        LDB     :V+1
        RTR     B,Y
        LDB     :V+2
        RTR     B,A
        LDB     :V+3
        RTR     B,Z
        TBB
        TBB
:D      STB     :X+4
        ALD     B,367       RESET COUNTER
        STB     :X+2
:E      ASU     X,Ø         SUBTRACT FROM NUMERATOR
:5      ZL      *-1
```

```
        ASU     Y,Ø
:6      ZL      %-1
        SKS     2
        RDC     X,X
        ASU     A,Ø
:7      ZL      %-1
        SKS     6
        ZC      1ØØ             (NOP)
        RDC     Y,Y
        SKS     2
        RDC     X,X
        RTR     X,X
        SKM     6
        ALD     B,ØØ1
        JMP     :F
        ZC      1ØØ             (NOP)
        LDB     :7              ADD BACK, DVSR DID NOT GO INTO NUM.
        AAD     A,B
        RLK     Y,Y
        RLK     X,X
        LDB     :6
        AAD     Y,B
        RLK     X,X
        LDB     :5
        AAD     X,B
        TBB
        TBB
:F      STB     :X+3            STORE QUOTIENT BIT
        LDB     :X+2            COUNTER REAC  8
        RIC     B,B
        SKR     2
        JMP     :H
        STB     :X+2
:G      RTR     X,B
        AAD     X,B             SHIFT LEFT ONE BIT
        RTR     Y,B
        AAD     Y,B
        RLK     X,X
        RTR     A,B
        AAD     A,B
        RLK     Y,Y
        RTR     Z,B
        AAD     Z,B
        RLK     A,A
        LDB     :X+3
        AAD     Z,B
        JMP     :E
:H      LDB     :X+4            WHICH PASS?
        SKZ     2
```

```
        AAD     Z,B         ADD QUOTIENT BIT
        LDB     :Z+1        SHOULD RESULT BE NEGATIVE?
        SKM     2
        JMP     :L
        AXR     Y,377       NEGATE RESULTS
        AXR     A,B
        AXR     Z,B
        RIC     Z,Z
        RLK     A,A
        RLK     Y,Y
:L      RTR     Y,B
        STB     PA+3        STORE RESULTS -
        RTR     A,B
        STB     PA+4
        RTR     Z,B
        STB     PA+5
        ELB     :X
        RTR     B,X
        ELB     :Z          RETURN
        RIC     B,P
*
*****************************************************************************
```

```
        JMP     :J
        RTR     Z,B         1ST PASS  STORE PARTIAL RESULTS
        STB     PA+3
        LDB     :V+4
        RTR     B,Z
        RDC     B,B
:I      STB     :X+4        SET PASS 2 SWITCH
        ALD     B,37Ø       RESET COUNTER
        STB     :X+2
        JMP     :G
:J      RTR     B,B         2ND PASS ?
        SKM     2
        JMP     :K          NO
        RTR     Z,B         STORE NEXT PARTIAL VALUE
        STB     PA+4
        LDB     :V+5
        RTR     B,Z
        ALD     B,ØØ1
        JMP     :I
:K      RTR     X,B         3RD PASS
        STB     PA          STORE REMAINDER
        RTR     Y,B
        STB     PA+1
        RTR     A,B
        STB     PA+2
        LDB     PA+4        PUT QUOTIENT INTO Y,A, AND Z
        RTR     B,A
        LDB     PA+3
        RTR     B,Y
        AAD     Y,B         SHIFT LEFT ONE BIT POSITION
        RTR     A,B
        AAD     A,B
        RLK     Y,Y
        RTR     Z,B
        AAD     Z,B
        RLK     A,A
        LDB     :X+3
```

APPENDIX A
PGS BOOTSTRAP LOADER

```
        *
        *       BUS II BOOTSTRAP
        *
        *       88YØØØ14A  REV A  1-11-69  SHEET 1 NEXT 1
        *
        *       OPERATING SEQUENCE
        *
        *         1.  ENTER BOOTSTRAP INTO MEMORY FROM THE
        *             CONSOLE SWITCHES AT LOCATION 1ØØØ.
        *         2.  SWITCH TTY ASR TO ON-LINE.
        *         3.  SET I REGISTER = 21ØØ.
        *         4.  SET P REGISTER = Ø777.
        *         5.  R/I SWITCH TO RUN AND  PRESS STEP.
        *         6.  PLACE SPC-12 BOOT FORMAT LOADER PAPER
        *             TAPE INTO THE TTY READER WITH THE FIRST
        *             CHARACTER OVER THE READ SENSORS.
        *         7.  SWITCH TTY READER TO START.
        *         8.  BOOTSTRAP LOADS ONLY THE LOADER AND WILL
        *             AUTOMATICALLY PASS CONTROL TO THE
        *             LOADER TO CONTINUE THE LOADING PROCESS.
        *         9.  WHEN READER RUNS OUT OF TAPE IT WILL STOP,
        *             ADDITIONAL TAPES MAY BE LOADED OR CONTROL
        *             MAY BE TRANSFERRED AS DESIRED THRU MANUAL
        *             PROCEDURES USING THE SPC-12 PROGRAMMER'S
        *             CONSOLE OR THRU THE POWER FAIL/AUTO
        *             RESTART FEATURE (AN OPTION).
                ZO      1ØØØ
                ZN
Ø1ØØØ B1        ALD     X,12Ø       SET X = INITIAL ADDRESS
Ø1ØØ3 B2        SHI     B           WAIT FOR SYNC BIT
Ø1ØØ5           SKZ     2
Ø1ØØ6           JMP     B2
Ø1Ø1Ø           RIC     P,YY        SET BIT COUNTER
Ø1Ø12           ALD     Z,3ØØ       FORCE 5.53 MS DELAY
Ø1Ø15 B4        RTR     P,ZZ        SET 8.85 MS DELAY
Ø1Ø17           SHI     A           INPUT BIT
Ø1Ø21 B5        RIC     Z,Z         DELAY LOOP
Ø1Ø23           RTR     ZZ,ZZ
Ø1Ø25           SKF     Ø
Ø1Ø26           SKZ     2
Ø1Ø27           JMP     B5
Ø1Ø31           RDC     YY,YY       COUNT BIT
Ø1Ø33           SKZ     2           IS CHARACTER IN?
Ø1Ø34           JMP     B4          NO
Ø1Ø36           GØS     A,X         YES,STORE CHARACTER
```

```
Ø1Ø4Ø         RIC    X,X          INCREMENT ADDRESS INDEX
Ø1Ø42         ASU    A,117
Ø1Ø45         SKZ    2            IS LOADER IN?
Ø1Ø46         JMP    B2           NO-GET NEXT CHARACTER
Ø1Ø5Ø         JMP    L1           YES-JUMP TO LOADER
        L1    ZL     264
              *
              ZD

Ø1ØØØ  ØØ4 111 12Ø Ø14 Ø45 131 142 ØØ3
Ø1Ø1Ø  Ø1Ø 164 ØØ4 113 3ØØ Ø1Ø 374 Ø14
Ø1Ø2Ø  Ø4Ø Ø1Ø 133 Ø1Ø 377 1ØØ 131 142
Ø1Ø3Ø  Ø21 Ø1Ø 266 131 142 Ø15 ØØØ 13Ø
Ø1Ø4Ø  Ø1Ø 111 ØØ4 21Ø 117 131 142 ØØ3
Ø1Ø5Ø  14Ø 264
```

APPENDIX B
SERIAL TELETYPE I/O, ASR 33, 35

```
                *
                *     SERIAL TTY I/O - ASR-33,-35
                *
                *     88YØØØØØA REV A  1-2-69  SHEET 1 NEXT 1
                *     ***************************************
                *     THE COMMANDS OF THIS ROUTINE
                *     MUST NOT BE ASSEMBLED SHARED
                *     THIS ROUTINE MAY BE ASSEMBLED
                *     INTO ANY MEMORY MODE
                *     ***************************************
                *     CALLING SEQUENCES:
                *       INPUT ONLY - R  IP,B
                *                    J  TI
                *       INPUT/ECHO - R  IP,B
                *                    J  TE
                *       REG'S @ ENTRY - A=NOT DEF
                *                       X=NOT DEF
                *                       Y=NOT DEF
                *                       Z=NOT DEF
                *       REG'S @ EXIT  - A=INPUT CHAR
                *                       X=NOT DISTRUBED
                *                       Y=NOT DISTRUBED
                *                       Z=NOT DISTRUBED
                *       OUTPUT      - R  IP,B
                *                    J  TO
                *       REG'S @ ENTRY - A=OUTPUT CHAR
                *                       X=NOT DEF
                *                       Y=NOT DEF
                *                       Z=NOT DEF
                *       REG'S @ EXIT  - A=377
                *                       X=NOT DISTRUBED
                *                       Y=NOT DISTRUBED
                *                       Z=NOT DISTRUBED
                *
                *
ØØ12Ø TO    STB   TB          TTY OUTPUT
ØØ122       TBB
ØØ124       STB   TB+1
ØØ126       SHRO  B           OUTPUT SPACE
ØØ13Ø       LDB   TI+1
ØØ132       STB   T4+4        SET'OR'CMD
ØØ134       ALD   B,12Ø
ØØ137       STB   T4+1        SET SERIAL CMD
ØØ141       LDB   TA
ØØ143 T1    STB   TC          SET BIT COUNT
```

```
ØØ145 T2    LDB   T3          9MS DELAY
ØØ147 T3    RIC   B,B         DELAY LOOP
ØØ151        STB   TD
ØØ153        STB   TD
ØØ155        STB   TD
ØØ157        STB   TD
ØØ161        SKZ   2
ØØ162        JMP   T3
ØØ164        LDB   TC          COUNT BIT
ØØ166        RIC   B,B
ØØ17Ø        STB   TC
ØØ172        SKR   4
ØØ173        SHLO  B           OUTPUT MARK
ØØ175        JMP   T2
ØØ177        SKM   4
ØØ2ØØ        ELB   TB
ØØ2Ø2        RIC   B,P         EXIT
ØØ2Ø4        SHCO  A           SERIAL IN/OUT/ECHO
ØØ2Ø6        AOR   A,2ØØ       OUT=2ØØ,IN=ØØØ
ØØ211        JMP   T2
      *
      *
ØØ213 TI    PLR                TTY INPUT
ØØ215        SKR   2
ØØ216 TE    PLS                TTY INPUT/ECHO
ØØ22Ø        STB   TB
ØØ222        TBB
ØØ224        STB   TB+1
ØØ226        LDB   TF
ØØ23Ø        SKS   2
ØØ231        LDB   T5+1
ØØ233        STB   T4+1         SET SERIAL CMD
ØØ235        TBB
ØØ237        STB   T4+4         SET'OR'CMD
ØØ241        LDB   TA
ØØ243        STB   TC           SET BIT COUNT
ØØ245 T5    SHI   A            LOOK FOR SPACE
ØØ247        SKZ   2
ØØ25Ø        JMP   T5
ØØ252        LDB   TI+1         4.5MS DELAY
ØØ254        JMP   T3+2
ØØ256 TA    ZC    366          BIT COUNT
ØØ257 TB    ZR    2            RTRN ADDR STO
ØØ261 TC    ZC    Ø            BIT COUNTER
ØØ262 TD    ZC    Ø            DELAY LOOP STO
ØØ263 TF    ZC    14Ø          INPUT/ECHO CCB
```

```
:
      ZD

ØØ12Ø  3ØØ 257 Ø14 2Ø5 3ØØ 26Ø Ø14 1Ø5
ØØ13Ø  34Ø 214 3ØØ 21Ø ØØ4 115 12Ø 3ØØ
ØØ14Ø  2Ø5 34Ø 256 3ØØ 261 34Ø 147 Ø1Ø
ØØ15Ø  155 3ØØ 262 3ØØ 262 3ØØ 262 3ØØ
ØØ16Ø  262 131 14Ø 147 34Ø 261 Ø1Ø 155
ØØ17Ø  3ØØ 261 1Ø6 Ø14 115 14Ø 145 116

ØØ2ØØ  16Ø 257 Ø1Ø 145 Ø14 12Ø ØØ4 Ø7Ø
ØØ21Ø  2ØØ 14Ø 145 Ø14 2ØØ 1Ø5 Ø14 2Ø1
ØØ22Ø  3ØØ 257 Ø14 2Ø5 3ØØ 26Ø 34Ø 263
ØØ23Ø  125 34Ø 246 3ØØ 2Ø5 Ø14 2Ø5 3ØØ
ØØ24Ø  21Ø 34Ø 256 3ØØ 261 Ø14 Ø4Ø 131
ØØ25Ø  14Ø 245 34Ø 214 14Ø 151 366 155
ØØ26Ø  15Ø ØØØ ØØØ 14Ø
```

APPENDIX C
PGS PAPER TAPE LOADER/PUNCH

```
        *
        *       PGS PAPER TAPE LOADER/PUNCHER
        *
        *       88YØØØØ1A REV A Ø1-1Ø-69  SHEET
        *
        *
        *       FORMATTED PAPER TAPE LOADER
        *
        *
        *       1.  LOAD BOOT FORMAT LOADER WITH SPC-12 SERIAL
        *           TTY BOOTSTRAP LOADER.
        *       2.  INITIATE LOADER AT LOCATION L1.
        *       3.  PLACE FORMATTED PAPER TAPE INTO ASR READER
        *           AND SWITCH READER TO START.
        *       4.  EACH TAPE RECORD CONTAINS A START ADDRESS,
        *           MEMORY MODE, AND 1 OR MORE DATA BLOCKS OF
        *           255 CHARACTERS AND A CHECKSUM.
        *       5.  AS EACH DATA BLOCK IS COMPLETED, THE LOADER
        *           WILL OUTPUT 7 RUBOUTS OR IF A CHECKSUM ERROR
        *           OCCURS ERR IS OUTPUT.
        *       6.  TO LOAD A TAPE AT AN ADDRESS OTHER THAN THE
        *           ADDRESS SPECIFIED ON THE TAPE, PERFORM THE
        *           FOLLOWING AT STEP 2.
        *
        *                SET X = MEMORY MODE.
        *                SET Y = START ADDRESS.
        *                INITIATE LOADER AT LOCATION L2.
        *
        *
        *
        *       SPC-12 CONSOLE ENTRIES
        *
                ZN
ØØ264 L1        TBB
ØØ266           TBB
ØØ27Ø           SKZ     2
ØØ271 L2        RDC     B,B
ØØ273           RDC     P,Z             RETURN ADDRESS
        *
        *       PROGRAM ENTRY
        *
ØØ275 LD        STB     LC              SET ORIGIN FLAG
ØØ277           RIC     P,B
ØØ3Ø1           JMP     TE              READ A CHARACTER
ØØ3Ø3           RTR     A,B
```

```
ØØ3Ø5          SKZ   2          CHARACTER = BLANK?
ØØ3Ø6          JMP   LD+2       NO
      *
ØØ31Ø          RIC   P,B
ØØ312          JMP   TE         READ A CHARACTER
ØØ314          RIC   A,B
ØØ316          SKZ   2          CHARACTER = RUBOUT?
ØØ317          JMP   LD+6       NO
      *
ØØ321          RIC   P,B        READ MEMORY MODE
ØØ323          JMP   TI
ØØ325          LDB   LC
ØØ327          SKN   2          MODE SPECIFIED IN X?
ØØ33Ø          RTR   A,X        NO
ØØ332          RTR   X,B
ØØ334          TBE              SET MODE
ØØ336          RIC   P,B        READ START ADDRESS
ØØ34Ø          JMP   TI
ØØ342          RTR   A,X
ØØ344          RIC   P,B
ØØ346          JMP   TI
ØØ35Ø          LDB   LC
ØØ352          SKN   4          ADDRESS SPECIFIED IN Y?
ØØ353          RTR   X,Y        NO - START ADDRESS TO Y
ØØ355          RTR   A,YY
ØØ357          RTR   Y,X
      *
      *        READ A BLOCK
      *
ØØ361 L3       RIC   P,B        READ BLOCK LENGTH
ØØ363          JMP   TI
ØØ365          RTR   A,B        INITIALIZE CHECKSUM
ØØ367          RTR   B,Y
ØØ371          SKN   2          BLOCK LENGTH = Ø?
ØØ372          RIC   Z,P        YES - EXIT
      *
      *        READ A CHARACTER
      *
ØØ374 L4       STB   LC         RESTORE CHECKSUM
ØØ376          RIC   P,B        READ A CHARACTER
ØØ4ØØ          JMP   TI
ØØ4Ø2          RTR   A,B        SAVE CHARACTER
ØØ4Ø4          STB   LS
ØØ4Ø6          LDB   LC         FORM NEW CHECKSUM
ØØ41Ø          AXR   A,B
ØØ412          SHC   A
ØØ414          RTR   Y,Y
ØØ416          SKZ   6          BLOCK LENGTH = Ø?
ØØ417          LDB   LS         NO - STORE CHARACTER
ØØ421          STB   Ø,X
```

```
ØØ423          RIC   X,X         INCREMENT LOCATION
ØØ425          RTR   A,B
ØØ427          RDC   Y,Y         DECREMENT BLOCK LENGTH
ØØ431          SKR   2           ALL DONE?
ØØ423          JMP   L4          NO
        *
        *      SEND ASSURANCE MESSAGE
        *
ØØ434 L5       STB   LC          RESTORE CHECKSUM
ØØ436          ELB   LM
ØØ44Ø          RTR   B,Y
        *
ØØ442 L6       GØL   A,Y         SEND 1 CHARACTER
ØØ444          SKN   2
ØØ445          JMP   L3
ØØ447          LDB   LC
ØØ451          SKN   2           CHECKSUM GOOD?
ØØ452          RDC   B,A         YES
ØØ454          RIC   P,B
ØØ456          JMP   TO
ØØ46Ø          RIC   Y,Y
ØØ462          JMP   L6
        *
ØØ464 LM       ZA    LE          ERROR MSG ADDRESS
ØØ466 LE       ZC    377         RUBOUT
ØØ467          ZC    215         CARRIAGE RETURN
ØØ47Ø          ZC    212         LINE FEED
ØØ471          ZC    3Ø5         E
ØØ472          ZC    322         R
ØØ473          ZC    322         R
ØØ474          ZC    Ø           EOM
        *
ØØ475 LC       ZC    377         CHECKSUM
ØØ476 LS       ZC    117         TEMP STORAGE
        *
               ZD

ØØ264  Ø14 2Ø5 Ø14 2Ø5
ØØ27Ø  131 Ø1Ø 255 Ø1Ø 234 3Ø1 Ø75 Ø1Ø
ØØ3ØØ  154 14Ø 216 Ø1Ø 35Ø 131 14Ø 277
ØØ31Ø  Ø1Ø 154 14Ø 216 Ø1Ø 15Ø 131 14Ø
ØØ32Ø  3Ø3 Ø1Ø 154 14Ø 213 341 Ø75 111
ØØ33Ø  Ø1Ø 31Ø Ø1Ø 351 Ø14 2Ø6 Ø1Ø 154
ØØ34Ø  14Ø 213 Ø1Ø 31Ø Ø1Ø 154 14Ø 213
ØØ35Ø  341 Ø75 112 Ø1Ø 321 Ø1Ø 369 Ø1Ø
ØØ36Ø  312 Ø1Ø 154 14Ø 213 Ø1Ø 35Ø Ø1Ø
ØØ37Ø  325 111 Ø1Ø 143 3Ø1 Ø75 Ø1Ø 154

ØØ4ØØ  14Ø 213 Ø1Ø 35Ø 3Ø1 Ø76 341 Ø75
ØØ41Ø  ØØ4 Ø4Ø Ø14 Ø2Ø Ø1Ø 322 133 341
ØØ42Ø  Ø76 32Ø ØØØ Ø1Ø 111 Ø1Ø 35Ø Ø1Ø
```

```
ØØ43Ø  222 1Ø5 14Ø 374 3Ø1 Ø75 161 Ø64
ØØ44Ø  Ø1Ø 325 ØØØ 2ØØ 111 14Ø 361 341
ØØ45Ø  Ø75 111 Ø1Ø 2Ø5 Ø1Ø 154 14Ø 12Ø
ØØ46Ø  Ø1Ø 122 141 Ø42 Ø66 ØØ1 377 215
ØØ47Ø  212 3Ø5 322 322 ØØØ 377 117
       *
       *********PGS PAPER TAPE LOADER/PUNCHER
       *       88YØØØØ1A REV A 1-1Ø-69  SHEET 2 NEXT 2
       *
       *         NO ENTRY POINT IN THIS PROGRAM WILL GENERATE
       *         A LEADER
       *
       ****    BOOT FORMAT PAPER TAPE PUNCHER
       *
       ***     BUS II ENTRY
       *
ØØ477 PK      ELB   PE          LOAD B WITH RETURN TO BUS
       *
       ***     PROGRAM ENTRY
       *         1.  SET X REGISTER = END ADDRESS (476)
       *         2.  SET Y REGISTER = START ADDRESS (12Ø)
       *         3.  SET Z REGISTER = BLOCK LENGTH (357)
       *         4.  SET B REGISTER = RETURN ADDRESS
       *         5.  JUMP TO PB
       *
       ***     CONSOLE ENTRY
       *         1.  SET X REGISTER = END ADDRESS (476)
       *         2.  SET Y REGISTER = START ADDRESS (12Ø)
       *         3.  SET Z REGISTER = BLOCK LENGTH (357)
       *         4.  SET B REGISTER = HALT ADDRESS (616)
       *         5.  TTY PAPER TAPE PUNCH ON
       *         6.  GO TO PB
       *
ØØ5Ø1 PM      STB    PS         STORE RETURN ADDRESS
ØØ5Ø3         TBB
ØØ5Ø5         STB    PS+1
ØØ5Ø7         JMP    P3         JUMP TO PUNCH
       *
       ****    LOADER FORMAT PAPER TAPE PUNCHER
       *
       ***     CONSOLE ENTRY
       *         1.  SET X REGISTER = MEMORY MODE
       *         2.  SET Y REGISTER = START ADDRESS
       *         3.  SET Z REGISTER = END ADDRESS
       *         4.  TTY PAPER TAPE PUNCH ON
       *         5.  GO TO P1
       *
ØØ511 P1      ELB   PA          EXIT HALT ADDRESS
```

```
      **    PROGRAM ENTRY
      *       1.  SET X REGISTER = MEMORY MODE
      *       2.  SET Y REGISTER = START ADDRESS
      *       3.  SET Z REGISTER = END ADDRESS
      *       4.  SET B REGISTER = RETURN ADDRESS
      *       5.  JUMP TO PP
      *
00513 PP    STB    PS         SAVE RETURN ADDRESS
00515       TBB
00517       STB    PS+1
00521       ALD    A,314      PUNCH AN L
00524       RIC    P,B
00526       JMP    TO
00530       AZE    A          PUNCH A BLANK
00532       RIC    P,B
00534       JMP    TO
00536       RIC    P,B        PUNCH A RUBOUT
00540       JMP    TO
00542       RTR    X,A        PUNCH MODE
00544       RIC    P,B
00546       JMP    TO
00550       RTR    X,B        SET MODE
00552       TBE
00554       RTR    Z,X        SAVE END ADDRESS
00556       RTR    Y,A        PUNCH START ADDRESS
00560       RIC    P,B
00562       JMP    TO
00564       RTR    YY,A
00566       RIC    P,B
00570       JMP    TO
      *
      *     PUNCH BLOCK LENGTH
      *
00572 P2    RIC    X,Z
00574       RTR    Y,B
00576       ASU    Z,B        Z = END ADDR + 1 - START ADDR
00600       RTR    ZZ,ZZ
00602       SKZ    2          DIFF   377?
00603       RDC    B,Z        YES
00605       RTR    Z,A        NO
00607       RIC    P,B
00611       JMP    TO         PUNCH BLOCK LENGTH
00613       RTR    Z,B
00615       SKN    4          IF BLOCK LENGTH=ZERO,EXIT.
      *
      *     EXIT
      *
00616 PX    ELB    PS         RETURN ADDRESS
00620       RIC    B,P
```

```
      ;
      ;     PUNCH A BLOCK
      ;
ØØ622 P3    STB   PC        INITIALIZE CHECKSUM
ØØ624       GØL   A,Y       PUNCH A CHARACTER
ØØ626       RIC   P,B
ØØ63Ø       JMP   TO
ØØ632       GØL   A,Y       FORM CHECKSUM
ØØ634       LDB   PC
ØØ636       AXR   A,B
ØØ64Ø       SHC   A
ØØ642       RTR   A,B
ØØ644       RIC   Y,Y       INCREMENT ADDRESS
ØØ646       RDC   Z,Z       DECREMENT BLOCK LENGTH
ØØ65Ø       SKZ   2         BLOCK LENGTH = Ø?
ØØ651       JMP   P3        NO
ØØ653       RIC   P,B       PUNCH CHECKSUM
ØØ655       JMP   TO
      ;
      ;
      ;     PUNCH ASSURANCE MESSAGE
      ;
ØØ657       ALD   Z,ØØ6
ØØ662 P4    RIC   P,B       PUNCH A RUBOUT
ØØ664       JMP   TO
ØØ666       RDC   Z,Z
ØØ67Ø       SKZ   2         ALL DONE?
ØØ671       JMP   P4        NO
ØØ673       JMP   P2        PUNCH NEXT BLOCK
      ;
ØØ675 PS    ZA    Ø         RETURN ADDRESS
ØØ677 PA    ZA    PX        EXIT HALT
ØØ7Ø1 PE    ZA    BB-2      BUS II RETURN
ØØ7Ø3 PC    ZC    377       CHECKSUM
      ;
            ZD
ØØ477  161

ØØ5ØØ  3Ø1 3Ø1 275 Ø14 2Ø5 3Ø1 276 141
ØØ51Ø  222 161 277 3Ø1 275 Ø14 2Ø5 3Ø1
ØØ52Ø  276 ØØ4 11Ø 314 Ø1Ø 154 14Ø 12Ø
ØØ53Ø  ØØ4 ØØØ Ø1Ø 154 14Ø 12Ø Ø1Ø 154
ØØ54Ø  14Ø 12Ø Ø1Ø 3Ø1 Ø1Ø 154 14Ø 12Ø
ØØ55Ø  Ø1Ø 351 Ø14 2Ø6 Ø1Ø 313 Ø1Ø 3Ø2
ØØ56Ø  Ø1Ø 154 14Ø 12Ø Ø1Ø 3Ø6 Ø1Ø 154
ØØ57Ø  14Ø 12Ø Ø1Ø 131 Ø1Ø 352 ØØ4 2Ø3

ØØ6ØØ  Ø1Ø 377 131 Ø1Ø 235 Ø1Ø 3Ø3 Ø1Ø
ØØ61Ø  154 14Ø 12Ø Ø1Ø 353 112 161 275
ØØ62Ø  Ø1Ø 145 3Ø1 3Ø3 ØØØ 2ØØ Ø1Ø 154
```

```
ØØ63Ø  14Ø 12Ø ØØØ 2ØØ 341 3Ø3 ØØ4 Ø4Ø
ØØ64Ø  Ø14 Ø2Ø Ø1Ø 35Ø Ø1Ø 122 Ø1Ø 233
ØØ65Ø  131 141 222 Ø1Ø 154 14Ø 12Ø ØØ4
ØØ66Ø  113 ØØ6 Ø1Ø 154 14Ø 12Ø Ø1Ø 233
ØØ67Ø  131 141 262 141 172 ØØØ ØØØ 216

ØØ7ØØ  ØØ1 1Ø3 Ø15 377
```

APPENDIX D
PGS BASIC UTILITY SYSTEM II (BUS II)

```
        *
        *    PGS BASIC UTILITY SYSTEM II (BUS-II)
        *    88Y0000-2A  REV A  1-15-69  SHEET 1 NEXT 2
        *
        *    BUS-II PROVIDES THE FUNCTIONS FOR TESTING
        *    AND UPGRADING PROGRAMS ON THE SPC-12.
        *
        *    BUS-II IS RELOCATABLE TO ANY MODE EXCEPT 0.
        *
        *    BUS-II USES THE GAI RECOMMENDED SHARED COMMAND
        *    BYTES.
        *
        *    BUS-II REQUIRES THE FOLLOWING PROGRAMS AND/OR
        *    DATA TO BE RESIDENT IN MODE 0:
        *
        *       GAI RECOMMENDED SHARED BYTES
        *       SERIAL TTY I/O SUBROUTINE
        *       PGS LOADER/PUNCHER
        *        BUS-II MODE CONTROL ROUTINES
        *
        *    BUS-II FUNCTION CONTROL PROGRAM
        *
16500       ZC    100
16501 BA    RIC   P,Z          SEND CARRET LINE-FEED
16503       JMP   CR
        *
16505#BB    RIC   P,B          ACCEPT INPUT
16506       JMP   OI
16510       RTR   Z,Y          SAVE INPUT
16512#      AZE   X
        *
16513 BC    LDB   BT,X         SEARCH FUNCTION CNTRL TABLE
16515       SKN   2            SEARCH COMPLETE?
16516       JMP   B?           YES
16520#      RTR   B,A
16521       LDB   OC           GET INPUT CNTRL CHARACTER
16523#      ASU   A,B
16524       SKZ   4            INPUT CHAR = TABLE CHAR?
            ZY
16525#      AAD   X,003        NO - GET NEXT ENTRY
16527       JMP   BC
        *
16531       LDB   BT+1,X       GET BUS-II FUNCTION ADDRESS
16533#      RTR   B,Z
16534       LDB   BT+2,X
```

```
16536       RTR   B,ZZ
16540       LDB   OM          GET MODE
16542#      RTR   B,X
16543       SKN   2
16544       RIC   X,X
16546       RDC   Z,P         EXIT
      ※
      ※     SEND ?
      ※
1655Ø#B     ALD   A,277       ?
16552#      RIC   P,B
16553       JMP   TO
16555       JMP   BA

            BUS-II FUNCTION CONTROL TABLE

16557 BT    ZC    215         CARRET
16560       ZA    BA          FUNCTION CONTROL
16562       ZC    240         SPACE
16563       ZA    CA          DISPLAY/ALTER LOCATION
16565       ZC    322         R
16566       ZA    R1          DISPLAY/ALTER REGISTER
16570       ZC    307         G
16571       ZA    GA          GO
16573       ZC    324         T
16574       ZA    H1          TRAP
16576       ZC    314         L
16577       ZA    LA          LOADER CALLER
166Ø1       ZC    257         /
166Ø2       ZA    A1          DUMP/PUNCHER CALLER
166Ø4 P     ZC    302         B
166Ø5       ZA    BP          BOOT LOADER PUNCH
166Ø7       ZC    Ø
      ※
            ZD

165ØØ  1ØØ Ø1Ø 134 157 147 Ø5Ø 157 253
1651Ø  Ø1Ø 323 Ø31 375 157 111 155 15Ø
1652Ø  Ø41 357 357 Ø24 132 Ø33 ØØ3 155
1653Ø  113 375 16Ø Ø47 375 161 Ø1Ø 375
1654Ø  357 361 Ø43 111 Ø1Ø 111 Ø1Ø 243
1655Ø  Ø21 277 Ø5Ø 14Ø 12Ø 155 1Ø1 215
1656Ø  1Ø1 Ø15 24Ø 21Ø Ø15 322 257 Ø15
1657Ø  307 241 Ø16 324 373 Ø15 314 312

166ØØ  Ø16 257 323 Ø16 3Ø2 362 Ø17 ØØØ
      ※     PGS BASIC UTILITY SYSTEM II (BUS-II)
      ※     88YØØØØ2A  REV A  1-15-69  SHEET 2 NEXT 3
      ※
      ※     BUS-II DISPLAY/CHANGE MEMORY ROUTINE
```

```
      ;
      ;     THIS ROUTINE TYPES OUT AND/OR CHANGES
      ;     THE CONTENTS OF THE LOCATION SPECIFIED
      ;     IN Y AND THE MODE SPECIFIED IN X.
      ;
      ;
16610#CA    RIC   P,B         PRINT ADDRESS
16611       JMP   AD
      ;
16613#      RIC   P,B         GET MEMORY CONTENTS
16614       JMP   ML
16616       RTR   A,Z
16620#      RIC   P,B         PRINT CONTENTS
16621       JMP   03
16623#      ALD   A,240       SEND SPACE
16625#      RIC   P,B
16626       JMP   TO
      ;
16630#      RIC   P,B         ACCEPT INPUT
16631       JMP   OI
16633       LDB   OF          DATA IN FLAG
16635       SKZ   6           ANY DATA IN?
16636       RTR   Z,A         YES - CHANGE LOCATION
16640       RIC   P,B
16642       JMP   MS
      ;
16644       LDB   OC          GET CONTROL CHARACTER
16646#      RTR   B,A
16647#      ASU   A,215
16651       SKN   2           CHARACTER - CARRET?
16652       JMP   BA          YES - EXIT
16654#      RIC   Y,Y         INCREMENT TO NEXT LOCATION
16655       JMP   CA
      ;
            ZD

16610  050 157 103 050 141 307 010 330
16620  050 157 161 021 240 050 140 120
16630  050 157 253 357 360 133 010 303
16640  010 154 141 305 357 357 041 025
16650  215 111 155 101 053 155 210
      ;
      ;     PGS BASIC UTILITY SYSTEM II (BUS-II)
      ;     88Y00002A  REV A  1-15-69  SHEET 3 NEXT 4
      ;
      ;     BUS-II DISPLAY/CHANGE REGISTER ROUTINE
      ;
      ;     THIS ROUTINE TYPES OUT AND/OR CHANGES
      ;     THE CONTENTS OF THE PSUEDO-REGISTERS
      ;     A, B, X, Y, AND Z.
```

```
      ※
      ※
16657 R1    RIC   P,Z         SEND CARRET LINE FEED
16661       JMP   CR
16663#      ALD   X,Ø14
      ※
16665 R2    LDB   RZ,X        PRINT REGISTER MNEMONIC
16667#      RTR   B,A
16670#      RIC   P,B
16671       JMP   TO
16673#      ALD   A,24Ø       SEND SPACE
16675#      RIC   P,B
16676       JMP   TO
      ※
167ØØ       LDB   RZ+1,X      PRINT REGISTER CONTENTS
167Ø2#      RTR   B,Z
167Ø3       LDB   RZ+2,X
167Ø5       RTR   B,ZZ
167Ø7#      RIC   P,B
1671Ø       JMP   O4
16712#      ALD   A,24Ø       SEND SPACE
16714#      RIC   P,B
16715       JMP   TO
      ※
16717#      RIC   P,B         ACCEPT INPUT
1672Ø       JMP   O1
16722       LDB   OF          GET DATA IN FLAG
16724       SKN   2           ANY DATA IN?
16725       JMP   R4          NO
16727#      RTR   Z,B         CHANGE REGISTER
1673Ø       STB   RZ+1,X
16732#      TBB
16733       STB   RZ+2,X
      ※
16735 R4    LDB   OC          GET CONTROL CHARACTER
16737#      RTR   B,A
16740#      ASU   A,215
16742       SKN   2           CHARACTER = CARRET?
16743       JMP   BA          YES - EXIT
16745#      ASU   X,ØØ3       GET NEXT REGISTER
16747       SKM   2           ALL DONE?
1675Ø       JMP   R2          NO
16752       JMP   BA          YES - EXIT
      ※
      ※     BUS-II PSUEDO-REGISTER TABLE
      ※
16754 RZ    ZC    332
16755       ZA    Ø           PSUEDO-Z
16757 RY    ZC    331
1676Ø       ZA    Ø           PSUEDO-Y
```

```
16762 RX    ZC    33Ø
16763       ZA    Ø          PSUEDO-X
16765 RB    ZC    3Ø2
16766       ZA    Ø          PSUEDO-B
1677Ø RA    ZC    3Ø1
16771       ZA    Ø          PSUEDO-A
      *
            ZD

16657  Ø1Ø
1666Ø  134 157 147 Ø32 Ø14 375 354 Ø41
1667Ø  Ø5Ø 14Ø 12Ø Ø21 24Ø Ø5Ø 14Ø 12Ø

167ØØ  375 355 Ø47 375 356 Ø1Ø 375 Ø5Ø
1671Ø  157 163 Ø21 24Ø Ø5Ø 14Ø 12Ø Ø5Ø
1672Ø  157 253 357 36Ø 111 155 335 Ø46
1673Ø  335 355 Ø61 335 356 357 357 Ø41
1674Ø  Ø25 215 111 155 1Ø1 Ø34 ØØ3 115
1675Ø  155 265 155 1Ø1 332 ØØØ ØØØ 331
1676Ø  ØØØ ØØØ 33Ø ØØØ ØØØ 3Ø2 ØØØ ØØØ
1677Ø  3Ø1 ØØØ ØØØ
      *
      *     PGS BASIC UTILITY SYSTEM II (BUS-II)
      *     88YØØØØ2A  REV A  1-15-69  SHEET 4 NEXT 5
      *
      *     BUS-II TRAP ROUTINE
      *
      *     THIS ROUTINE INSERTS UP TO TWO TRAPS (BREAK-POINT)
      *     AT THE ADDRESS SPECIFIED IN Y AND THE MODE SPEC-
      *     IFIED IN X.
      *     AFTER A TRAP IS ENCOUNTERED, TRAP RETURN SAVES THE
      *     CONTENTS OF REGISTERS A, B, X, Y, AND Z IN PSUEDO-
      *     REGISTERS A, B, X, Y, AND Z, PRINTS THE ADDRESS OF
      *     THE TRAP, AND TRANSFERS CONTROL TO THE DISPLAY
      *     REGISTER ROUTINE.
      *
      *
16773 H1    ELB   HB         TRAP TABLE BASE ADDRESS
16775#      RTR   B,Z
16776#      TBB
16777#      TBB
17ØØØ       STB   HC         RESET SWITCH
      *
17ØØ2 H2    LDB   OF         GET DATA IN FLAG
17ØØ4       SKN   2          ANY DATA IN?
17ØØ5       JMP   H3         NO - REMOVE TRAP
17ØØ7#      G4L   A,Z        GET TRAP FLAG
17Ø1Ø       SKZ   2          IS TRAP SET?
17Ø11       JMP   HX         YES
      *
```

```
      *     SET TRAP
      *
17Ø13       GØS   Y,Z         SAVE TRAP ADDRESS
17Ø15#      RIC   P,B         SAVE DATA FROM TRAP LOCATION
17Ø16       JMP   ML
17Ø2Ø#      G2S   A,Z
17Ø21#      G6L   A,Z         INSERT TRAP BYTE 1
17Ø22#      RIC   P,B
17Ø23       JMP   MS
17Ø25       RDC   B,A         SET TRAP FLAG
17Ø27#      G4S   A,Z
17Ø3Ø       RIC   Z,Z
17Ø32       GØS   YY,Z        SAVE TRAP ADDRESS
17Ø34#      RIC   Y,Y
17Ø35#      RIC   P,B         SAVE DATA FROM TRAP LOCATION + 1
17Ø36       JMP   ML
17Ø4Ø#      G2S   A,Z
17Ø41#      G6L   A,Z         INSERT TRAP BYTE 2
17Ø42#      RIC   P,B
17Ø43       JMP   MS
17Ø45       G4S   X,Z         SAVE TRAP MODE
17Ø47       JMP   BA          EXIT
      *
17Ø51#H3    G4L   A,Z         GET TRAP FLAG
17Ø52       SKN   2           IS TRAP SET?
17Ø53       JMP   HX          NO
      *
      *     RESET TRAP AND RESTORE MEMORY
      *
17Ø55 H4    GØL   Y,Z         GET MEMORY ADDRESS
17Ø57       RIC   Z,Z
17Ø61       GØL   YY,Z
17Ø63       G4L   X,Z         GET MEMORY MODE
17Ø65       RDC   Z,Z
17Ø67#      G2L   A,Z         GET MEMORY DATA
17Ø7Ø#      RIC   P,B
17Ø71       JMP   MS          RESTORE DATA
17Ø73#      AZE   A           RESET TRAP FLAG
17Ø74#      G4S   A,Z
17Ø75#      RIC   Y,Y
17Ø76       RIC   Z,Z
171ØØ#      G2L   A,Z         GET MEMORY DATA
171Ø1#      RIC   P,B
171Ø2       JMP   MS          RESTORE DATA
171Ø4       RDC   Z,Z
171Ø6#      RDC   Y,Y
171Ø7#      RIC   P,B         PRINT TRAP ADDRESS
1711Ø       JMP   AD
17112       LDB   HC          GET TRAP SWITCH
17114       SKN   2           SWITCH ON?
```

```
17115        JMP   BA          NO - EXIT TO BUS-II
17117#       TBB
1712Ø        STB   HC          RESET TRAP SWITCH
17122        JMP   R1          EXIT TO REGISTER DISPLAY
             *
             *     GET NEXT TRAP
             *
17124 HX     RTR   Z,A
17126        ELB   HB          TRAP TABLE BASE ADDRESS
1713Ø#       ASU   A,B
17131        SKZ   2           ALL TRAPS EXAMINED?
17132        JMP   B?          YES - EXIT
17134        AAD   Z,Ø1Ø
17137        JMP   H2
             *
             *
             *     BUS-II TRAP RETURN
             *
17141 HR     ELB   MB          SAVE B REGISTER
17143        STB   RB+1
17145#       TBB
17146        STB   RB+2
1715Ø#       RTR   A,B         SAVE A REGISTER
17151        STB   RA+1
17153#       TBB
17154        STB   RA+2
17156#       RTR   X,B         SAVE X REGISTER
17157        STB   RX+1
17161#       TBB
17162        STB   RX+2
17164#       RTR   Y,B         SAVE Y REGISTER
17165        STB   RY+1
17167#       TBB
1717Ø        STB   RY+2
17172#       RTR   Z,B         SAVE Z REGISTER
17173        STB   RZ+1
17175#       TBB
17176        STB   RZ+2
172ØØ#       AZE   A
172Ø1        SKR   2           TRAP 1 RETURN?
             ZY                RETAIN AUTO-SHARE
172Ø2#       ALD   A,Ø1Ø       NO
172Ø4        ELB   HB          TRAP TABLE BASE ADDRESS
172Ø6#       AAD   A,B
172Ø7        RTR   A,Z
17211#       RDC   B,B
17212        STB   HC
17214        JMP   H4
             *
17216 HB     ZA    HA          TRAP TABLE BASE ADDRESS
```

```
17220 HC    ZC    Ø          TRAP SWITCH
      ※
      ※     BUS-II TRAP TABLE
      ※
      ※     TRAP 1
      ※
17221 HA    ZA    Ø          TRAP 1 ADDRESS
17223       ZA    Ø          TRAP 1 SAVE LOCATIONS
17225       ZC    Ø          TRAP 1 FLAG
17226       ZC    Ø          TRAP 1 MODE
17227       JMP   MT         TRAP 1
      ※
      ※     TRAP 2
      ※
17231       ZA    Ø          TRAP 2 ADDRESS
17233       ZA    Ø          TRAP 2 SAVE LOCATIONS
17235       ZC    Ø          TRAP 2 FLAG
17236       ZC    Ø          TRAP 2 MODE
17237       JMP   MT+2       TRAP 2
      ※
            ZD

16773  176 216 Ø47 Ø61 Ø61

17ØØØ  316 22Ø 357 36Ø 111 156 Ø51 222
17Ø1Ø  131 156 124 ØØØ 332 Ø5Ø 141 3Ø7
17Ø2Ø  225 223 Ø5Ø 141 3Ø5 Ø1Ø 2Ø5 226
17Ø3Ø  Ø1Ø 133 ØØØ 336 Ø53 Ø5Ø 141 3Ø7
17Ø4Ø  225 223 Ø5Ø 141 3Ø5 ØØ2 331 155
17Ø5Ø  1Ø1 222 111 156 124 ØØØ 3Ø2 Ø1Ø
17Ø6Ø  133 ØØØ 3Ø6 ØØ2 3Ø1 Ø1Ø 233 221
17Ø7Ø  Ø5Ø 141 3Ø5 Ø2Ø 226 Ø53 Ø1Ø 133

171ØØ  221 Ø5Ø 141 3Ø5 818 233 Ø54 Ø5Ø
1711Ø  157 1Ø3 356 22Ø 111 155 1Ø1 Ø61
1712Ø  316 22Ø 155 257 Ø1Ø 3Ø3 176 216
1713Ø  Ø24 131 155 15Ø ØØ4 313 Ø1Ø 156
1714Ø  ØØ2 161 376 315 366 Ø61 315 367
1715Ø  Ø4Ø 315 371 Ø61 315 372 Ø42 315
1716Ø  363 Ø61 315 364 Ø44 315 36Ø Ø61
1717Ø  315 361 Ø46 315 355 Ø61 315 356

172ØØ  Ø2Ø 1Ø5 Ø21 Ø1Ø 176 216 822 Ø1Ø
1721Ø  33Ø Ø52 316 22Ø 156 Ø55 221 Ø16
1722Ø  ØØØ ØØØ ØØØ ØØØ ØØØ ØØØ ØØØ 141
1723Ø  353 ØØØ ØØØ ØØØ ØØØ ØØØ ØØØ 141
1724Ø  355
      ※
      ※     PGS BASIC UTILITY SYSTEM II (BUS-II)
      ※     88YØØØØ2A  REV A  1-15-69  SHEET 5 NEXT 6
      ※
```

```
      *     BUS-II GO ROUTINE
      *
      *     THIS ROUTINE LOADS REGISTERS A, B, X, Y, AND Z
      *     WITH THE CONTENTS OF PSUEDO-REGISTERS A, B, X,
      *     Y, AND Z AND TRANSFERS CONTROL TO THE LOCATION
      *     SPECIFIED IN Y AND THE MODE SPECIFIED IN OM.
      *
      *
17241 GA    LDB   OF          GET DATA IN FLAG
17243       SKN   2           ANY DATA IN?
17244       JMP   B?          NO - EXIT
      *
17246       ELB   RB+1        SAVE PSUEDO-B IN LOW-CORE
17250       STB   MB
17252#      TBB
17253       STR   MB+1
17255       RTR   YY,A        BUILD JUMP INSTRUCTION
17257#      AAD   A,140
17261#      RTR   A,B
17262       STB   MH
17264#      RTR   Y,B
17265       STB   MH+1
      *
17267       ELB   RA+1        LOAD A
17271#      RTR   B,A
17272       ELB   RX+1        LOAD X
17274#      RTR   B,X
17275       ELB   RY+1        LOAD Y
17277#      RTR   B,Y
17300       ELB   RZ+1        LOAD Z
17302#      RTR   B,Z
17303       LDB   OM          GET MODE
17305       SKN   2
17306       RIC   H,B
17310       JMP   MG          EXIT TO MODE CONTROL
      *
            ZD

17241  357 360 111 155 150 175 366
17250  301 376 061 301 377 010 306 023
17260  140 040 301 374 044 301 375 175
17270  371 041 175 363 043 175 360 045

17300  175 355 047 357 361 111 010 155
17310  141 371
      *
      *     PGS BASIC UTILITY SYSTEM II (BUS-II)
      *     88Y00002A  REV A  1-15-69  SHEET 6 NEXT 7
      *
      *     BUS-II LOADER CALLER
```

```
      ※
      ※     THIS ROUTINE TRANSFERS CONTROL FROM
      ※     BUS-II TO THE LOADER PROGRAM LOCATED
      ※     IN MODE Ø.
      ※
      ※
17312#LA    AZE   A
17313       LDB   OF          GET DATA IN FLAG
17315       SKZ   2           ANY DATA IN?
17316       RDC   B,A
1732Ø#      RTR   A,B         SET LOADER FLAG
17321       JMP   MF          EXIT TO MODE CONTROL
      ※
            ZD

17312  Ø2Ø 357 36Ø 131 Ø1Ø 2Ø5
1732Ø  Ø4Ø 141 342
      ※
      ※     PGS BASIC UTILITY SYSTEM II (BUS-II)
      ※     88YØØØØ2A  REV A  1-15-69  SHEET 7 NEXT 8
      ※
      ※     BUS-II DUMP/PUNCHER CALLER
      ※
      ※     THIS ROUTINE CALLS THE DUMP OR PUNCHER
      ※     PROGRAM DEPENDING ON THE KEYBOARD INPUT.
      ※     IN THE CASE OF PUNCHER, THIS ROUTINE OUT-
      ※     PUTS 64 BLANKS BEFORE AND AFTER THE CALL.
      ※
      ※
17323#A1    RIC   P,B         ACCEPT END ADDRESS
17324       JMP   OI
17326       LDB   OF          GET DATA IN FLAG
1733Ø       SKN   2           ANY DATA IN?
17331       JMP   B?          NO - EXIT
17333       LDB   OC          GET CONTROL CHARACTER
17335#      RTR   B,A
17336#      ASU   A,3Ø4
1734Ø       SKN   2           IS CHARACTER A D?
17341       JMP   D1          YES - GO TO DUMP
      ※
17343#A2    ASU   A,Ø14
17345       SKZ   2           IS CHARACTER A P?
17346       JMP   B?          NO - EXIT
      ※
      ※     PUNCH LEADER
      ※
1735Ø#      RTR   X,B         SAVE MODE
17351       STB   OM
17353#      RIC   P,B
17354       JMP   AT
```

```
      ;
      ;     PUNCH TAPE
      ;
17356       LDB   OM          RESTORE MODE
1736Ø#      RTR   B,X
17361       JMP   MP          EXIT TO MODE CONTROL
      ;
      ;     BUS-II PUNCHER RETURN
      ;
      ;     PUNCH TRAILER
      ;
17363#A3    RIC   P,B
17364       JMP   AT
17366       JMP   BB          EXIT TO BUS-II
      ;
      ;     LEADER/TRAILER SUBROUTINE
      ;
1737Ø AT    STB   AS          SAVE RETURN ADDR
17372#      TBB
17373       STB   AS+1
17375#      ALD   X,1ØØ
17377#AR    AZE   A           PUNCH A BLANK
174ØØ#      RIC   P,B
174Ø1       JMP   TO
174Ø3       RDC   X,X
174Ø5       SKZ   2           ALL DONE?
174Ø6       JMP   AR          NO
1741Ø       ELB   AS          EXIT
17412#      RIC   B,P
      ;
17413 AS    ZA    Ø           RETURN ADDRESS
      ;
            ZD

17323  Ø5Ø 157 253 357 36Ø
1733Ø  111 155 15Ø 357 357 Ø41 Ø25 3Ø4
1734Ø  111 157 Ø15 Ø25 Ø14 131 155 15Ø
1735Ø  Ø42 317 361 Ø5Ø 156 37Ø 357 361
1736Ø  Ø43 141 332 Ø5Ø 156 37Ø 155 1Ø5
1737Ø  317 Ø13 Ø61 317 Ø14 Ø32 1ØØ Ø2Ø

174ØØ  Ø5Ø 14Ø 12Ø Ø1Ø 211 131 156 377
1741Ø  177 Ø13 Ø51 ØØØ ØØØ
      ;
      ;     PGS BASIC UTILITY SYSTEM II (BUS-II)
      ;     88YØØØØ2A  REV A  1-15-69  SHEET 8 NEXT 9
      ;
      ;     BUS-II DUMP ROUTINE
      ;
```

```
      *     THIS ROUTINE PRINTS THE ADDRESSES AND
      *     CONTENTS OF MEMORY SPECIFIED:
      *
      *        X = MEMORY MODE
      *        Y = START ADDRESS
      *        Z = END ADDRESS
      *
      *
17415#D1    RTR   Z,B         SAVE END ADDRESS
17416       STB   DZ
1742Ø#      TBB
17421      STB   DZ+1
      *
      *     NEW PARAGRAPH
      *
17423 D2    RIC   P,Z         SEND CARRET LINE FEED
17425       JMP   CR
      *
      *     NEW LINE
      *
17427#D3    RIC   P,B         PRINT ADDRESS
1743Ø       JMP   AD
      *
      *     NEW LOCATION
      *
17432#D4    ALD   A,24Ø       SEND SPACE
17434#      RIC   P,B
17435       JMP   TO
17437#      RIC   P,B         GET DATA FROM MEMORY
1744Ø       JMP   ML
17442       RTR   A,Z
17444#      RIC   P,B         PRINT MEMORY CONTENTS
17445       JMP   03
17447       ELB   DZ          GET END ADDRESS
17451#      RTR   B,A
17452#      RTR   Y,B
17453#      ASU   A,B         END ADDR - START ADDR
17454       SKN   2
            ZY                RETAIN AUTO-SHARE
17455#      RTR   A,B
17456#      TBB
17457       SKN   2           ALL DONE?
1746Ø       JMP   BA          YES - EXIT
17462#      RIC   Y,Y         GET NEXT LOCATION
17463       RTR   Y,A
17465#      AND   A,Ø77
17467       SKN   2           NEW PARAGRAPH?
1747Ø       JMP   D2          YES
17472       AND   AØØ7
```

```
17474        SKN   2           NEW LINE?
17475        JMP   D3          YES
17477        JMP   D4
      *
175Ø1 DZ     ZA    Ø           END ADDRESS
      *
             ZD

17415  Ø46 317 1Ø1
1742Ø  Ø61 317 1Ø2 Ø1Ø 134 157 147 Ø5Ø
1743Ø  157 1Ø3 Ø21 24Ø Ø5Ø 14Ø 12Ø Ø5Ø
1744Ø  141 3Ø7 Ø1Ø 33Ø Ø5Ø 157 161 177
1745Ø  1Ø1 Ø41 Ø44 Ø24 111 Ø4Ø Ø61 111
1746Ø  155 1Ø1 Ø53 Ø1Ø 3Ø2 Ø26 Ø77 111
1747Ø  157 Ø23 Ø26 ØØ7 111 Ø57 Ø27 157

175ØØ  Ø32 ØØØ ØØØ
      *
      *      PGS BASIC UTILITY SYSTEM II (BUS-II)
      *      88YØØØØ2A  REV A  1-15-69  SHEET 9 NEXT 1Ø
      *
      *      BUS-II PRINT ADDRESS SUBROUTINE
      *
      *      THIS ROUTINE PRINTS A CARRET LINE FEED,
      *      THE MEMORY MODE IN X, THE MEMORY ADDRESS
      *      IN Y, AND A SPACE.
      *
      *      CALLING SEQUENCE
      *
      *         RIC   P,B
      *         JMP   AD
      *
      *
175Ø3 AD     STB   AE          SAVE RETURN ADDRESS
175Ø5#       TBB
175Ø6        STB   AE+1
      *
1751Ø        RIC   P,Z         SEND CARRET LINE FEET
17512        JMP   CR
17514        RTR   YY,A
17516#       AND   A,Ø1Ø       CHECK FOR UPPER MODE
1752Ø        SKZ   2           UPPER MODE?
17521        RTR   X,A         YES
17523#       AAD   A,26Ø
17525#       RIC   P,B         PRINT MODE
17526        JMP   TO
1753Ø        RTR   Y,Z
17532#       RIC   P,B         PRINT ADDRESS
17533        JMP   04
17535#       ALD   A,24Ø       SEND SPACE
```

```
17537#      RIC   P,B
17540       JMP   TO
17542       ELB   AE          EXIT
17544#      RIC   B,P
      ※
17545 AE    ZA    Ø           RETURN ADDRESS
      ※
            ZD

17503  317 145 061 317 146
17510  010 134 157 147 010 306 026 010
17520  131 010 301 023 260 050 140 120
17530  010 332 050 157 163 021 240 050
17540  140 120 177 145 051 000 000
      ※
      ※
      ※     PGS BASIC UTILITY SYSTEM II (BUS-II)
      ※     88Y00002A  REV A 1-15-69  SHEET 10 NEXT 11
      ※
      ※     BUS-II CARRIAGE RETURN LINE FEED SUBROUTINE
      ※
      ※     THIS ROUTINE OUTPUTS 1 CARRIAGE RETURN
      ※     AND 1 LINE FEED.
      ※
      ※
17547#CR    ALD   A,215       SEND CARRET
17551#      RIC   P,B
17552       JMP   TO
17554#      ALD   A,212       SEND LINE FEED
17556#      RTR   Z,B
17557       JMP   TO
      ※
            ZD

17547  021
17550  215 050 140 120 021 212 046 140
17560  120
      ※
      ※     PGS BASIC UTILITY SYSTEM II (BUS-II)
      ※     88Y00002A  REV A 1-15-69  SHEET 11 NEXT 12
      ※
      ※     BUS-II OCTAL OUTPUT SUBROUTINE
      ※
      ※     THIS ROUTINE CONVERTS FROM BINARY TO
      ※     3 OR 4 OCTAL ASCII DIGITS AND OUTPUTS
      ※     THEM TO THE TTY ASR.
      ※
      ※     CALLING SEQUENCE
      ※
      ※     Z = 12-BIT OR 8-BIT BINARY NUMBER
```

```
      *
      *          RIC   P,B
      *          JMP   03    (OUTPUTS 3 DIGITS)
      *     OR   JMP   04    (OUTPUTS 4 DIGITS)
      *
      *
17561#03      PLR               RESET LINK
17562         SKR   2
17563 04      PLS               SET LINK
17565         STB   OS          SAVE RETURN ADDRESS
17567#        TBB
1757Ø         STB   OS+1
17572         SKS   2           4 DIGITS?
17573         JMP   05          NO
17575         RTR   ZZ,A
17577#        SHR   A
176ØØ#        AAD   A,26Ø
176Ø2#        RIC   P,B         PRINT DIGIT 4
176Ø3         JMP   TO
      *
176Ø5 05      RTR   ZZ,A
176O7#        SHR   A
1761Ø         RTR   Z,A
17612#        SHL   A
17613#        SHR   A
17614#        SHR   A
17615#        SHR   A
17616#        SHR   A
17617#        SHR   A
1762Ø#        AAD   A,26Ø
17622#        RIC   P,B         PRINT DIGIT 3
17623         JMP   TO
17625         RTR   Z,A
17627#        SHR   A
1763Ø#        SHR   A
17631#        SHR   A
17632#        AND   A,ØØ7
17634#        AAD   A,26Ø
17636#        RIC   P,B         PRINT DIGIT 2
17637         JMP   TO
17641         RTR   Z,A
17643#        AND   A,ØØ7
17645#        AAD   A,26Ø
17647         ELB   OS
17651         JMP   TO          PRINT DIGIT 1
```

```
          ZD

17561  Ø71 1Ø5 Ø14 2Ø1 317 355 Ø61
1757Ø  317 356 125 157 2Ø5 Ø1Ø 3Ø7 Ø66

176ØØ  Ø23 26Ø Ø5Ø 14Ø 12Ø Ø1Ø 3Ø7 Ø66
1761Ø  Ø1Ø 3Ø3 Ø7Ø Ø66 Ø66 Ø66 Ø66 Ø66
1762Ø  Ø23 26Ø Ø5Ø 14Ø 12Ø Ø1Ø 3Ø3 Ø66
1763Ø  Ø66 Ø66 Ø26 ØØ7 Ø23 26Ø Ø5Ø 14Ø
1764Ø  12Ø Ø1Ø 3Ø3 Ø26 ØØ7 Ø23 26Ø 177
1765Ø  355 14Ø 12Ø
      *
      *     PGS BASIC UTILITY SYSTEM II (BUS-II)
      *     88YØØØØ2A  REV A  1-15-69  SHEET 12 NEXT 13
      *
      *     BUS-II OCTAL INPUT SUBROUTINE
      *
      *     THIS ROUTINE ACCEPTS UP TO 5 OCTAL ASCII
      *     DIGITS FROM THE TTY ASR AND CONVERTS THEM
      *     TO BINARY.
      *
      *     CALLING SEQUENCE
      *
      *        RIC   P,B
      *        JMP   O1
      *
      *     UPON EXIT
      *        Z = 4 LEAST SIGNIFICANT OCTAL
      *            DIGITS IN BINARY
      *        OM = MOST SIGNIFICANT OCTAL DIGIT
      *             IN BINARY
      *        OC - LAST CHARACTER RCVD (NON-OCTAL)
      *        OF = ØØØ, NO OCTAL DIGITS RCVD
      *           = 377, AT LEAST 1 DIGIT RCVD
      *
      *
17653 OI    STB   OS          SAVE RETURN ADDRESS
17655#      TBB
17656       STB   OS+1
1766Ø       AZE   Z
17662#      RTR   Z,B
17663       STB   OF          RESET DATA IN FLAG
17665       STB   ØØ          ZERO MSB
      *
17667#OJ    RIC   P,B         ACCEPT INPUT
1767Ø       JMP   TE
17672#      AND   A,177
17674       SKN   2           CHARACTER = BLANK?
17675       JMP   OJ          YES - IGNORE
17677#      AAD   A,2ØØ       FORCE LEVEL-8
```

```
17701#      RTR   A,B
17702       STB   OC          SAVE CHARACTER
17704#      ASU   A,270
17706       SKM   2           IS CHARACTER GREATER THAN 7?
17707       JMP   OX          YES
17711#      AAD   A,0100
17713       SKP   2           IS CHARACTER LESS THAN 0?
17714       JMP   OX          YES
17716       RTR   ZZ,A        SAVE MSB
17720#      SHR   A
17721#      RTR   A,B
17722       STB   OM
17724#      RTR   Z,B         MULT OLD NO. BY 8
17725       AAD   Z,B
17727#      RTR   Z,B
17730       AAD   Z,B
17732#      RTR   Z,B
17733       AAD   Z,B
17735       LDB   OC          GET NEW DIGIT
17737#      RTR   B,A
17740#      ASU   A,260       CONVERT BINARY
17742#      RTR   A,B
17743       AAD   Z,B         OLD + NEW
17745#      RDC   B,B
17746       STB   OF          SET DATA IN FLAG
17750       JMP   OJ
            *
            *     EXIT
            *
17752 OX    ELB   OS
17754#      RIC   B,P
            *
17755 OS    ZA    0           USER RETURN ADDRESS
17757 OC    ZC    0           LAST CHARACTER RCVD
17760 OF    ZC    0           DATA IN FLAG
17761 OM    ZC    0           MOST SIGNIFICANT DIGIT (MODE)
            *
                  ZD

17653  317 355 061 317 356
17660  004 003 046 317 360 317 361 050
17670  140 216 026 177 111 157 267 023

17700  200 040 317 357 025 270 115 157
17710  352 023 010 135 157 352 010 307
17720  066 040 317 361 046 004 303 046
17730  004 303 046 004 303 357 357 041
17740  025 260 040 004 303 052 317 360
17750  157 267 177 355 051 000 000 000
17760  000 000
```

```
      *
      *     PGS BASIC UTILITY SYSTEM II (BUS-II)
      *     88YØØØØ2A  REV A  1-15-69  SHEET 13 NEXT 14
      *
      *     BUS II BOOT LOADER PUNCH CALLER
      *
      *     THIS ROUTINE OUTPUTS 64 BLANKS AND THEN TRANSFERS
      *     CONTROL FROM BUS II TO THE PUNCHER PROGRAM LOCATE
      *     IN MODE Ø.
      *
17762#BP    RIC   P,B
17763       JMP   AT          OUTPUT LEADER
17765#      ALD   X,117
17767       RIC   B,Y         START ADDRESS TO Y
17771       STB   LS          LOAD LOADER IN CONSTANT
17773#      AAD   X,357       END ADDRESS IN X
17775#      RTR   B,Z         BLOCK LENGTH TO Z
17776       JMP   PK
      *
            ZD

17762  Ø5Ø 156 37Ø Ø32 117 Ø1Ø
1777Ø  125 3Ø1 Ø76 Ø33 357 Ø47 141 Ø77
      *
      *     PGS BASIC UTILITY SYSTEM II (BUS-II)
      *     88YOOOO2A  REV A  1-27-69  SHEET 14 NEXT 14
      *
      *     BUS-II MODE CONTROL ROUTINES
      *
      *     THESE ROUTINES PROVIDE THE MODE CONTROL
      *     FUNCTIONS REQUIRED FOR BUS-II TO COMMUNICATE
      *     TO ANY MEMORY MODE IN THE SPC-12.
      *     LOCATION MM CONTAINS THE CURRENT BUS-II MODE.
      *     IF BUS-II IS RELOCATED TO ANY OTHER MODE, MM
      *     MUST BE CHANGED TO THAT MODE.
      *
      *
      *     CURRENT BUS-II MODE
      *
ØØ7Ø4 MM    ZC    ØØ1
      *
      *     MEMORY STORE AND LOAD SUBROUTINE
      *
ØØ7Ø5#MS    PLS               SET LINK
ØØ7Ø6       SKS   2
ØØ7Ø7 ML    PLR               RESET LINK
ØØ711       STB   MB          SAVE RETURN ADDR
ØØ713#      TBB
ØØ714       STB   MB+1
ØØ716#      RTR   X,B         SET MODE
```

```
ØØ717#      TBE
ØØ72Ø       SKR   2          STORE?
ØØ721       GØS   A,Y        YES
ØØ723#      GØL   A,Y
ØØ724       LDB   MM         SET BUS-II MODE
ØØ726#      TBE
ØØ727       ELB   MB         EXIT
ØØ731#      RIC   B,P
      *
      *     PUNCHER CALLER SUBROUTINE
      *
ØØ732#MP    RIC   P,B        GO TO PUNCHER
ØØ733       JMP   PP
ØØ735       LDB   MM         SET BUS-II MODE
ØØ737#      TBE
ØØ74Ø       JMP   A3         EXIT
      *
      *     LOADER CALLER SUBROUTINE
      *
ØØ742 MF    RIC   P,Z        GO TO LOADER
ØØ744       JMP   LD
ØØ746       LDB   MM         SET BUS-II MODE
ØØ75Ø#      TBE
ØØ751       JMP   BB         EXIT
      *
      *     TRAP RETURN ROUTINE
      *
ØØ753#MT    PLR              TRAP 1 RETURN
ØØ754       SKR   2
ØØ755       PLS              TRAP 2 RETURN
ØØ757       STB   MB         SAVE B REGISTER
ØØ761#      TBB
ØØ762       STB   MB+1
ØØ764       LDB   MM         SET BUS-II MODE
ØØ766#      TBE
ØØ767       JMP   RR         EXIT
      *
      *     GO EXIT ROUTINE
      *
ØØ771#MG    TBE              SET MODE
ØØ772       ELB   MB         LOAD B REGISTER
ØØ774 MH    JMP   MH         GO
      *
ØØ776 MB    ZA    Ø          B-REGISTER SAVE
      *
            ZD

ØØ7Ø4  ØØ1 Ø72 125 Ø14
ØØ71Ø  2ØØ 3Ø1 376 Ø61 3Ø1 377 Ø42 Ø62
ØØ72Ø  1Ø5 ØØØ 23Ø 21Ø 341 3Ø4 Ø62 161
```

```
ØØ73Ø  376 Ø51 Ø5Ø 141 113 341 3Ø4 Ø62
ØØ74Ø  156 363 Ø1Ø 134 14Ø 275 341 3Ø4
ØØ75Ø  Ø62 155 1Ø5 Ø71 1Ø5 Ø14 2Ø1 3Ø1
ØØ76Ø  376 Ø61 3Ø1 377 341 3Ø4 Ø62 156
ØØ77Ø  141 Ø62 161 376 141 374 ØØØ ØØØ
```

APPENDIX E
GA RECOMMENDED SHARED COMMAND BYTES

```
        *   GAI RECOMMENDED SHARED COMMAND BYTES
        *   88YØØØ15A  REV A  Ø1-27-69  SHEET 1 NEXT 1
        *
        *   THIS RECOMMENDATION IS BASED UPON THESE
        *   REGISTERS BEING ASSIGNED TO THE FOLLOWING
        *   GENERAL FUNCTIONS.
        *
        *     A=PROCESSING REGISTER
        *     B=OPERAND BUFFER AND SUBROUTINE RETURN
        *       ADDRESS REGISTER
        *     X=INDEXING CONTROL REGISTER
        *     Y=ADDRESSING OR INDEX CONTROL REGISTER
        *     Z=ADDRESSING OR INDEX CONTROL OR ALTERNATE
        *       SUBROUTINE RETURN ADDRESS REGISTER
        *
        *
ØØØ2Ø       ZE    AZE   A
ØØØ21       ZE    ALD   A,Ø
ØØØ22       ZE    AAD   A,B
ØØØ23       ZE    AAD   A,Ø
ØØØ24       ZE    ASU   A,B
ØØØ25       ZE    ASU   A,Ø
ØØØ26       ZE    AND   A,Ø
ØØØ27       ZE    AXR   A,Ø
ØØØ3Ø       ZE    AOR   A,Ø
ØØØ31       ZE    AZE   X
ØØØ32       ZE    ALD   X,Ø
ØØØ33       ZE    AAD   X,Ø
ØØØ34       ZE    ASU   X,Ø
        *
ØØØ4Ø       ZE    RTR   A,B        REGISTER TRANSFER
ØØØ41       ZE    RTR   B,A        SHARED INSTRUCTIONS
ØØØ42       ZE    RTR   X,B
ØØØ43       ZE    RTR   B,X
ØØØ44       ZE    RTR   Y,B
ØØØ45       ZE    RTR   B,Y
ØØØ46       ZE    RTR   Z,B
ØØØ47       ZE    RTR   B,Z
ØØØ5Ø       ZE    RIC   P,B
ØØØ51       ZE    RIC   B,P
ØØØ52       ZE    RDC   B,B
ØØØ53       ZE    RIC   Y,Y
ØØØ54       ZE    RDC   Y,Y
        *
```

```
ØØØ6Ø       ZE    INE                 PULSE/SHIFT SHARED
ØØØ61       ZE    TBB
ØØØ62       ZE    TBE
ØØØ63       ZE    FOB
ØØØ64       ZE    DOB
ØØØ65       ZE    DIB
ØØØ66       ZE    SHR   A
ØØØ67       ZE    SHC   A
ØØØ7Ø       ZE    SHL   A
ØØØ71       ZE    PLR
ØØØ72       ZE    PLS
      ×
ØØ1ØØ       ZE    GØL   A,X           AUGMENTED SHARED
ØØ1Ø1       ZE    GØS   A,X
ØØ1Ø2       ZE    GØL   A,Y
ØØ1Ø3       ZE    GØS   A,Y
ØØ1Ø4       ZE    GØL   A,Z
ØØ1Ø5       ZE    GØS   A,Z
ØØ1Ø6       ZE    GØL   A,X,I
ØØ1Ø7       ZE    GØS   A,X,I
      ×
      ×     THE FOLLOWING SHARED COMMAND BYTES ASSUME
      ×     X IS INDEXING CONTROL SELECTED IN THE
      ×     INDIRECT ADDRESS WORD
      ×
ØØ11Ø       ZE    GØL   A,Y,I
ØØ111       ZE    GØS   A,Y,I
ØØ112       ZE    GØL   A,Z,I
ØØ113       ZE    GØS   A,Z,I
      ×
      ×
            ZD    S

ØØØ2Ø  ØØØ 11Ø 3ØØ 31Ø 2ØØ 21Ø Ø3Ø Ø5Ø
ØØØ3Ø  Ø7Ø ØØ1 111 311 211

ØØØ4Ø  35Ø 3Ø5 351 315 352 325 353 335
ØØØ5Ø  154 145 255 122 222

ØØØ6Ø  2Ø7 2Ø5 2Ø6 2Ø2 2Ø3 2Ø4 ØØØ Ø2Ø
ØØØ7Ø  Ø1Ø 2ØØ 2Ø1

ØØ1ØØ  1ØØ 13Ø 2ØØ 23Ø 3ØØ 33Ø 14Ø 17Ø
ØØ11Ø  24Ø 27Ø 34Ø 37Ø
```

APPENDIX F
PGS CAS II SOURCE TAPE PREPARATION ROUTINE

```
                ZO   62ØØ
          *
          *     PGS CAS II SOURCE TAPE PREPARATION
          *     88YØØØ17A REV A  1-18-69  SHEET 1 NEXT 1
          *
          *     THIS ROUTINE IS USED IN THE PREPARATION OF
          *     PGS CAS I SOURCE TAPES.
          *     THE PROPER SPACING BETWEEN THE DIFFERENT FIELDS
          *     IS AUTOMATICALLY PROVIDED FOR AS IS THE
          *     TERMINATION OF THE COMMENT FIELD.
          *
16200     ZC    1ØØ
          *
          *     CARRIAGE RETURN
          *
16201 CR  ALD   A,215      LOAD A WITH CARRIAGE RETURN
16204     RIC   P,B
16206     JMP   TO          OUTPUT CARRIAGE RETURN
          *
          *     LINE FEED AND RUBOUTS
          *
16210 LF  ALD   A,212      LOAD A WITH LINE FEED
16213     RIC   P,B
16215     JMP   TO          OUTPUT LINE FEED
16217     ALD   X,Ø12       LOAD X WITH NUMBER OF RUBOUTS
16222 L1  RIC   P,B
16224     JMP   TO          OUTPUT RUBOUTS
16226     RDC   X,X         DECREMENT RUBOUT COUNT
16230     SKZ   2           ALL DONE?
16231     JMP   L1          NO, CONTINUE
          *
          *     CHECK FIRST CHARACTER
          *
16233 CC  RIC   P,B         INPUT CHARACTER
16235     JMP   TE
16237     AOR   A,2ØØ       FORCE BIT 8 ON
16242     AXR   A,377       IGNORE RUBOUT
16245     SKN
16246     JMP   CC
16250     AXR   A,177       IGNORE BLANKS
16253     SKN   2
16254     JMP   CC
16256     ALD   Y,ØØ6
16261     AXR   A,Ø52
16264     SKN   2           IS INPUT AN ASTERISK?
```

```
16265          JMP   AS          YES, JUMP TO ASTERISK INPUT
16267          ALD   X,ØØ3       LOAD INDEX
16272          AXR   A,Ø12       NO
16275          SKN   6           IS INPUT A SPACE?
16276          RIC   P,Z         YES, GO TO OUTPUT SPACES
163ØØ          JMP   SP
163Ø2          RDC   X,X         ADVANCE TO NEXT FIELD COUNT
         *
         *     STANDARD INPUT
         *
163Ø4 S1       LDB   FC-1,K      LOAD FIELD CONSTANT
163Ø6          STB   S2+2
16312 S2       ALD   Y,649       FIELD CONSTANT TO Y
16313 S3       RIC   P,B
16315          JMP   TE          INPUT CHARACTER
16317          AXR   A,377
16322          SKN   4           IS INPUT RUBOUT?
16323          RIC   Y,Y         YES, INCREMENT FIELD CONSTANT
16325          JMP   S3
16327          RDC   Y,Y         DECREMENT FIELD CONSTANT
16331          AXR   A,162       NO
16334          SKN   2           IS INPUT A CARRET?
16335          JMP   LF          YES, JUMP TO LINE FEED
16337          AXR   A,155       NO
16342          SKN   4           IS INPUT A SPACE?
16343          RIC   P,Z         YES
16345          JMP   SP          JUMP TO OUTPUT SPACES
16347          RTR   Y,Y         NO, TEST FIELD CONSTANT
16351          SKZ   2           NEW FIELD?
16352          JMP   S3          NO, CONTINUE
16354          RDC   X,X         YES
16356          SKZ   2           TEST WHICH FIELD
16357          JMP   S1          NOT COMMENT FIELD, CONTINUE
16361          ALD   Y,844       COMMENT FIELD, LOAD COMMENT CONSTANT
16364          JMP   CF
         *
         *     OUTPUT SPACES
         *
16366 SP       ALD   A,24Ø       LOAD A WITH A SPACE
16371          RIC   P,5
16373          JMP   TO          OUTPUT SPACE
16375          RDC   Y,Y         ALL DONE?
16377          SKZ   2
164ØØ          JMP   SP          NO
164Ø2          RIC   Z,P         YES
         *
         *     ASTERISK INPUT
         *
164Ø4 AS       ALD   Y,Ø72       LOAD LONG COMMENT CONSTANT
         *
```

```
      ;
      ;     COMMENT FIELD
      ;
16407 CF    RIC   P,B        INPUT COMMENT
16411       JMP   TE
16413       AXR   A,377
16416       SKN   4          IS INPUT A RUBOUT?
16417       RIC   Y,Y
16421       JMP   CF         YES, DELETE CHARACTER
16423       AXR   A,162      NO
16426       SKN   2          IS INPUT A CARRET?
16427       JMP   LF         YES, JUMP TO LINE FEED
16431       RDC   Y,Y
16433       SKZ   2          IS COMMENT TOO LONG?
16434       JMP   CF         NO, CONTINUE
16436       JMP   CR         YES, JUMP TO CARRET
      ;
16440 FC    ZC    13
16441       ZC    6
16442       ZC    6
      TO    ZL    120
      TE    ZL    216
      ;
            ZD

162ØØ  1ØØ ØØ4 11Ø 215 Ø1Ø 154 14Ø 12Ø
1621Ø  ØØ4 11Ø 212 Ø1Ø 154 14Ø 12Ø ØØ4
1622Ø  111 Ø12 Ø1Ø 154 14Ø 12Ø Ø1Ø 211
1623Ø  131 154 222 Ø1Ø 154 14Ø 216 ØØ4
1624Ø  Ø7Ø 2ØØ ØØ4 Ø5Ø 377 111 154 233
1625Ø  ØØ4 Ø5Ø 177 111 154 233 ØØ4 112
1626Ø  ØØ6 ØØ4 Ø5Ø Ø52 111 155 ØØ4 ØØ4
1627Ø  111 ØØ3 ØØ4 Ø5Ø Ø12 113 Ø1Ø 134
163ØØ  154 366 Ø1Ø 211 375 Ø37 314 312
1631Ø  ØØ4 112 ØØØ Ø1Ø 154 14Ø 216 ØØ4
1632Ø  Ø5Ø 377 112 Ø1Ø 122 154 313 Ø1Ø
1633Ø  222 ØØ4 Ø5Ø 162 111 154 21Ø ØØ4
1634Ø  Ø5Ø Ø55 112 Ø1Ø 134 154 366 Ø1Ø
1635Ø  322 131 154 313 Ø1Ø 211 131 154
1636Ø  3Ø4 ØØ4 112 Ø44 155 ØØ7 ØØ4 11Ø
1637Ø  24Ø Ø1Ø 154 14Ø 12Ø Ø1Ø 222 131

164ØØ  154 366 Ø1Ø 143 ØØ4 112 Ø72 Ø1Ø
1641Ø  154 14Ø 216 ØØ4 Ø5Ø 377 112 Ø1Ø
1642Ø  122 155 ØØ7 ØØ4 Ø5Ø 162 111 154
1643Ø  21Ø Ø1Ø 222 131 155 ØØ7 154 2Ø1
1644Ø  Ø13 ØØ6 ØØ6
```