PHILIPS

Electronic components and materials

PHILIPS

FIXED POINT DECIMAL ARITHMETIC ROUTINES AS55

AN APPLICATION MEMO

signetics

INTRODUCTION

The numbers used in digital systems are usually expressed in binary notation. Some commonly used formats are:

- magnitudes only for unsigned numbers
- 1's complement and 2's complement for signed numbers.

However, binary numbers are difficult to interpret, and man-machine interface can be greatly improved by presenting numbers in decimal notation. Since virtually all digital systems operate on numbers in binary form (i.e., 1's and 0's), decimal numbers must be converted to binary during the input process, and reconverted to decimal notation during the output process. In cases where decimal input and/or output is required, the ideal solution would be a digital system capable of interpreting and processing decimal numbers.

This applications memo describes several methods of handling binary-coded-decimal (BCD) numbers with the Signetics 2650 microprocessor. Special provisions in the 2650 for these operations, including the Interdigit Carry (IDC) flag bit and the Decimal Adjust Register (DAR) instruction, are discussed. These provisions greatly simplify interfacing of the 2650 to decimal-oriented peripheral devices, such as CRT display terminals, printers, and keyboards. Basic arithmetic routines (add, subtract, multiply, and divide) for both signed integers and signed fixed-point numbers are given.

BCD NOTATION

In BCD notation, each decimal digit requires a 4-bit code as indicated below:

0 = 0000	5 = 0101
1 = 0001	6 = 0110
2 = 0010	7 = 0111
3 = 0011	8 = 1000
4 = 0100	9 = 1001

Codes 1010 through 1111 are not used.

Two decimal digits can be packed into one 8-bit byte—the size of a 2650 data word. The range within 1 byte is consequently 00_{10} through 99_{10}. For instance, the number 15_{10} is coded as 00010101.

CARRY (C) AND INTERDIGIT CARRY (IDC) FLAGS

The Program Status Lower (PSL) of the 2650's Program Status Word (PSW) register contains 2 carry flags: Carry (C) and Interdigit Carry (IDC). During execution of any arithmetic instruction, both flags are set or reset depending on the result of the operation, as illustrated in Figure 1:

- The Carry (C) flag is set as a result of a carry (or no borrow) out of the most-significant-bit (bit 7) of the affected register Rx, and hence out of the most-significant BCD digit.
- The Interdigit Carry (IDC) flag is set as a result of a carry (or no borrow) out of bit 3, and hence out of the least-significant BCD digit and into the most-significant BCD digit.

DECIMAL ADJUST REGISTER (DAR) INSTRUCTION

If 2 BCD numbers are added or subtracted by means of binary arithmetic instructions, the result may not be a BCD number. For example:

$$23_{16} + 56_{16} = 79_{16};$$
but
$$18_{16} + 35_{16} = 4D_{16}.$$

Since the binary codes 1010 (A_{16}) through 1111 (F_{16}) are not used in BCD, the result of a binary arithmetic instruction may need a correction of (+6) in case of an add operation or (−6) in case of a subtract operation. The 2650 performs this correction by means of the Decimal Adjust Register (DAR) instruction. This 1-byte instruction conditionally adds a decimal 10 (2's complement negative 6 in a 4-bit binary number system) to either the high order 4-bits and/or the low order 4 bits of a specified register Rx, which may be any of the 2650's seven CPU registers.

The truth table of Figure 2 indicates the logical operation performed. The operation proceeds based on the values of the Carry (C) and Interdigit Carry (IDC) flags in the Program Status Word. The C and IDC remain unchanged by the execution of this instruction.

The WC (With/Without Carry) bit in PSL has no influence on the DAR instruction.

GENERAL SUBTRACTION RULES

In the case of subtraction, a correction of (−6) is required for the digit(s) which generate a borrow upon execution of the subtract instruction. This can be performed directly by the DAR instruction.

Single-Byte Operands/ Result:

Subtraction of single-byte operands is done by performing the subtract instruction and then performing the DAR instruction; the borrow bit must be cleared initially. See Example A.

SETTING THE CARRY AND INTERDIGIT CARRY FLAGS

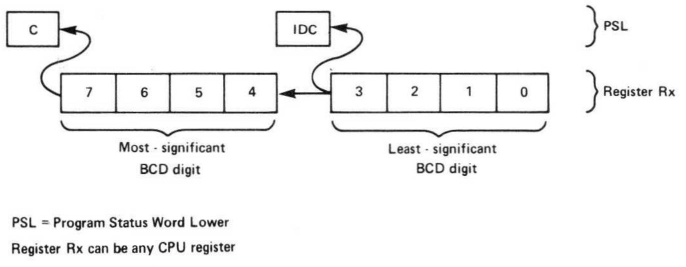

C
IDC
PSL
7
6
5
4
3
2
1
0
Register Rx
Most - significant BCD digit
Least - significant BCD digit
PSL = Program Status Word Lower
Register Rx can be any CPU register

Figure 1

TRUTH TABLE FOR DAR INSTRUCTION

BEFORE: DAR, Rx				AFTER: DAR, Rx			
		Rx				Rx	
C	IDC	MSD	LSD	C	IDC	MSD	LSD
0	0	a	b	0	0	$a+10_{10}$	$b+10_{10}$
0	1	a	b	0	1	$a+10_{10}$	b
1	0	a	b	1	0	a	$b+10_{10}$
1	1	a	b	1	1	a	b

NOTE

IDC is not added to the upper digit in the '$a+10_{10}$' operation.

Figure 2

If the With Carry (WC) bit in PSL is zero (no carry/borrow), the first instruction is not required.

```
        PPSL      C        CLEAR BORROW
        LODA,R3   OPR1     FETCH FIRST OPERAND
        SUBA,R3   OPR2     SUBTRACT SECOND OPERAND
        DAR,R3             DECIMAL ADJUST RESULT
        STRA,R3   RSLT     STORE RESULT
```

Example A

Multiple-Byte Operands/ Result:

When dealing with multiple-byte operands, arithmetic operations *including carry,* are required. Hence, the WC bit in PSL must be set to 1 prior to execution. If indexing is used, multiple-byte subtraction is simple, as illustrated in Example B.

NOTE: OPR1, OPR2 and RSLT are the most-significant bytes.

```
        PPSL      WC+C         ARITHMETIC WITH CARRY, CLEAR BORROW
        LODI,R3   LENG         LOAD INDEX REGISTER
DSUL    LODA,R0   OPR1,R3,-    FETCH BYTE OF OPERAND1
        SUBA,R0   OPR2,R3      SUBTRACT BYTE OF OPERAND2
        DAR,R0                 DECIMAL ADJUST RESULT
        STRA,R0   RSLT,R3      STORE RESULTING BYTE
        BRNR,R3   DSUL         CONTINUE LOOP IF NOT DONE
```

Example B

GENERAL ADDITION RULES

For addition, a correction of (+6) is required if the sum of the most-significant digits or least-significant digits exceeds 9. This is accomplished by first adding an offset of (+6) to each of the digits of the first operand (addition of H'66') and then adding the second operand.

If the sum of the least-significant digits did exceed 9, it now (including the (+6) correction) will exceed 15_{10}, (H'F'); an Interdigit Carry will be generated. If an IDC is generated, the result is correct and, as shown in Figure 2, the DAR instruction will have no effect on the sum. If not, the (+6) correction will be cancelled by adding 10 (equivalent to subtracting 6). Correction of the most-significant digit sum operates similarly, with the C bit controlling the final correction.

Single-Byte Operands/Result:

If the 2650 is conditioned for arithmetic without carry (WC = 0), addition can be performed as shown in Example C.

```
        LODA,R3   OPR1     FETCH FIRST OPERAND
        ADDI,R3   H'66'    ADD OFFSET FOR BCD ADD
        ADDA,R3   OPR2     ADD SECOND OPERAND
        DAR,R3             DECIMAL ADJUST RESULT
        STRA,R3   RSLT     STORE RESULT
```

Example C

In the case of arithmetic with carry (WC = 1), it should be noted that the addition of the offset H'66' may generate a carry (if OPR1 = 99 and carry was set); this carry will be added during the addition of OPR2, giving an incorrect sum.

Multiple-Byte Operands/Result:

When using multiple-byte operands, linking of the bytes by means of the carry bit is required. Hence, arithmetic with carry must be performed (WC in PSL is set to 1). Because of the two successive additions (of the offset H'66' and of the second operand), the problem mentioned in the previous section can also arise here. Two straightforward solutions to this problem, listed below, are illustrated in the flowchart of Figure 3.

Method 1: In this method, each byte of the first operand is first increased by the offset H'66', after which addition of the second operand is performed. See Example D.

```
        CPSL      C            CLEAR CARRY
        PPSL      WC           ARITHMETIC WITH CARRY
        LODI,R3   LENG         LOAD INDEX REGISTER
ADD0    LODA,R0   OPR1,R3,-    FETCH BYTE OF OPERAND1
        ADDI,R0   H'66'        ADD OFFSET FOR BCD ADD
        STRA,R0   RSLT,R3      STORE INTERMEDIATE RESULT
        BRNR,R3   ADD0         BRANCH IF ALL BYTES NOT READY
        LODI,R3   LENG         LOAD INDEX REGISTER
ADD1    LODA,R0   RSLT,R3,-    FETCH BYTE OF INTERMEDIATE SUM
        ADDA,R0   OPR2,R3      ADD BYTE OF OPERAND2
        DAR,R0                 DECIMAL ADJUST RESULT
        STRA,R0   RSLT,R3      STORE RESULT
        BRNR,R3   ADD1         BRANCH IF ALL BYTES NOT READY
```

Example D

Method 2: In this method, the complete addition is handled on a byte-by-byte basis. This means that the true interbyte-carry must be saved and restored, and the carry must be cleared at the appropriate time. This can be performed by using one additional register to retain the interbyte-carry. See Example E.

The second method is faster and requires fewer bytes of code (24 versus 30) but requires an additional register.

The program listing of Figure 5 summarizes the basic BCD addition and subtraction routines.

ROUTINES FOR SIGNED INTEGER ARITHMETIC

There are several possible ways of representing signed decimal numbers. The best known are ten's complement notation and sign-magnitude notation. The sign-magnitude notation, illustrated in Figure 4, is used here because it is easy to interpret and lends itself to interfacing with peripherals. It is also simpler to use in multiplication, division, and in aligning and rounding routines. The numbers are stored in memory in the form of a sign followed by the absolute value of the number.

The length of the numbers is defined by the number of bytes (including the sign byte) they require. This parameter can be modified by changing the definition of LENG in the source program. Note that for clarity, each routine is written in a "stand-alone" form. If more than 1 routine is required in a program, considerable savings in the program space required can be realized by breaking out common operations as subroutines.

```
        CPSL      C           CLEAR CARRY
        PPSL      WC          ARITHMETIC/ROTATE WITH CARRY
        LODI,R3   LENG        LOAD INDEX REGISTER
        LODI,R1   0           CLEAR INTERBYTE-CARRY REGISTER
DADL    LODA,R0   OPR1,R3,-   FETCH BYTE OF OPERAND1
        ADDI,R0   H'66'       ADD OFFSET FOR BCD ADD
        RRR,R1                RESTORE INTERBYTE-CARRY TO CARRY
        ADDA,R0   OPR2,R3     ADD BYTE OF OPERAND2
        DAR,R0                DECIMAL ADJUST RESULT
        STRA,R0   RSLT,R3     STORE RESULT
        RRL,R1                SAVE INTERBYTE-CARRY IN R1, CLEAR C
        BRNR,R3   DADL        BRANCH IF NOT READY
```

Example E

GENERAL ADDITION FOR MULTIPLE-BYTE, UNSIGNED BCD NUMBERS

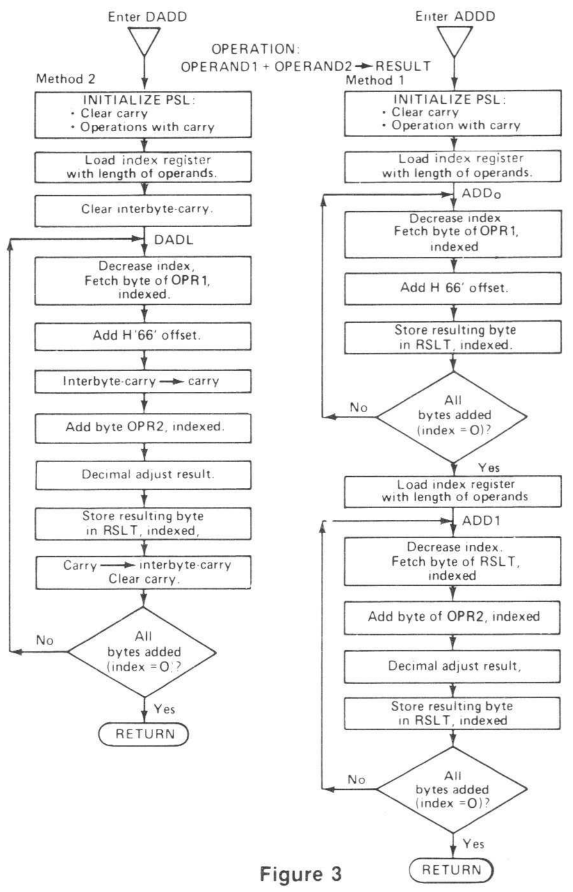

Enter DADD
Enter ADDD
OPERATION:
OPERAND1 + OPERAND2 → RESULT
Method 2
Method 1
INITIALIZE PSL: • Clear carry • Operations with carry
Load index register with length of operands.
Clear interbyte-carry.
DADL
Decrease index, Fetch byte of OPR1, indexed.
Add H'66' offset.
Interbyte-carry → carry
Add byte OPR2, indexed.
Decimal adjust result.
Store resulting byte in RSLT, indexed.
Carry → interbyte-carry Clear carry.
All bytes added (index = 0)?
No
Yes
RETURN
INITIALIZE PSL: • Clear carry • Operation with carry
Load index register with length of operands.
ADD0
Decrease index Fetch byte of OPR1, indexed
Add H'66' offset.
Store resulting byte in RSLT, indexed.
All bytes added (index = 0)?
No
Yes
Load index register with length of operands
ADD1
Decrease index. Fetch byte of RSLT, indexed
Add byte of OPR2, indexed
Decimal adjust result,
Store resulting byte in RSLT, indexed
All bytes added (index = 0)?
No
Yes
RETURN

Figure 3

SIGN-MAGNITUDE NOTATION

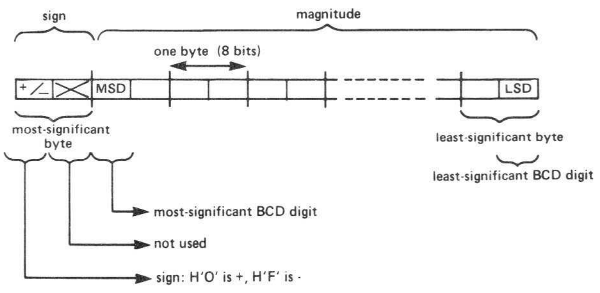


Figure 4

BCD ADDITION AND SUBTRACTION ROUTINES

```
TWIN ASSEMBLER VER 1.0                              PAGE 0001

LINE ADDR  OBJECT  E SOURCE

                 * PD760087
                 ************************************************
                 * DECIMAL ADDITION/SUBTRACTION FOR PACKED-BCD *
                 ************************************************
                 * OPERATION: OPERAND1 +/- OPERAND2 --> RESULT
                 * OPERAND1 IS IN: OPR1,OPR1+1,OPR1+2,ETC.
                 * OPERAND2 IS IN: OPR2,OPR2+1,OPR2+2,ETC.
                 * RESULT   IS IN: RSLT,RSLT+1,RSLT+2,ETC.
                 * OPR1,OPR2 AND RSLT ARE MOST-SIGNIFICANT BYTES.
                 * ALL NUMBERS ARE OF EQUAL LENGTH (IN BYTES).
                 * LENGTH IS DEFINED BY:  LENG
                 *
                 * DEFINITIONS OF SYMBOLS:
                 *
0000             R0   EQU     0             PROCESSOR REGISTERS
0001             R1   EQU     1
0002             R2   EQU     2
0003             R3   EQU     3
0008             WC   EQU     H'08'         PSL:  1=WITH, 0=WITHOUT CARRY
0001             C    EQU     H'01'               CARRY/BORROW
0003             UN   EQU     3             BRANCH CONDITION: UNCONDITIONAL
                 *
0005             LENG EQU     5             LENGTH OF OPERANDS/RESULT IN BYTES
                 *
0000                  ORG     H'700'        PARAMETERS
                 *
0700             OPR1 RES     LENG          OPERAND1
0705             OPR2 RES     LENG          OPERAND2
070A             RSLT RES     LENG          RESULT
                 *
070F                  ORG     H'450'
                 *
                 **************************************************
                 * ADDITION OF UNSIGNED, SINGLE-BYTE BCD NUMBERS *
                 **************************************************
                 * OPERATION: OPERAND1 + OPERAND2 --> RESULT
                 *
0450 0F0700      ADD  LODA,R3 OPR1          FETCH FIRST OPERAND
0453 8766             ADDI,R3 H'66'         ADD OFFSET FOR BCD ADD
0455 8F0705           ADDA,R3 OPR2          ADD SECOND OPERAND
0458 97               DAR,R3                DECIMAL ADJUST RESULT
0459 CF070A           STRA,R3 RSLT          STORE RESULT
                 *
                 *****************************************************
                 * SUBTRACTION OF UNSIGNED, SINGLE-BYTE BCD NUMBERS *
                 *****************************************************
                 * OPERATION: OPERAND1 - OPERAND2 --> RESULT
                 *
045C 0F0700      SUBT LODA,R3 OPR1          FETCH FIRST OPERAND
045F AF0705           SUBA,R3 OPR2          SUBTRACT SECOND OPERAND
0462 97               DAR,R3                DECIMAL ADJUST RESULT
0463 CF070A           STRA,R3 RSLT          STORE RESULT
                 *
```

```
TWIN ASSEMBLER VER 1.0                              PAGE 0002

LINE ADDR  OBJECT  E SOURCE

                 ****************************************************
                 *  ADDITION OF UNSIGNED MULTIPLE-BYTE BCD NUMBERS *
                 ****************************************************
                 * OPERATION: OPERAND1 + OPERAND2 --> RESULT
                 *
0466 7501        DADD CPSL    C             CLEAR CARRY
0468 7708             PPSL    WC            ARITHMETIC/ROTATE WITH CARRY
046A 0705             LODI,R3 LENG          LOAD INDEX REGISTER
046C 0500             LODI,R1 0             CLEAR INTERBYTE-CARRY
046E 0F4700      DADL LODA,R0 OPR1,R3,-     FETCH BYTE OF OPERAND1
0471 8466             ADDI,R0 H'66'         ADD OFFSET FOR BCD ADD
0473 51               RRR,R1                RESTORE INTERBYTE-CARRY TO C
0474 8F6705           ADDA,R0 OPR2,R3       ADD BYTE OF OPERAND2
0477 94               DAR,R0                DECIMAL ADJUST RESULT
0478 CF670A           STRA,R0 RSLT,R3       STORE RESULTING BYTE
047B D1               RRL,R1                SAVE INTERBYTE-CARRY IN R1, CLEAR C
047C 5B70             BRNR,R3 DADL          BRANCH IF NOT READY
                 *
                 ***************************************************
                 * ADDITION OF UNSIGNED MULTIPLE-BYTE BCD NUMBERS *
                 *       ALTERNATE METHOD                         *
                 ***************************************************
                 * OPERATION: OPERAND1 + OPERAND2 --> RESULT
                 *
047E 7501        ADDD CPSL    C             CLEAR CARRY
0480 7708             PPSL    WC            ARITHMETIC WITH CARRY
0482 0705             LODI,R3 LENG          LOAD INDEX REGISTER
0484 0F4700      ADD0 LODA,R0 OPR1,R3,-     FETCH BYTE OF OPERAND1
0487 8466             ADDI,R0 H'66'         ADD OFFSET FOR BCD-ADD
0489 CF670A           STRA,R0 RSLT,R3       STORE INTERMEDIATE RESULT
048C 5B76             BRNR,R3 ADD0          BRANCH IF ALL BYTES NOT READY
048E 0705             LODI,R3 LENG          LOAD INDEX REGISTER
0490 0F470A      ADD1 LODA,R0 RSLT,R3,-     FETCH BYTE OF INTERMEDIATE SUM
0493 8F6705           ADDA,R0 OPR2,R3       ADD BYTE OF OPERAND2
0496 94               DAR,R0                DECIMAL ADJUST RESULT
0497 CF670A           STRA,R0 RSLT,R3       STORE RESULT
049A 5B74             BRNR,R3 ADD1          BRANCH IF ALL BYTES NOT READY
                 *
                 *
                 ******************************************************
                 * SUBTRACTION OF UNSIGNED MULTIPLE-BYTE BCD NUMBERS *
                 ******************************************************
                 * OPERATION: OPERAND1 - OPERAND2 --> RESULT
                 *
049C 7709        DSUB PPSL    WC+C          ARITHMETIC WITH CARRY, CLEAR BORROW
049E 0705             LODI,R3 LENG          LOAD INDEX REGISTER
04A0 0F4700      DSUL LODA,R0 OPR1,R3,-     FETCH BYTE OF OPERAND1
04A3 AF6705           SUBA,R0 OPR2,R3       SUBTRACT BYTE OF OPERAND2
04A6 94               DAR,R0                DECIMAL ADJUST RESULT
04A7 CF670A           STRA,R0 RSLT,R3       STORE RESULTING BYTE
04AA 5B74             BRNR,R3 DSUL          BRANCH IF NOT READY
                 *
0000                  END     0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 5

Program Title
DECIMAL ADDITION/SUBTRACTION FOR SIGNED INTEGERS (PACKED BCD)

Function
Addition or subtraction of 2 decimal integers in sign-magnitude notation. Operands and result are of equal length, as defined by LENG.

OPERAND1 +/- OPERAND2►OPERAND2

Parameters
Input:
Length of numbers (in bytes) is defined by LENG.

OPR1, OPR1+1, OPR1+2, etc., contain augend or subtrahend.

OPR2, OPR2+1, OPR2+2, etc., contain addend or minuend.

Output:
OPR2, OPR2+1, OPR2+2, etc., contain sum or difference.

Overflow is detected.

OPERATION
Subtraction is performed by changing the sign of the second operand before entering the signed addition routine. Prior to adding or subtracting, the sign of the result must be determined. This requires a comparison of the magnitudes of both operands if they have opposite signs. In this case, the subtrahend and minuend for the operation are also designated by the comparison.

Refer to Figures 6 and 7 for flowchart and program listing.

FLOWCHART FOR DECIMAL ADDITION/SUBTRACTION FOR SIGNED INTEGERS

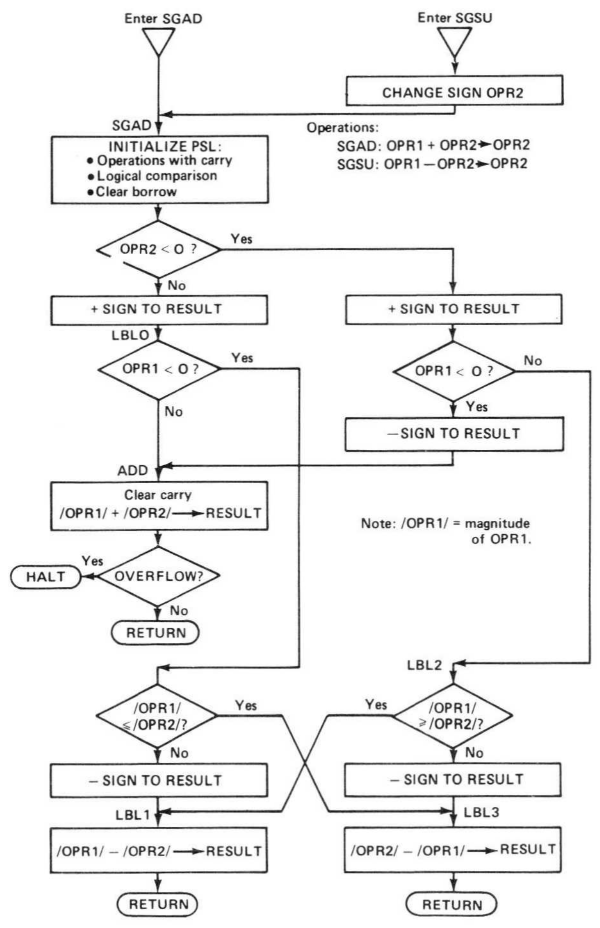


Figure 6

HARDWARE AFFECTED							
REGISTERS	R0 X	R1 X	R2	R3 X	R1′	R2′	R3′
PSU	F	II	SP				
PSL	CC X	IDC X	RS	WC X	OVF X	COM X	C X

RAM REQUIRED (BYTES): 2 X LENG

ROM REQUIRED (BYTES): 127

MAXIMUM SUBROUTINE NESTING LEVELS: 1

ASSEMBLER/COMPILER USED: TWIN VER 1.0

DECIMAL ADDITION/SUBTRACTION FOR SIGNED INTEGERS

```
TWIN ASSEMBLER VER 1.0                                 PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                 * PD760086
0002                 *******************************************************
0003                 * DECIMAL ADDITION/SUBTRACTION FOR SIGNED-INTEGERS    *
0004                 * NUMBERS ARE IN PACKED BCD, SIGN-MAGNITUDE NOTATION *
0005                 *******************************************************
0006                 * OPERATION:  OPERAND1 +/- OPERAND2 --> OPERAND2
0007                 * OPERAND1        IS IN:  OPR1,OPR1+1,OPR1+2, ETC.
0008                 * OPERAND2        IS IN:  OPR2,OPR2+1,OPR2+2, ETC.
0009                 * SUM/DIFFERENCE IS IN:   OPR2,OPR2+1,OPR2+2, ETC.
0010                 * OPERAND2 IS DESTROYED AFTER ADD/SUBTRACT.
0011                 * OPR1,OPR2 ARE MOST-SIGNIFICANT BYTES.
0012                 * LENGTH OF NUMBERS (IN BYTES) IS DEFINED BY: LENG
0013                 *    ALLOWED RANGE:  1 < LENG < 255.
0014                 * MS BYTE HOLDS SIGN INFORMATION:  H'00' FOR +, H 'F0' FOR -
0015                 *
0016                 * DEFINITIONS OF SYMBOLS:
0017                 *
0018 0000            R0   EQU    0        PROCESSOR REGISTERS
0019 0001            R1   EQU    1
0020 0002            R2   EQU    2
0021 0003            R3   EQU    3
0022 0008            WC   EQU    H'08'    PSL:  1=WITH, 0=WITHOUT CARRY
0023 0002            COM  EQU    H'02'          1=LOGIC, 0=ARITH.COMPARE
0024 0001            C    EQU    H'01'          CARRY/BORROW
0025 0000            Z    EQU    0        BRANCH CONDITION:  ZERO
0026 0002            N    EQU    2                           NEGATIVE
0027 0000            EQ   EQU    0                           EQUAL
0028 0001            GT   EQU    1                           GREATER THAN
0029 0002            LT   EQU    2                           LESS THAN
0030 0003            UN   EQU    3                           UNCONDITIONAL
0031                 *
0032                 * PARAMETERS *
0033                 *
0034 0005            LENG EQU    5        LENGTH OF OPERANDS (IN BYTES)
0035                 *
0036 0000                 ORG    H'700'
0037                 *
0038 0700            OPR1 RES    LENG     OPERAND1
0039 0705            OPR2 RES    LENG     OPERAND2/RESULT
0040                 *
0041 070A                 ORG    H'500'
0042                 *
0043                 ***********************************************************
0044                 * SUBROUTINE TO COMPARE OPERAND1 WITH OPERAND2 (UPDATE CC) *
0045                 ***********************************************************
0046 0500 0500       C012 LODI,R1 0       CLEAR R1; MS BITS ARE USED TO SAVE CC DATA
0047 0502 0704            LODI,R3 LENG-1  LOAD INDEX REG
0048 0504 0F6700     COM0 LODA,R0 OPR1,R3 FETCH BYTE OF OPERAND1
0049 0507 EF6705          COMA,R0 OPR2,R3 COMPARE WITH BYTE OF OPERAND2
0050 050A 1802            BCTR,EQ COM1    BRANCH IF EQUAL
0051 050C 13              SPSL            PSL TO R0
0052 050D C1              STRZ    R1      SAVE PSL IN R1
0053 050E FB74       COM1 BDRR,R3 COM0    BRANCH IF ALL BYTES NOT TESTED
0054 0510 01              LODZ    R1      UPDATE CC WITH STATUS COMPARE
0055 0511 17              RETC,UN         RETURN
0056                 *


TWIN ASSEMBLER VER 1.0                                 PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0057                 *
0058                 ***********************************
0059                 * SUBTRACTION FOR SIGNED INTEGERS *
0060                 ***********************************
0061 0512 0C0705     SGSU LODA,R0 OPR2    FETCH SIGN OF OPERAND2
0062 0515 24F0            EORI,R0 H'F0'   CHANGE SIGN
0063 0517 CC0705          STRA,R0 OPR2    RESTORE SIGN OF OPERAND2
0064                 ********************************
0065                 * ADDITION FOR SIGNED INTEGERS *
0066                 ********************************
0067 051A 770B       SGAD PPSL    WC+COM+C OPERATIONS WITH CARRY,
0068                 *                      LOGICAL COMPARE, CLEAR BORROW
0069 051C 20              EORZ    R0      CLEAR R0
0070 051D 0D0705          LODA,R1 OPR2    FETCH SIGN OF OPERAND2
0071 0520 CC0705          STRA,R0 OPR2    CLEAR SIGN OF OPERAND2 (=RESULT)
0072 0523 9A23            BCFR,N  LBL0    BRANCH IF OPR2 NOT NEGATIVE
0073 0525 0C0700          LODA,R0 OPR1    FETCH SIGN OF OPERAND1
0074 0528 9A3C            BCFR,N  LBL2    BRANCH IF OPR1 NOT NEGATIVE
0075 052A 04F0            LODI,R0 H'F0'   FETCH MINUS SIGN
0076 052C CC0705          STRA,R0 OPR2    STORE IN MS-BYTE RESULT
0077                 *
0078 052F 7501       ADD  CPSL    C       OPR1 + OPR2 --> OPR2,
0079                 *                      CLEAR CARRY.
0080 0531 0704            LODI,R3 LENG-1  LOAD INDEX REGISTER
0081 0533 0500            LODI,R1 0       CLEAR INTERBYTE-CARRY
0082 0535 0F6700     ADD0 LODA,R0 OPR1,R3 FETCH BYTE OF OPERAND1
0083 0538 8466            ADDI,R0 H'66'   ADD OFFSET
0084 053A 51              RRR,R1          INTERBYTE-CARRY TO CARRY
0085 053B 8F6705          ADDA,R0 OPR2,R3 ADD BYTE OF OPERAND2
0086 053E 94              DAR,R0          DECIMAL ADJUST RESULT
0087 053F CF6705          STRA,R0 OPR2,R3 STORE RESULTING BYTE
0088 0542 D1              RRL,R1          CARRY (=INTERBYTE-CARRY) TO R1,
0089                 *                      CLEAR CARRY.
0090 0543 FB70            BDRR,R3 ADD0    BRANCH IF NOT READY
0091 0545 9838            BCFR,Z  OVFL    BRANCH IF OVERFLOW
0092 0547 17              RETC,UN         RETURN
0093                 *


TWIN ASSEMBLER VER 1.0                                 PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0095 0548 0C0700     LBL0 LODA,R0 OPR1    FETCH SIGN OF OPERAND1
0096 054B 9A62            BCFR,N  ADD     BRANCH IF OPR1 NOT NEGATIVE
0097 054D 3F0500          BSTA,UN C012    COMPARE OPR1 WITH OPR2,
0098                 *                      (MAGNITUDES ONLY)
0099 0550 991E            BCFR,GT LBL3    BRANCH IF OPR1 < OR = TO OPR2
0100 0552 04F0            LODI,R0 H'F0'   FETCH MINUS SIGN
0101 0554 CC0705          STRA,R0 OPR2    STORE IN MS-BYTE RESULT
0102                 *
0103                 *                    OPR1 - OPR2 --> OPR2
0104 0557 0704       LBL1 LODI,R3 LENG-1  LOAD INDEX REGISTER
0105 0559 0F6700     SU12 LODA,R0 OPR1,R3 FETCH BYTE OF OPERAND1
0106 055C AF6705          SUBA,R0 OPR2,R3 SUBTRACT BYTE OF OPERAND2
0107 055F 94              DAR,R0          DECIMAL ADJUST RESULT
0108 0560 CF6705          STRA,R0 OPR2,R3 STORE RESULTING BYTE IN OPR2
0109 0563 FB74            BDRR,R3 SU12    BRANCH IF NOT READY
0110 0565 17              RETC,UN         RETURN
0111                 *
0112 0566 3F0500     LBL2 BSTA,UN C012    COMPARE OPR1 WITH OPR2,
0113                 *                      (MAGNITUDES ONLY)
0114 0569 9A6C            BCFR,LT LBL1    BRANCH IF OPR1 > OR = OPR2
0115 056B 04F0            LODI,R0 H'F0'   FETCH MINUS SIGN
0116 056D CC0705          STRA,R0 OPR2    STORE IN MS-BYTE OF RESULT
0117                 *
0118                 *                    OPR2 - OPR1 --> OPR2
0119 0570 0704       LBL3 LODI,R3 LENG-1  LOAD INDEX REGISTER
0120 0572 0F6705     SU21 LODA,R0 OPR2,R3 FETCH BYTE OF OPERAND2
0121 0575 AF6700          SUBA,R0 OPR1,R3 SUBTRACT BYTE OF OPERAND1
0122 0578 94              DAR,R0          DECIMAL ADJUST RESULT
0123 0579 CF6705          STRA,R0 OPR2,R3 STORE RESULTING BYTE
0124 057C FB74            BDRR,R3 SU21    BRANCH IF NOT READY
0125 057E 17              RETC,UN         RETURN
0126                 *
0127 057F 40         OVFL HALT            ARITHMETIC OVERFLOW
0128                 *
0129 0000                 END 0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 7

Program Title

DECIMAL MULTIPLICATION FOR SIGNED INTEGERS (PACKED BCD)

FUNCTION

Multiplication of 2 decimal integers in sign-magnitude notation.

Multiplicand, multiplier, and product are of equal length as defined by LENG.

MULTIPLICAND X MULTIPLIER ➤ MULTIPLIER

Parameters

Input:

Length of numbers (in bytes) is defined by LENG.

MPLC, MPLC+1, MPLC+2, etc., contain multiplicand.

MPLR, MPLR+1, MPLR+2, etc., contain multiplier.

Output:

MPLR, MPLR+1, MPLR+2, etc., contain product.

Multiplier is destroyed after multiplication.

Overflow is detected.

OPERATION

Prior to the multiplication algorithm (which is an unsigned operation), the sign of the product is determined. The multiplication gives a double-length result, of which only the least-significant half is retained as the product. If the most-significant half is unequal to zero, an overflow is detected. A "minus-zero" is excluded by means of a test for zero product.

Refer to Figures 8 and 9 for flowchart and program listing.

FLOWCHART FOR DECIMAL MULTIPLICATION OF SIGNED INTEGERS (PACKED BCD)

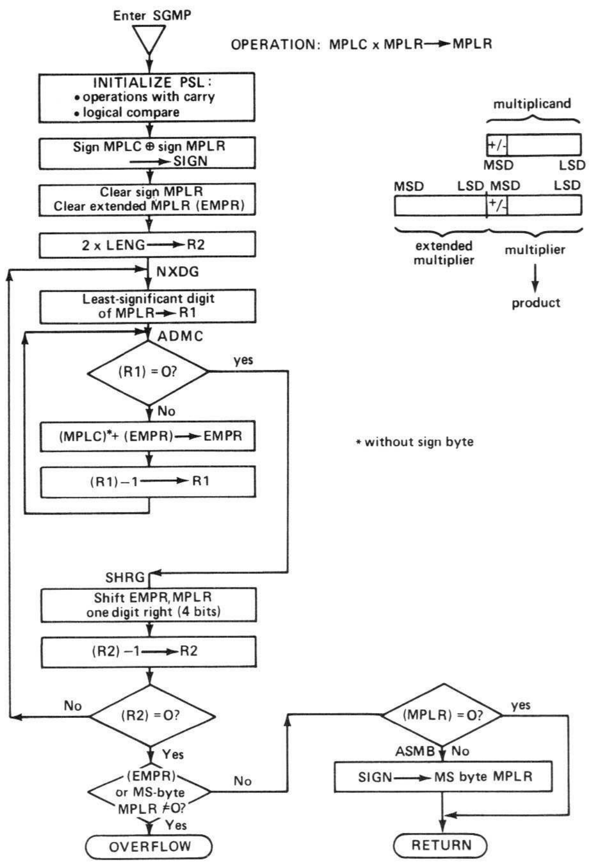


Figure 8

HARDWARE AFFECTED							
REGISTERS	R0 X	R1 X	R2 X	R3 X	R1′	R2′	R3′
PSU	F	II	SP				
PSL	CC X	IDC X	RS	WC X	OVF X	COM X	C X

RAM REQUIRED (BYTES): (3 X LENG) + 1

ROM REQUIRED (BYTES): 111

MAXIMUM SUBROUTINE NESTING LEVELS: None

ASSEMBLER/COMPILER USED: TWIN VER 1.0

DECIMAL MULTIPLICATION FOR SIGNED INTEGERS

```
TWIN ASSEMBLER VER 1.0                                   PAGE 0001

LINE ADDR  OBJECT  E SOURCE

                       * PD760085
                       *********************************************************
                       * DECIMAL MULTIPLICATION FOR SIGNED-INTEGERS.           *
                       * NUMBERS ARE IN PACKED BCD, SIGN-MAGNITUDE NOTATION *
                       *********************************************************
                       * OPERATION: MULTIPLICAND X MULTIPLIER --> MULTIPLIER
                       * MULTIPLICAND IS IN:  MPLC,MPLC+1,MPLC+2, ETC.
                       * MULTIPLIER   IS IN:  MPLR,MPLR+1,MPLR+2, ETC.
                       * PRODUCT      IS IN:  MPLR,MPLR+1,MPLR+2, ETC.
                       * MULTIPLIER IS DESTROYED AFTER MULTIPLICATION.
                       * MPLC,MPLR ARE MOST-SIGNIFICANT BYTES.
                       * LENGTH OF NUMBERS (IN BYTES) IS DEFINED BY: LENG
                       *    ALLOWED RANGE:  1 < LENG < 65.
                       * MS BYTE REPRESENTS SIGN: H'00' FOR +, H'F0' FOR -
                       *
                       * DEFINITIONS OF SYMBOLS:
                       *
     0000              R0   EQU     0        PROCESSOR-REGISTERS
     0001              R1   EQU     1
     0002              R2   EQU     2
     0003              R3   EQU     3
     0008              WC   EQU     H'08'    PSL:  1=WITH, 0=WITHOUT CARRY
     0002              COM  EQU     H'02'          1=LOGIC, 0=ARITH.COMPARE
     0001              C    EQU     H'01'          CARRY/BORROW
     0000              Z    EQU     0        BRANCH CONDITION:  ZERO
     0003              UN   EQU     3                           UNCONDITIONAL
                       *
                       * PARAMETERS *
                       *
     0005              LENG EQU     5        LENGTH OF OPERANDS (BYTES)
                       *
     0000                   ORG     H'700'
                       *
     0700              MPLC RES     LENG     MULTIPLICAND
     0705              EMPR RES     LENG     EXTENDED MULTIPLIER
     070A              MPLR RES     LENG     MULTIPLIER
                       * NOTE: EMPR AND MPLR MUST BE IN SUCCESSIVE
                       *      RAM LOCATIONS FOR DOUBLE-LENGTH SHIFT.
     070F              SIGN RES     1        TEMPORARY SIGN
                       *
                       *                     ***************************
     0710                   ORG     H'500'   * MULTIPLICATION PROGRAM *
                       *                     ***************************
                       *
     0500 770A         SGMP PPSL    WC+COM   OPERATIONS WITH CARRY, LOGICAL COMPARE
     0502 0C0700            LODA,R0 MPLC     FETCH SIGN MULTIPLICAND
     0505 2C070A            EORA,R0 MPLR     TAKE EX-OR WITH SIGN MULTIPLIER
     0508 CC070F            STRA,R0 SIGN     SAVE PRODUCT SIGN IN SIGN
     050B 20                EORZ    R0       CLEAR R0
     050C 0706              LODI,R3 LENG+1   LOAD INDEX REGISTER
     050E CF4705       CLEM STRA,R0 EMPR,R3,- CLEAR EXTENDED MULTIPLIER AND SIGN OF MULTIPLIER
     0511 5B7B              BRNR,R3 CLEM     BRANCH IF NOT DONE
     0513 060A              LODI,R2 LENG+LENG LOAD LOOP COUNTER WITH NUMBER OF DIGITS
     0515 0D070E       NXDG LODA,R1 MPLR+LENG-1  FETCH LS-BYTE MULTIPLIER
     0518 450F              ANDI,R1 H'0F'    CLEAR MS-DIGIT
     051A 1826              BCTR,Z  SHRG     BRANCH IF LS-DIGIT IS ZERO
```

```
TWIN ASSEMBLER VER 1.0                                   PAGE 0002

LINE ADDR  OBJECT  E SOURCE

                       *                     ADD MULTIPLICAND TO EXTENDED
                       *                       MULTIPLIER WITHOUT SIGN
     051C 7501         ADMC CPSL    C        CLEAR CARRY
     051E 0704              LODI,R3 LENG-1   LOAD INDEX REGISTER
     0520 0F6705       ADM0 LODA,R0 EMPR,R3  FETCH BYTE OF EXTENDED MULTIPLIER
     0523 8466              ADDI,R0 H'66'    ADD OFFSET FOR DECIMAL ADJUST
     0525 CF6705            STRA,R0 EMPR,R3  RESTORE INTERMEDIATE SUM
     0528 FB76              BDRR,R3 ADM0     BRANCH IF ALL BYTES NOT READY
     052A 0704              LODI,R3 LENG-1   LOAD INDEX REGISTER
     052C 0F6705       ADM1 LODA,R0 EMPR,R3  FETCH BYTE OF INTERMEDIATE SUM
     052F 8F6700            ADDA,R0 MPLC,R3  ADD BYTE OF MULTIPLICAND
     0532 94                DAR,R0           DECIMAL ADJUST RESULT
     0533 CF6705            STRA,R0 EMPR,R3  STORE RESULTING BYTE
     0536 FB74              BDRR,R3 ADM1     BRANCH IF NOT READY
     0538 0C0705            LODA,R0 EMPR     FETCH MS-BYTE EXTENDED MULTIPLIER
     053B 8400              ADDI,R0 0        ADD CARRY
     053D CC0705            STRA,R0 EMPR     RESTORE MS-BYTE EXTENDED MULTIPLIER
     0540 F95A              BDRR,R1 ADMC     BRANCH IF NOT READY WITH DIGIT
                       *
                       *                     SHIFT EMPR AND MPLR ONE DIGIT
                       *                       POSITION RIGHT (4 BITS)
     0542 0504         SHRG LODI,R1 4        LOAD LOOP COUNTER
     0544 7501         SHR0 CPSL    C        CLEAR CARRY
     0546 07F6              LODI,R3 -LENG-LENG  LOAD INDEX REGISTER
     0548 0F660F       SHR1 LODA,R0 EMPR-256+LENG+LENG,R3  FETCH BYTE OF EXTENDED MULTIPLIER
     054B 50                RRR,R0           ROTATE RIGHT WITH CARRY
     054C CF660F            STRA,R0 EMPR-256+LENG+LENG,R3  RESTORE BYTE
     054F DB77              BIRR,R3 SHR1     BRANCH IF ALL NOT SHIFTED
     0551 F971              BDRR,R1 SHR0     BRANCH IF 4 BITS NOT SHIFTED
                       *
     0553 FA40              BDRR,R2 NXDG     BRANCH IF ALL DIGITS NOT READY
                       *
                       *                     TEST FOR OVERFLOW; OVERFLOW IF
                       *                       (EMPR) OR MS-BYTE MPLR ARE UNEQUAL TO ZERO
     0555 0706              LODI,R3 LENG+1   LOAD INDEX REGISTER
     0557 0F4705       TOVF LODA,R0 EMPR,R3,- FETCH BYTE OF EXTENDED MPLR
     055A 9813              BCFR,Z  OVFL     BRANCH IF NOT ZERO
     055C 5B79              BRNR,R3 TOVF     BRANCH IF ALL BYTES NOT TESTED
     055E 0704              LODI,R3 LENG-1   TEST IF PRODUCT=0; LOAD INDEX.
     0560 0F670A       TZER LODA,R0 MPLR,R3  FETCH BYTE OF PRODUCT
     0563 9803              BCFR,Z  ASMB     BRANCH IF NOT ZERO
     0565 FB79              BDRR,R3 TZER     BRANCH IF ALL BYTES NOT TESTED
     0567 17                RETC,UN          PRODUCT=0; SIGN REMAINS ZERO.
                       *
     0568 0C070F       ASMB LODA,R0 SIGN     FETCH PRODUCT SIGN
     056B CC070A            STRA,R0 MPLR     STORE IN MS-BYTE MPLR
     056E 17                RETC,UN          RETURN
                       *
     056F 40           OVFL HALT             ARITHMETIC OVERFLOW
                       *
     0000                   END     0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 9

Program Title
DECIMAL DIVISION FOR SIGNED INTEGERS (PACKED BCD)

Function
Division of 2 decimal integers in sign-magnitude notation.

Dividend, divisor, quotient, and remainder are of equal length as defined by LENG.

DIVIDEND: DIVISOR ➞ DIVIDEND, REMAINDER

Parameters

Input:
Length of numbers (in bytes) is defined by LENG.

DVDN, DVDN+1, DVDN+2, etc., contain dividend.

DVSR, DVSR+1, DVSR+2, etc., contain divisor.

Output:
DVDN, DVDN+1, DVDN+2, etc., contain quotient.

RMDR, RMDR+1, RMDR+2, etc., contain remainder.

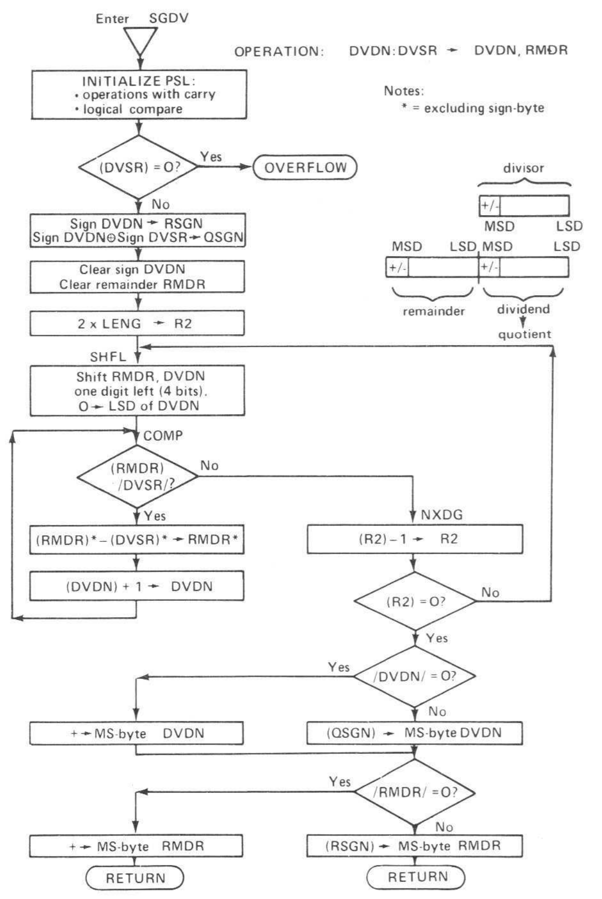
Dividend is destroyed after division.

Overflow is detected.

OPERATION:
Prior to the division, which in itself is an unsigned operation, the signs of the remainder and quotient are determined. Because the division can result in a zero quotient and/or remainder, the possibility of a "minus zero" is excluded by tests. If the divisor is zero, overflow is detected.

Refer to Figures 10 and 11 for flowchart and program listing.

FLOWCHART FOR DECIMAL DIVISION OF SIGNED INTEGERS



/OPR/ = magnitude operand

Figure 10

HARDWARE AFFECTED							
REGISTERS	R0 X	R1 X	R2 X	R3 X	R1′	R2′	R3′
PSU	F	II	SP				
PSL	CC X	IDC X	RS	WC X	OVF X	COM X	C X

RAM REQUIRED (BYTES): (3 x LENG) + 4

ROM REQUIRED (BYTES): 144

MAXIMUM SUBROUTINE NESTING LEVELS: 1

ASSEMBLER/COMPILER USED: TWIN VER 1.0

DECIMAL DIVISION FOR SIGNED INTEGERS

```
TWIN ASSEMBLER VER 1.0                                PAGE 0001

LINE ADDR  OBJECT  E SOURCE

                  * PD760084
                  **********************************************************
                  * DECIMAL DIVISION FOR SIGNED INTEGERS                   *
                  * NUMBERS ARE IN PACKED BCD, SIGN-MAGNITUDE NOTATION *
                  **********************************************************
                  * OPERATION:
                  *    DIVIDEND : DIVISOR --> DIVIDEND, REMAINDER
                  * DIVIDEND  IS IN: DVDN,DVDN+1,DVDN+2, ETC.
                  * DIVISOR   IS IN: DVSR,DVSR+1,DVSR+2, ETC.
                  * QUOTIENT  IS IN: DVDN,DVDN+1,DVDN+2, ETC.
                  * REMAINDER IS IN: RMDR,RMDR+1,RMDR+2, ETC.
                  * DIVIDEND IS DESTROYED AFTER DIVISION.
                  * DVDN,DVSR AND RMDR ARE MOST-SIGNIFICANT BYTES.
                  * LENGTH OF NUMBERS (IN BYTES) IS DEFINED BY: LENG.
                  *    ALLOWED RANGE:  1 < LENG < 65.
                  * MS-BYTE HOLDS SIGN INFORMATION: H'00' FOR +, H'F0' FOR -
                  *
                  * DEFINITIONS OF SYMBOLS:
                  *
0000              R0   EQU    0        PROCESSOR REGISTERS
0001              R1   EQU    1
0002              R2   EQU    2
0003              R3   EQU    3
0008              WC   EQU    H'08'    PSL:  1=WITH, 0=WITHOUT CARRY
0002              COM  EQU    H'02'          1=LOGIC, 0=ARITH.COMPARE
0001              C    EQU    H'01'          CARRY/BORROW
0000              Z    EQU    0        BRANCH COND.:  ZERO
0001              P    EQU    1                       POSITIVE
0002              N    EQU    2                       NEGATIVE
0000              EQ   EQU    0                       EQUAL
0002              LT   EQU    2                       LESS THAN
0003              UN   EQU    3                       UNCONDITIONAL
                  *
                  * PARAMETERS *
                  *
0005              LENG EQU    5        LENGTH OF OPERANDS (IN BYTES)
                  *
0000                   ORG    H'700'
                  *
0700              RMDR RES    LENG     REMAINDER
0705              DVDN RES    LENG     DIVIDEND
                  * NOTE: RMDR AND DVDN MUST BE IN SUCCESSIVE
                  *       RAM LOCATIONS, BECAUSE OF DOUBLE-LENGTH SHIFT.
070A              DVSR RES    LENG     DIVISOR
070F              TEMP RES    2        TEMPORARY STORAGE FOR ADDRESS
0711              QSGN RES    1        QUOTIENT SIGN
0712              RSGN RES    1        REMAINDER SIGN
                  *


TWIN ASSEMBLER VER 1.0                                PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0713                   ORG    H'500'
                  *
                  ****************************************
                  * SUBROUTINE TO TEST OPERAND FOR ZERO *
                  ****************************************
                  * OPERAND ADDRESS MUST BE IN R0,R1 (HIGH,LOW ADDR.)
                  * ALL BYTES, EXCEPT MS-BYTE (=SIGN) ARE TESTED.
                  * CONDITION CODE BECOMES 00 IF OPERAND WAS ZERO.
                  *
0500 CC070F       TZER STRA,R0 TEMP       SAVE OPERAND ADDRESS
0503 CD0710            STRA,R1 TEMP+1
0506 0704              LODI,R3 LENG-1     LOAD INDEX REGISTER
0508 0FE70F       TZE0 LODA,R0 *TEMP,R3   FETCH BYTE OF OPERAND
050B 15                RETC,P             RETURN IF POSITIVE (CC=01)
050C 16                RETC,N             RETURN IF NEGATIVE (CC=10)
050D FB79              BDRR,R3 TZE0       BRANCH IF ALL NOT TESTED
050F 17                RETC,UN            RETURN WITH CC=00
                  *
                  *                      ********************
                  *                      * DIVISION PROGRAM *
                  *                      ********************
                  *
0510 770A         SGDV PPSL    WC+COM    OPERATIONS WITH CARRY,
                  *                         LOGICAL COMPARISON.
0512 0407              LODI,R0 <DVSR     HIGH-ADDRESS DIVISOR TO R0
0514 050A              LODI,R1 >DVSR     LOW- ADDRESS DIVISOR TO R1
0516 3F0500            BSTA,UN TZER      TEST DIVISOR FOR ZERO
0519 1C0595            BCTA,Z  OVFL      BRANCH IF ZERO
051C 0C0705            LODA,R0 DVDN      FETCH SIGN DIVIDEND
051F CC0712            STRA,R0 RSGN      SAVE IN REMAINDER SIGN
0522 2C070A            EORA,R0 DVSR      TAKE EX-OR WITH DVSR SIGN
0525 CC0711            STRA,R0 QSGN      SAVE IN QUOTIENT SIGN
0528 20                EORZ    R0        CLEAR R0
0529 0706              LODI,R3 LENG+1    LOAD INDEX REGISTER
052B CF4700       CLRM STRA,R0 RMDR,R3,- CLEAR REMAINDER AND SIGN DVDN
052E 5B7B              BRNR,R3 CLRM      BRANCH IF NOT DONE
                  *
0530 060A              LODI,R2 LENG+LENG NUMBER OF DIGITS TO LOOP COUNTER
                  *
                  *                      SHIFT RMDR/DVDN 4 BITS LEFT
                  *                         INSERTING ZEROES IN LS-BITS
0532 0504         SHFL LODI,R1 4         LOAD BIT COUNTER
0534 7501         SHF0 CPSL    C         CLEAR CARRY
0536 070A              LODI,R3 LENG+LENG LOAD INDEX REGISTER
0538 0F4700       SHF1 LODA,R0 RMDR,R3,- FETCH BYTE OF RMDR/DVDN
053B D0                RRL,R0            ROTATE LEFT WITH CARRY
053C CF6700            STRA,R0 RMDR,R3   RESTORE SHIFTED BYTE
053F 5B77              BRNR,R3 SHF1      BRANCH IF ALL NOT SHIFTED
0541 F971              BDRR,R1 SHF0      BRANCH IF 4 BITS NOT SHIFTED


TWIN ASSEMBLER VER 1.0                                PAGE 0003

LINE ADDR  OBJECT  E SOURCE

                  *                      COMPARE RMDR AND DVSR TO TEST
                  *                         IF SUBTRACTION IS POSSIBLE
0543 0500         COMP LODI,R1 0         CLEAR R1, MS-BIT OF R1 BECOMES
                  *                         1 FOR RMDR < DVSR
0545 0704              LODI,R3 LENG-1    LOAD INDEX REGISTER
0547 0F6700       COM0 LODA,R0 RMDR,R3   FETCH BYTE OF REMAINDER
054A EF670A            COMA,R0 DVSR,R3   COMPARE WITH BYTE OF DIVISOR
054D 1802              BCTR,EQ COM1      BRANCH IF EQUAL
054F 13                SPSL              PSL TO R0
0550 C1                STRZ    R1        SAVE PSL IN R1
0551 FB74         COM1 BDRR,R3 COM0      BRANCH IF ALL BYTES NOT TESTED
0553 01                LODZ    R1        FETCH STATUS OF COMPARISON
0554 1A1A              BCTR,LT NXDG      BRANCH IF RMDR < DVSR
                  *
                  *                      SUBTRACT DIVISOR FROM
                  *                         REMAINDER WITHOUT MS-BYTES.
0556 7701              PPSL    C         CLEAR BORROW
0558 0704              LODI,R3 LENG-1    LOAD INDEX REGISTER
055A 0F6700       SURD LODA,R0 RMDR,R3   FETCH BYTE OF REMAINDER
055D AF670A            SUBA,R0 DVSR,R3   SUBTRACT BYTE OF DIVISOR
0560 94                DAR,R0            DECIMAL ADJUST RESULT
0561 CF6700            STRA,R0 RMDR,R3   RESTORE IN REMAINDER
0564 FB74              BDRR,R3 SURD      BRANCH IF NOT READY
                  *
0566 0C0709            LODA,R0 DVDN+LENG-1  FETCH LS-BYTE QUOTIENT
0569 D800              BIRR,R0 $+2       INCREASE R0
056B CC0709            STRA,R0 DVDN+LENG-1 RESTORE INCREMENTED QUOTIENT
056E 1B53              BCTR,UN COMP      BRANCH FOR NEXT COMPARISON
                  *
0570 FA40         NXDG BDRR,R2 SHFL      BRANCH IF DIVISION NOT READY
0572 0E0711            LODA,R2 QSGN      FETCH SIGN QUOTIENT
0575 0407              LODI,R0 <DVDN     HIGH-ADDRESS QUOTIENT TO R0
0577 0505              LODI,R1 >DVDN     LOW- ADDRESS QUOTIENT TO R1
0579 3F0500            BSTA,UN TZER      TEST IF QUOTIENT IS ZERO
057C 9802              BCFR,Z  STQS      BRANCH IF NOT ZERO
057E 0600              LODI,R2 0         CLEAR R0
0580 CE0705       STQS STRA,R2 DVDN      STORE SIGN IN MS-BYTE DVDN
0583 0E0712            LODA,R2 RSGN      FETCH REMAINDER SIGN
0586 0407              LODI,R0 <RMDR     HIGH-ADDRESS REMAINDER TO R0
0588 0500              LODI,R1 >RMDR     LOW- ADDRESS REMAINDER TO R1
058A 3F0500            BSTA,UN TZER      TEST IF REMAINDER IS ZERO
058D 9802              BCFR,Z  STRS      BRANCH IF NOT ZERO
058F 0600              LODI,R2 0         CLEAR R2
0591 CE0700       STRS STRA,R2 RMDR      STORE SIGN IN MS-BYTE RMDR
0594 17                RETC,UN           RETURN
                  *
0595 40           OVFL HALT              OVERFLOW LOCATION
                  *
0000                   END     0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 11

ROUTINES FOR SIGNED FIXED-POINT ARITHMETIC

As illustrated in Figure 12, the numbers used in these arithmetic routines are in sign-magnitude notation with decimal point indication. The latter gives the number of decimals, and has a minimum of zero and a maximum limit of 15 or the number of digits, whichever is smaller.

The length of the numbers is defined by the number of bytes (including the sign byte) they require. This parameter can be modified by changing the definition of LENG in the source program. Note that for clarity, each routine is written in a "stand-alone" form. If more than one routine is required in a program, considerable savings in the program space required can be realized by breaking out common operations as subroutines.

Program Title

ALIGNMENT SUBROUTINE FOR FIXED-POINT NUMBERS

Function

Aligns a fixed-point number to the decimal point position indicated by the contents of register DPNT. Performs rounding as specified.

Parameters

Input:
R0 contains the high address of the operand.

R1 contains the low address of the operand.

DPNT contains the required decimal point.

ROUN contains the rounding constant: (ROUN) = H'00' for no rounding; (ROUN) = H'05' for 5/4 rounding; and (ROUN) = H'09' for round-up.

Prior to entry, WC in PSL must be 1.

Length of operand (in bytes) is defined by LENG.

Output:
Aligned operand; rounded if specified.

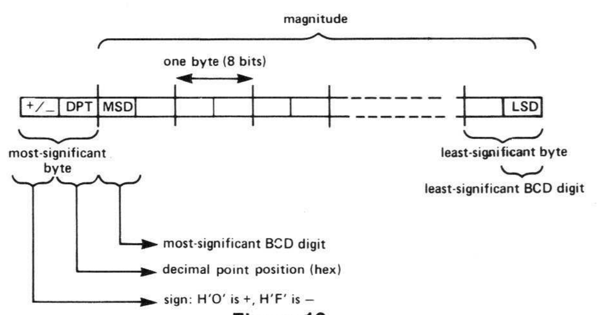
Alignment overflow is detected.

Operation:

The results of a fixed-point operation must be aligned to the required number of decimals. By means of this aligning routine, the numbers are shifted left or right, if necessary, until the appropriate decimal point position is obtained. This position must have previously been stored in a register designated DPNT. During left alignment, overflow can occur if a non-zero digit drops out of the most-significant digit position. During aligning it is also possible to perform rounding of the operand. This is done by adding a rounding digit to the most-significant digit of the decimals which are truncated by the right alignment. This rounding digit must have previously been stored in register ROUN and gives the possibilities listed above. Since rounding can only be performed during right alignment, the required decimal point position must be less than 15 if rounding is desired. If the aligned result is minus zero, the sign is changed.

Refer to Figures 13 and 14 for flowchart and program listing.

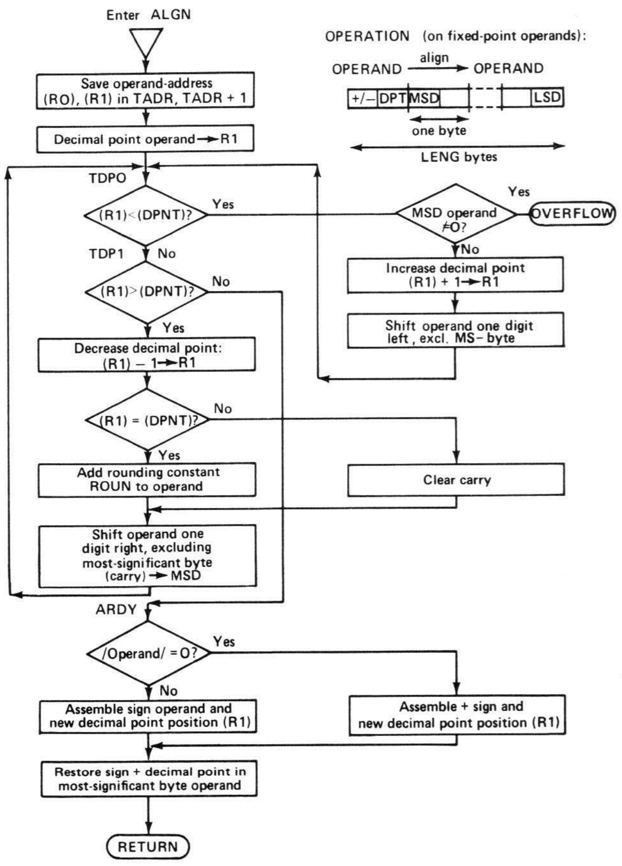
FIXED-POINT, SIGN-MAGNITUDE NOTATION



magnitude
one byte (8 bits)
+/_
DPT
MSD
LSD
most-significant byte
least-significant byte
least-significant BCD digit
most-significant BCD digit
decimal point position (hex)
sign: H'O' is +, H'F' is –

Figure 12

FLOWCHART FOR FIXED-POINT ALIGNMENT SUBROUTINE



Enter ALGN
Save operand-address (RO), (R1) in TADR, TADR + 1
Decimal point operand→R1
TDPO
(R1)<(DPNT)?
Yes
No
TDP1
(R1)>(DPNT)?
No
Yes
Decrease decimal point: (R1) – 1→R1
(R1) = (DPNT)?
No
Yes
Add rounding constant ROUN to operand
Clear carry
Shift operand one digit right, excluding most-significant byte (carry)→MSD
ARDY
/Operand/ = O?
Yes
No
Assemble sign operand and new decimal point position (R1)
Assemble + sign and new decimal point position (R1)
Restore sign + decimal point in most-significant byte operand
RETURN
OPERATION (on fixed-point operands):
OPERAND → OPERAND
align
+/–
DPT
MSD
LSD
one byte
LENG bytes
MSD operand ≠O?
Yes
OVERFLOW
No
Increase decimal point (R1) + 1→R1
Shift operand one digit left, excl. MS-byte

Figure 13

HARDWARE AFFECTED							
REGISTERS	R0 X	R1 X	R2 X	R3 X	R1′	R2′	R3′
PSU	F	II	SP				
PSL	CC X	IDC X	RS	WC	OVF X	COM	C X

RAM REQUIRED (BYTES): 4

ROM REQUIRED (BYTES): 120

MAXIMUM SUBROUTINE NESTING LEVELS: None

ASSEMBLER/COMPILER USED: TWIN VER 1.0

FIXED-POINT ALIGNMENT SUBROUTINE

```
TWIN ASSEMBLER VER 1.0                                    PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                 * PD760080
0002                 ************************************
0003                 * FIXED POINT ALIGNMENT SUBROUTINE *
0004                 ************************************
0005                 *
0006                 * DEFINITIONS OF SYMBOLS:
0007                 *
0008 0000            R0   EQU    0       PROCESSOR REGISTERS
0009 0001            R1   EQU    1
0010 0002            R2   EQU    2
0011 0003            R3   EQU    3
0012 0008            WC   EQU    H'08'   PSL:  1=WITH, 0=WITHOUT CARRY
0013 0001            C    EQU    H'01'         CARRY/BORROW
0014 0000            Z    EQU    0       BRANCH COND.:  ZERO
0015 0000            EQ   EQU    0                      EQUAL
0016 0001            GT   EQU    1                      GREATER THAN
0017 0002            LT   EQU    2                      LESS THAN
0018 0003            UN   EQU    3                      UNCONDITIONAL
0019                 *
0020                 * PARAMETERS *
0021                 *
0022 0005            LENG EQU    5       LENGTH OF OPERAND (BYTES)
0023                 *
0024 0000                ORG    H'440'
0025                 *
0026 0440            DPNT RES    1       REQUIRED DECIMAL POINT (0 THROUGH 15)
0027 0441            ROUN RES    1       ROUNDING CONSTANT (0,5 OR 9)
0028 0442            TADR RES    2       TEMPORARY STORAGE FOR ADDRESS
0029                 *
0030 0444                ORG    H'450'   START OF SUBROUTINE
0031                 *
0032                 * OPERAND IS ALIGNED TO DECIMAL POINT POSITION AS
0033                 * INDICATED BY REGISTER DPNT.
0034                 * ROUNDING IS PERFORMED UNDER FOLLOWING CONDITIONS:
0035                 *    (ROUN) CONTAINS H'00' FOR NO ROUNDING
0036                 *    (ROUN) CONTAINS H'05' FOR 5/4 ROUNDING
0037                 *    (ROUN) CONTAINS H'09' FOR ROUND-UP
0038                 * (DPNT) MUST BE < 15 IF ROUNDING IS REQUIRED.
0039                 * ALIGNMENT-OVERFLOW IS DETECTED.
0040                 * PRIOR TO ENTRY: WC IN PSL MUST BE 1.
0041                 *                R0 CONTAINS HIGH-ADDR OF OPERAND
0042                 *                R1 CONTAINS LOW- ADDR OF OPERAND
0043                 *                DPNT CONTAINS DECIMAL POINT
0044                 *                ROUN CONTAINS ROUNDING CONSTANT
0045                 *
0046 0450 CC0442     ALGN STRA,R0 TADR      SAVE HI-ADDRESS OF OPERAND
0047 0453 CD0443          STRA,R1 TADR+1    SAVE LO-ADDRESS OF OPERAND
0048 0456 0D8442          LODA,R1 *TADR     FETCH MS-BYTE OF OPERAND
0049 0459 450F            ANDI,R1 H'0F'     REMOVE SIGN, KEEP DECIMAL POINT
0050 045B 0604       TDP0 LODI,R2 4         LOAD LOOP COUNTER
0051 045D ED0440          COMA,R1 DPNT      COMPARE ACTUAL DECIMAL POINT WITH
0052                 *                         REQUIRED DECIMAL POINT.
0053 0460 9A1C            BCFR,LT TDP1      BRANCH IF EQUAL OR TOO BIG
0054 0462 20              EORZ    R0        CLEAR R0
0055 0463 0CA442          LODA,R0 *TADR,R0,+   FETCH MS-DIGITS OF OPERAND
0056 0466 44F0            ANDI,R0 H'F0'     CLEAR LS-DIGIT (TEST MSD = 0)
```

```
TWIN ASSEMBLER VER 1.0                                    PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0057 0468 9C04C8          BCFA,Z  OVF0      BRANCH IF ALIGNMENT OVERFLOW
0058 046B D900            BIRR,R1 $+2       INCREASE DECIMAL POINT
0059                 *                      SHIFT OPERAND ONE DIGIT LEFT,
0060                 *                         EXCEPT MS-BYTE (SIGN+DPNT)
0061 046D 7501       SHL0 CPSL    C         CLEAR CARRY
0062 046F 0704            LODI,R3 LENG-1    LOAD INDEX REG
0063 0471 0FE442     SHL1 LODA,R0 *TADR,R3  FETCH BYTE OF OPERAND
0064 0474 D0              RRL,R0            ROTATE LEFT WITH CARRY
0065 0475 CFE442          STRA,R0 *TADR,R3  RESTORE
0066 0478 FB77            BDRR,R3 SHL1      BRANCH IF ALL NOT SHIFTED
0067 047A FA71            BDRR,R2 SHL0      BRANCH IF 4 BITS NOT SHIFTED
0068 047C 1B5D            BCTR,UN TDP0      BRANCH FOR NEXT TEST
0069                 *
0070 047E 9933       TDP1 BCFR,GT ARDY      BRANCH IF DECIMAL POINT IS CORRECT
0071 0480 F900            BDRR,R1 $+2       DECREASE DECIMAL POINT
0072 0482 ED0440          COMA,R1 DPNT      TEST IF LS-DIGIT IS ROUNDING DIGIT
0073 0485 9818            BCFR,EQ SHR0      BRANCH IF NOT
0074                 *                      ADD (ROUN) TO ROUNDING-DIGIT:
0075 0487 7501            CPSL    C         CLEAR CARRY
0076 0489 0704            LODI,R3 LENG-1    LOAD INDEX REGISTER
0077 048B 0FE442     RND0 LODA,R0 *TADR,R3  FETCH BYTE OF OPERAND
0078 048E 8466            ADDI,R0 H'66'     ADD OFFSET FOR BCD ADD
0079 0490 E704            COMI,R3 LENG-1
0080 0492 9803            BCFR,EQ RND1      BRANCH IF NOT LS-BYTE
0081 0494 8C0441          ADDA,R0 ROUN      ADD ROUNDING CONSTANT
0082 0497 94         RND1 DAR,R0            DECIMAL ADJUST RESULT
0083 0498 CFE442          STRA,R0 *TADR,R3  RESTORE RESULT
0084 049B FB6E            BDRR,R3 RND0      BRANCH IF ALL BYTES NOT READY
0085 049D 1B02            BCTR,UN SHR1      BRANCH TO RIGHT-SHIFT OPERAND
0086                 *                      SHIFT OPERAND ONE DIGIT RIGHT,
0087                 *                         EXCEPT MS-BYTE (SIGN+DPNT)
0088 049F 7501       SHR0 CPSL    C         CLEAR CARRY
0089 04A1 0700       SHR1 LODI,R3 0         CLEAR INDEX
0090 04A3 0FA442     SHR2 LODA,R0 *TADR,R3,+ FETCH BYTE OF OPERAND
0091 04A6 50              RRR,R0            ROTATE RIGHT WITH CARRY
0092 04A7 CFE442          STRA,R0 *TADR,R3  RESTORE BYTE
0093 04AA E704            COMI,R3 LENG-1
0094 04AC 9875            BCFR,EQ SHR2      BRANCH IF ALL NOT SHIFTED
0095 04AE FA6F            BDRR,R2 SHR0      BRANCH IF 4 BITS NOT SHIFTED
0096 04B0 1F045B          BCTA,UN TDP0      BRANCH FOR NEXT TEST
0097                 *
0098 04B3 0E8442     ARDY LODA,R2 *TADR     FETCH MS-BYTE OF OPERAND
0099 04B6 46F0            ANDI,R2 H'F0'     REMOVE DECIMAL POINT, KEEP SIGN
0100 04B8 0704            LODI,R3 LENG-1    LOAD INDEX REGISTER FOR ZERO TEST
0101 04BA 0FE442     TZER LODA,R0 *TADR,R3  FETCH BYTE OF ALIGNED OPERAND
0102 04BD 9803            BCFR,Z  NZER      BRANCH IF NON-ZERO
0103 04BF FB79            BDRR,R3 TZER      BRANCH IF ALL BYTES NOT READY
0104 04C1 C2              STRZ    R2        ZERO RESULT; CLEAR SIGN
0105 04C2 02         NZER LODZ    R2        FETCH SIGN
0106 04C3 61              IORZ    R1        ASSEMBLE SIGN AND DECIMAL POINT
0107 04C4 CC8442          STRA,R0 *TADR     STORE IN MS-BYTE OF OPERAND
0108 04C7 17              RETC,UN           RETURN
0109                 *
0110 04C8 40         OVF0 HALT              ALIGNMENT OVERFLOW
0111                 *
0112 0000                 END  0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 14

Program Title
FIXED-POINT ADDITION/SUBTRACTION OF SIGNED, PACKED BCD NUMBERS

Function
Addition/subtraction of 2 decimal fixed-point numbers.

Operands and result are of equal length as defined by LENG.

OPERAND1 +/- OPERAND2 ➔ OPERAND2

Parameters

Input:
Length of numbers (in bytes) defined by LENG.

OPR1, OPR1+1, OPR1+2, etc. contain augend or subtrahend.

OPR2, OPR2+1, OPR2+2, etc., contain addend or minuend.

In the alignment subroutine, the decimal-point position is in DPNT and the rounding constant is in ROUN.

Output:
OPR2, OPR2+1, OPR2+2, etc., contain sum or difference.

Result and operand1 are aligned (and rounded if specified).

Overflow is detected.

Special Requirements
Software: Fixed-point alignment subroutine ALGN.

Operation
Subtraction is performed by changing the sign of the second operand before entering the (signed) addition routine. Prior to the addition/subtraction of the magnitudes of the operands, both operands are aligned (and rounded if programmed), the sign of the result is determined and, in the event the operands have opposite signs, the subtrahend and minuend are designated.

Refer to Figures 15 and 16 for flowchart and program listing.

FLOWCHART FOR ADDITION/SUBTRACTION OF FIXED-POINT NUMBERS

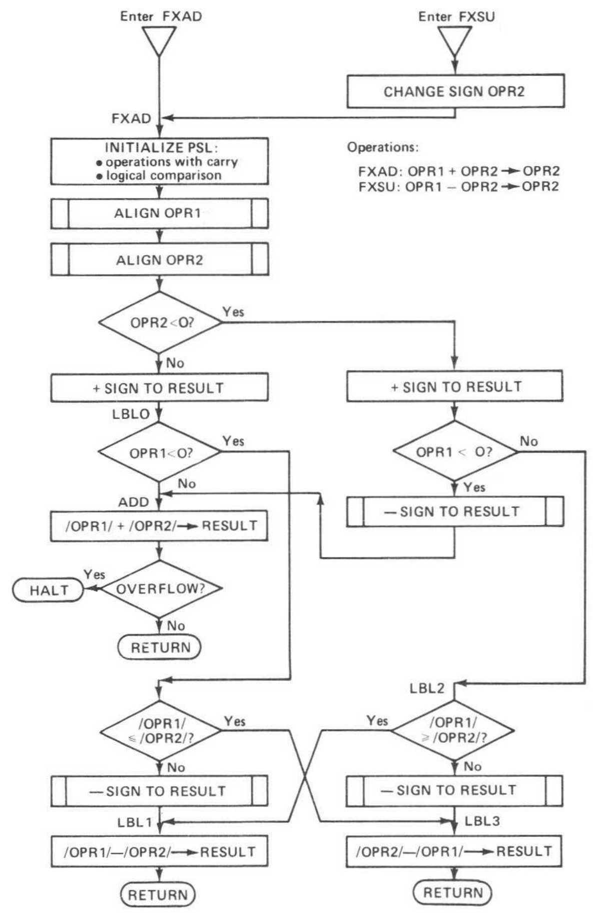


Figure 15

HARDWARE AFFECTED							
REGISTERS	R0 X	R1 X	R2 X	R3 X	R1′	R2′	R3′
PSU	F	II	SP				
PSL	CC X	IDC X	RS	WC X	OVF X	COM X	C X

RAM REQUIRED (BYTES): 2 x LENG
ROM REQUIRED (BYTES): 151

MAXIMUM SUBROUTINE NESTING LEVELS: 1

ASSEMBLER/COMPILER USED: TWIN VER 1.0

FIXED-POINT DECIMAL ADDITION/SUBTRACTION FOR SIGNED, PACKED BCD NUMBERS

```
TWIN ASSEMBLER VER 1.0                                  PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                 * PD760082
0002                 *********************************************
0003                 * FIXED-POINT DECIMAL ADDITION/SUBTRACTION *
0004                 * FOR SIGNED, PACKED BCD NUMBERS.           *
0005                 *********************************************
0006                 * OPERATION:  OPERAND1 +/- OPERAND2 --> OPERAND2
0007                 * OPERAND1        IS IN:  OPR1,OPR1+1,OPR1+2, ETC.
0008                 * OPERAND2        IS IN:  OPR2,OPR2+1,OPR2+2, ETC.
0009                 * SUM/DIFFERENCE IS IN:  OPR2,OPR2+1,OPR2+2, ETC.
0010                 * OPERAND2 IS DESTROYED AFTER ADD/SUBTRACT.
0011                 * OPR1,OPR2 ARE MOST-SIGNIFICANT BYTES.
0012                 * LENGTH OF NUMBERS (IN BYTES) IS DEFINED BY: LENG.
0013                 *    ALLOWED RANGE:  1 < LENG < 255.
0014                 * NUMBERS ARE IN SIGN-MAGNITUDE NOTATION.
0015                 * MS-BYTE HOLDS SIGN AND DECIMAL POINT INFORMATION:
0016                 *    SIGN IS IN MS 4 BITS:  H'0' IS +, H'F' IS -
0017                 *    DECIMAL POINT IS IN LS 4 BITS: BINARY CODED,
0018                 *       RANGE (0 THRU 15) EQUALS NUMBER OF DECIMALS.
0019                 *
0020                 * DEFINITIONS OF SYMBOLS:
0021                 *
0022 0000            R0   EQU     0         PROCESSOR REGISTERS
0023 0001            R1   EQU     1
0024 0002            R2   EQU     2
0025 0003            R3   EQU     3
0026 0008            WC   EQU     H'08'     PSL:  1=WITH, 0=WITHOUT CARRY
0027 0002            COM  EQU     H'02'           1=LOGIC, 0=ARITH.COMPARE
0028 0001            C    EQU     H'01'           CARRY/BORROW
0029 0000            Z    EQU     0         BRANCH COND.:  ZERO
0030 0002            N    EQU     2                        NEGATIVE
0031 0000            EQ   EQU     0                        EQUAL
0032 0001            GT   EQU     1                        GREATER THAN
0033 0002            LT   EQU     2                        LESS THAN
0034 0003            UN   EQU     3                        UNCONDITIONAL
0035                 *
0036                 * PARAMETERS *
0037                 *
0038 0450            ALGN EQU     H'450'    ADDRESS OF ALIGNMENT SUBROUTINE
0039 0005            LENG EQU     5         LENGTH OF OPERANDS (IN BYTES)
0040                 *
0041 0000                 ORG     H'700'
0042                 *
0043 0700            OPR1 RES     LENG      OPERAND1
0044 0705            OPR2 RES     LENG      OPERAND2/RESULT
0045                 *


TWIN ASSEMBLER VER 1.0                                  PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0047 070A                 ORG     H'500'
0048                 *
0049                 ***************************************************************
0050                 * SUBROUTINE TO COMPARE OPERAND1 WITH OPERAND2 (UPDATE CC) *
0051                 ***************************************************************
0052 0500 0500       CO12 LODI,R1 0         CLEAR R1; MS-BITS ARE USED
0053                 *                         TO SAVE CC INFORMATION
0054 0502 0704            LODI,R3 LENG-1    LOAD INDEX REGISTER
0055 0504 0F6700     COM0 LODA,R0 OPR1,R3   FETCH BYTE OF OPERAND1
0056 0507 EF6705          COMA,R0 OPR2,R3   COMPARE WITH BYTE OF OPERAND2
0057 050A 1802            BCTR,EQ COM1      BRANCH IF EQUAL
0058 050C 13              SPSL              PSL TO R0
0059 050D C1              STRZ    R1        SAVE PSL IN R1
0060 050E FB74       COM1 BDRR,R3 COM0      BRANCH IF ALL BYTES NOT TESTED
0061 0510 01              LODZ    R1        UPDATE CC WITH STATUS COMPARE
0062 0511 17              RETC,UN           RETURN
0063                 *
0064                 ************************************************
0065                 * SUBROUTINE TO SET SIGN OF RESULT TO NEGATIVE *
0066                 ************************************************
0067 0512 0C0705     SSGN LODA,R0 OPR2      FETCH SIGN OF RESULT
0068 0515 64F0            IORI,R0 H'F0'     SET NEGATIVE SIGN
0069 0517 CC0705          STRA,R0 OPR2      RESTORE
0070 051A 17              RETC,UN           RETURN
0071                 *
0072                 *                      ****************************
0073                 *                      * FIXED-POINT SUBTRACTION *
0074                 *                      ****************************
0075                 *
0076 051B 0C0705     FXSU LODA,R0 OPR2      FETCH SIGN OF OPERAND2
0077 051E 24F0            EORI,R0 H'F0'     CHANGE SIGN
0078 0520 CC0705          STRA,R0 OPR2      RESTORE SIGN OF OPERAND2
0079                 *
0080                 *                      *************************
0081                 *                      * FIXED-POINT ADDITION *
0082                 *                      *************************
0083                 *
0084 0523 770A       FXAD PPSL    WC+COM    OPERATIONS WITH CARRY, LOGICAL COMPARE
0085 0525 0407            LODI,R0 <OPR1     HIGH-ADDRESS OPR1 TO R0
0086 0527 0500            LODI,R1 >OPR1     LOW- ADDRESS OPR1 TO R1
0087 0529 3F0450          BSTA,UN ALGN      ALIGN OPERAND1
0088 052C 0407            LODI,R0 <OPR2     HIGH-ADDRESS OPR2 TO R0
0089 052E 0505            LODI,R1 >OPR2     LOW- ADDRESS OPR2 TO R1
0090 0530 3F0450          BSTA,UN ALGN      ALIGN OPERAND2
0091 0533 0C0705          LODA,R0 OPR2      FETCH SIGN OPERAND2
0092 0536 C1              STRZ    R1        SAVE IN R1
0093 0537 440F            ANDI,R0 H'0F'     REMOVE SIGN
0094 0539 CC0705          STRA,R0 OPR2      SET SIGN OF RESULT TO +
0095 053C 01              LODZ    R1        FETCH SIGN OPERAND2
0096 053D 9A21            BCFR,N  LBL0      BRANCH IF OPR2 NOT NEGATIVE
0097 053F 0C0700          LODA,R0 OPR1      FETCH SIGN OPERAND1
0098 0542 9A3A            BCFR,N  LBL2      BRANCH IF OPR1 NOT NEGATIVE
0099 0544 3F0512          BSTA,UN SSGN      SET NEGATIVE SIGN RESULT


TWIN ASSEMBLER VER 1.0                                  PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0101                 *                      (OPR1) + (OPR2) --> OPR2
0102 0547 7501       ADD  CPSL    C         CLEAR CARRY
0103 0549 0704            LODI,R3 LENG-1    LOAD INDEX REGISTER
0104 054B 0500            LODI,R1 0         CLEAR INTERBYTE CARRY
0105 054D 0F6700     ADD0 LODA,R0 OPR1,R3   FETCH BYTE OF OPERAND1
0106 0550 8466            ADDI,R0 H'66'     ADD OFFSET FOR BCD ADD
0107 0552 51              RRR,R1            INTERBYTE CARRY TO CARRY
0108 0553 8F6705          ADDA,R0 OPR2,R3   ADD BYTE OF OPERAND2
0109 0556 94              DAR,R0            DECIMAL ADJUST RESULT
0110 0557 CF6705          STRA,R0 OPR2,R3   STORE RESULTING BYTE
0111 055A D1              RRL,R1            CARRY (=INTERBYTE CARRY) TO R1, CLEAR CARRY
0112 055B FB70            BDRR,R3 ADD0      BRANCH IF NOT READY
0113 055D 9838            BCFR,Z  OVFL      BRANCH IF INTERBYTE CARRY = 1
0114 055F 17              RETC,UN           RETURN
0115                 *
0116 0560 0C0700     LBL0 LODA,R0 OPR1      FETCH SIGN OF OPERAND1
0117 0563 9A62            BCFR,N  ADD       BRANCH IF OPR1 NOT NEGATIVE
0118 0565 3F0500          BSTA,UN CO12      COMPARE OPR1 WITH OPR2,
0119                 *                         (MAGNITUDES ONLY).
0120 0568 991C            BCFR,GT LBL3      BRANCH IF OPR1 < OR = OPR2
0121 056A 3F0512          BSTA,UN SSGN      SET NEGATIVE SIGN OF RESULT
0122                 *
0123                 *                      (OPR1) - (OPR2) --> OPR2:
0124 056D 0704       LBL1 LODI,R3 LENG-1    LOAD INDEX REGISTER
0125 056F 7701            PPSL    C         CLEAR BORROW
0126 0571 0F6700     SU12 LODA,R0 OPR1,R3   FETCH BYTE OF OPERAND1
0127 0574 AF6705          SUBA,R0 OPR2,R3   SUBTRACT BYTE OF OPERAND2
0128 0577 94              DAR,R0            DECIMAL ADJUST RESULT
0129 0578 CF6705          STRA,R0 OPR2,R3   STORE RESULTING BYTE IN OPR2
0130 057B FB74            BDRR,R3 SU12      BRANCH IF NOT READY
0131 057D 17              RETC,UN           RETURN
0132                 *
0133 057E 3F0500     LBL2 BSTA,UN CO12      COMPARE OPR1 WITH OPR2,
0134                 *                         (MAGNITUDES ONLY)
0135 0581 9A6A            BCFR,LT LBL1      BRANCH IF OPR1 > OR = OPR2
0136 0583 3F0512          BSTA,UN SSGN      SET NEGATIVE SIGN OF RESULT
0137                 *
0138                 *                      (OPR2) - (OPR1) --> OPR2:
0139 0586 0704       LBL3 LODI,R3 LENG-1    LOAD INDEX REGISTER
0140 0588 7701            PPSL    C         CLEAR BORROW
0141 058A 0F6705     SU21 LODA,R0 OPR2,R3   FETCH BYTE OF OPERAND2
0142 058D AF6700          SUBA,R0 OPR1,R3   SUBTRACT BYTE OF OPERAND1
0143 0590 94              DAR,R0            DECIMAL ADJUST RESULT
0144 0591 CF6705          STRA,R0 OPR2,R3   STORE RESULTING BYTE
0145 0594 FB74            BDRR,R3 SU21      BRANCH IF NOT READY
0146 0596 17              RETC,UN           RETURN
0147                 *
0148 0597 40         OVFL HALT              ARITHMETIC OVERFLOW
0149                 *
0150 0000                 END     0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 16

Program Title

FIXED-POINT DECIMAL MULTIPLICATION FOR SIGNED, PACKED BCD NUMBERS

Function

Multiplication of 2 decimal fixed-point numbers.

Multiplicand, multiplier, and product are of equal length as defined by LENG.

MULTIPLICAND x MULTIPLIER $\rightarrow$ MULTIPLIER

Parameters

Input:
Length of numbers (in bytes) is defined by LENG.

MPLC, MPLC+1, MPLC+2, etc., contain multiplicand.

MPLR, MPLR+1, MPLR+2, etc., contain multiplier.

Output:
MPLR, MPLR+1, MPLR+2, etc., contain product.

Multiplier is destroyed after multiplication.

Overflow is detected.

Special Requirements

Software: Fixed-point alignment subroutine ALGN

Operation

Prior to the multiplication algorithm (which is an unsigned operation), the product sign is determined. The product is formed in a double-length register and is right aligned until the decimal point is 15 or less; this is required due to the fixed-point format. Then the product length is reduced to the single-length, fixed-point format; if this is not possible, overflow is detected. A "minus zero" product result is excluded by means of a test during aligning.

Refer to Figures 17 and 18 for flowchart and program listing.

FLOWCHART FOR MULTIPLICATION OF FIXED-POINT NUMBERS

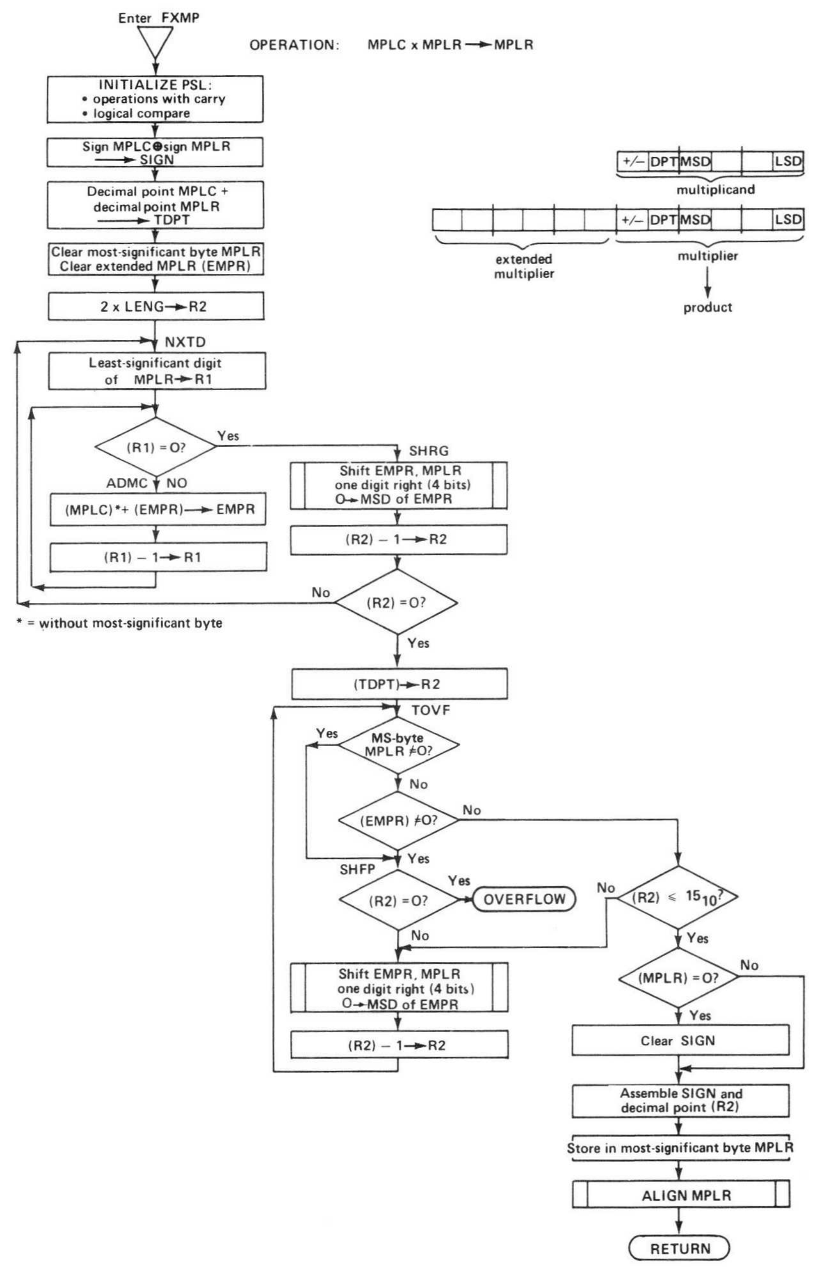

Enter FXMP
OPERATION: MPLC x MPLR → MPLR
INITIALIZE PSL: • operations with carry • logical compare
Sign MPLC⊕sign MPLR → SIGN
Decimal point MPLC + decimal point MPLR → TDPT
Clear most-significant byte MPLR Clear extended MPLR (EMPR)
2 x LENG → R2
NXTD
Least-significant digit of MPLR → R1
(R1) = 0?
Yes
ADMC
NO
(MPLC)*+ (EMPR) → EMPR
(R1) − 1 → R1
SHRG
Shift EMPR, MPLR one digit right (4 bits) 0 → MSD of EMPR
(R2) − 1 → R2
No
(R2) = 0?
Yes
* = without most-significant byte
+/− DPT MSD LSD
multiplicand
extended multiplier
multiplier
product
(TDPT) → R2
TOVF
Yes
MS-byte MPLR ≠0?
No
(EMPR) ≠0?
No
SHFP
Yes
(R2) = 0?
Yes
OVERFLOW
No
Shift EMPR, MPLR one digit right (4 bits) 0 → MSD of EMPR
(R2) − 1 → R2
No
(R2) ≤ 15₁₀?
Yes
(MPLR) = 0?
No
Yes
Clear SIGN
Assemble SIGN and decimal point (R2)
Store in most-significant byte MPLR
ALIGN MPLR
RETURN

Figure 17

HARDWARE AFFECTED							
REGISTERS	R0 X	R1 X	R2 X	R3 X	R1′	R2′	R3′
PSU	F	II	SP				
PSL	CC X	IDC X	RS	WC X	OVF X	COM X	C X

RAM REQUIRED (BYTES): (3 X LENG) +4

ROM REQUIRED (BYTES): 144

MAXIMUM SUBROUTINE NESTING LEVELS: 1

ASSEMBLER/COMPILER USED: TWIN VER 1.0

FIXED-POINT DECIMAL MULTIPLICATION FOR SIGNED, PACKED BCD NUMBERS

```
TWIN ASSEMBLER VER 1.0                                   PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                 * PD760083
0002                 ******************************************
0003                 * FIXED POINT DECIMAL MULTIPLICATION FOR *
0004                 * SIGNED, PACKED-BCD NUMBERS             *
0005                 ******************************************
0006                 * OPERATION: MULTIPLICAND X MULTIPLIER --> MULTIPLIER
0007                 * MULTIPLICAND IS IN:  MPLC,MPLC+1,MPLC+2, ETC.
0008                 * MULTIPLIER   IS IN:  MPLR,MPLR+1,MPLR+2, ETC.
0009                 * PRODUCT      IS IN:  MPLR,MPLR+1,MPLR+2, ETC.
0010                 * MULTIPLIER IS DESTROYED AFTER MULTIPLICATION.
0011                 * MPLC,MPLR ARE MOST-SIGNIFICANT BYTES.
0012                 * LENGTH OF NUMBERS (IN BYTES) IS DEFINED BY: LENG
0013                 *    ALLOWED RANGE:  1 < LENG < 65.
0014                 * REQUIRED NUMBER OF DECIMALS IN PRODUCT MUST BE
0015                 *    STORED IN LOCATION:  DPNT  (MAX = 15).
0016                 * NUMBERS ARE IN SIGN-MAGNITUDE NOTATION.
0017                 * MS-BYTE HOLDS SIGN AND DECIMAL POINT INFORMATION:
0018                 *    SIGN IS IN MS 4 BITS:  H'0' IS +, H'F' IS -
0019                 *    DECIMAL POINT IS IN LS 4 BITS: BINARY CODED,
0020                 *       RANGE (0 THRU 15) EQUALS NUMBER OF DECIMALS.
0021                 *
0022                 * DEFINITIONS OF SYMBOLS:
0023                 *
0024 0000            R0   EQU    0         PROCESSOR REGISTERS
0025 0001            R1   EQU    1
0026 0002            R2   EQU    2
0027 0003            R3   EQU    3
0028 0008            WC   EQU    H'08'     PSL:  1=WITH, 0=WITHOUT CARRY
0029 0002            COM  EQU    H'02'           1=LOGIC, 0=ARITH.COMPARE
0030 0001            C    EQU    H'01'           CARRY/BORROW
0031 0000            Z    EQU    0         BRANCH COND.:  ZERO
0032 0002            LT   EQU    2                        LESS THAN
0033 0003            UN   EQU    3                        UNCONDITIONAL
0034                 *
0035                 * PARAMETERS *
0036                 *
0037 0450            ALGN EQU    H'450'    ADDRESS OF ALIGNMENT SUBROUTINE
0038 0005            LENG EQU    5         LENGTH OF PARAMETERS (BYTES)
0039                 *
0040 0000                 ORG    H'700'
0041                 *
0042 0700            MPLC RES    LENG      MULTIPLICAND
0043 0705            EMPR RES    LENG      EXTENDED MULTIPLIER
0044 070A            MPLR RES    LENG      MULTIPLIER
0045                 * NOTE: EMPR AND MPLR MUST BE IN SUCCESSIVE
0046                 *       RAM LOCATIONS FOR DOUBLE-LENGTH SHIFT.
0047 070F            SIGN RES    1         TEMPORARY SIGN
0048 0710            TEMP RES    2         TEMPORARY STORAGE FOR ADDRESS
0049 0712            TDPT RES    1         TEMPORARY STORAGE FOR DECIMAL POINT
0050                 *
0051 0713                 ORG    H'500'


TWIN ASSEMBLER VER 1.0                                   PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0053                 ********************************************************
0054                 * SUBROUTINE TO SHIFT EMPR AND MPLR ONE DIGIT RIGHT *
0055                 ********************************************************
0056                 * PRIOR TO ENTRY:  WC IN PSL MUST BE 1.
0057                 *
0058 0500 0504       SHEM LODI,R1 4          LOAD LOOP COUNTER
0059 0502 07F6       SHE0 LODI,R3 -LENG-LENG  LOAD INDEX REGISTER
0060 0504 7501            CPSL    C          CLEAR CARRY
0061 0506 0F660F     SHE1 LODA,R0 EMPR-256+LENG+LENG,R3  FETCH BYTE
0062 0509 50              RRR,R0             ROTATE RIGHT
0063 050A CF660F          STRA,R0 EMPR-256+LENG+LENG,R3  RESTORE BYTE
0064 050D DB77            BIRR,R3 SHE1       BRANCH IF ALL NOT SHIFTED
0065 050F F971            BDRR,R1 SHE0       BRANCH IF 4 BITS NOT SHIFTED
0066 0511 17              RETC,UN            RETURN
0067                 *
0068                 ******************************
0069                 * FIXED POINT MULTIPLICATION *
0070                 ******************************
0071                 *
0072 0512 770A       FXMP PPSL    WC+COM     OPERATIONS WITH CARRY, LOGICAL COMPARE
0073 0514 0D0700          LODA,R1 MPLC       FETCH MS-BYTE MULTIPLICAND
0074 0517 01              LODZ    R1         SAVE IN R0
0075 0518 0E070A          LODA,R2 MPLR       FETCH MS-BYTE MULTIPLIER
0076 051B 22              EORZ    R2         TAKE EX-OR OF SIGNS
0077 051C 44F0            ANDI,R0 H'F0'      REMOVE NON-SIGN DIGIT
0078 051E CC070F          STRA,R0 SIGN       SAVE SIGN
0079 0521 01              LODZ    R1         MS-BYTE OF MPLC TO R0
0080 0522 440F            ANDI,R0 H'0F'      REMOVE SIGN MPLC, KEEP DECIMAL POINT
0081 0524 460F            ANDI,R2 H'0F'      REMOVE SIGN MPLR, KEEP DECIMAL POINT
0082 0526 7501            CPSL    C          CLEAR CARRY
0083 0528 82              ADDZ    R2         ADD DECIMAL POINT POSITIONS
0084 0529 CC0712          STRA,R0 TDPT       SAVE NEW DECIMAL POINT POSITION
0085                 *
0086 052C 20              EORZ    R0         CLEAR R0
0087 052D 0706            LODI,R3 LENG+1     LOAD INDEX REGISTER
0088 052F CF4705     CLEM STRA,R0 EMPR,R3,-  CLEAR MS-BYTE MPLR, ALL EMPR
0089 0532 5B7B            BRNR,R3 CLEM       BRANCH IF NOT DONE
0090                 *
0091 0534 060A            LODI,R2 LENG+LENG NUMBER OF DIGITS TO LOOP COUNTER
0092 0536 0D070E     NXTD LODA,R1 MPLR+LENG-1  FETCH LS-BYTE MULTIPLIER
0093 0539 450F            ANDI,R1 H'0F'      TAKE ONLY LS-DIGIT
0094 053B 1826            BCTR,Z  SHRG       BRANCH IF ZERO
0095                 *
0096                 *                       ADD MPLC (WITHOUT MS-BYTE) TO EMPR
0097 053D 7501       ADMC CPSL    C          CLEAR CARRY
0098 053F 0704            LODI,R3 LENG-1     LOAD INDEX REGISTER
0099 0541 0F6705     ADM0 LODA,R0 EMPR,R3    FETCH BYTE OF EXTENDED MULTIPLIER
0100 0544 8466            ADDI,R0 H'66'      ADD OFFSET
0101 0546 CF6705          STRA,R0 EMPR,R3    RESTORE INTERMEDIATE SUM
0102 0549 FB76            BDRR,R3 ADM0       BRANCH IF ALL BYTES NOT ADDED
0103 054B 0704            LODI,R3 LENG-1     LOAD INDEX REGISTER
0104 054D 0F6705     ADM1 LODA,R0 EMPR,R3    FETCH BYTE OF INTERMEDIATE SUM
0105 0550 8F6700          ADDA,R0 MPLC,R3    ADD BYTE OF MULTIPLICAND
0106 0553 94              DAR,R0             DECIMAL ADJUST RESULT
0107 0554 CF6705          STRA,R0 EMPR,R3    RESTORE RESULTING BYTE


TWIN ASSEMBLER VER 1.0                                   PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0108 0557 FB74            BDRR,R3 ADM1       BRANCH IF NOT READY
0109 0559 0C0705          LODA,R0 EMPR       FETCH MS-BYTE EXTENDED MULTIPLIER
0110 055C 8400            ADDI,R0 0          ADD CARRY
0111 055E CC0705          STRA,R0 EMPR       RESTORE
0112 0561 F95A            BDRR,R1 ADMC       DECREMENT DIGIT, BRANCH IF NOT 0
0113 0563 3F0500     SHRG BSTA,UN SHEM       SHIFT EMPR AND MPLR RIGHT ONE DIGIT POSITION
0114 0566 FA4E            BDRR,R2 NXTD       BRANCH IF MULTIPLICATION NOT READY
0115                 *
0116 0568 0E0712          LODA,R2 TDPT       DECIMAL POINT TO R2
0117 056B 0706       TOVF LODI,R3 LENG+1     TEST OVERFLOW, LOAD INDEX REGISTER
0118 056D 0F4705     TOV0 LODA,R0 EMPR,R3,-  FETCH BYTE OF EMPR OR MS-BYTE
0119                 *                          OF MPLR TO TEST FOR ZERO.
0120 0570 9814            BCFR,Z  SHFP       BRANCH IF NOT ZERO
0121 0572 5B79            BRNR,R3 TOV0       BRANCH IF ALL NOT TESTED
0122                 *
0123 0574 E610            COMI,R2 16         TEST IF DECIMAL POINT IS < 16.
0124 0576 9A11            BCFR,LT SHF0       BRANCH IF TOO BIG
0125 0578 6E070F          IORA,R2 SIGN       ASSEMBLE SIGN AND DECIMAL POINT
0126 057B CE070A     ASMB STRA,R2 MPLR       STORE IN MS-BYTE MPLR.
0127 057E 0407            LODI,R0 <MPLR      HIGH-ORDER ADDRESS MPLR TO R0
0128 0580 050A            LODI,R1 >MPLR      LOW- ORDER ADDRESS MPLR TO R1
0129 0582 3F0450          BSTA,UN ALGN       ALIGN PRODUCT; SET + SIGN IF
0130                 *                          PRODUCT IS ZERO.
0131 0585 17              RETC,UN
0132                 *
0133 0586 02        SHFP LODZ    R2         UPDATE CC FOR NUMBER OF DECIMALS
0134 0587 1807            BCTR,Z  OVFL       BRANCH IF ZERO: OVERFLOW
0135 0589 FA00       SHF0 BDRR,R2 $+2        DECREASE DECIMAL POINT
0136 058B 3F0500          BSTA,UN SHEM       SHIFT EMPR + MPLR RIGHT
0137 058E 1B5B            BCTR,UN TOVF       BRANCH FOR OVERFLOW TEST
0138                 *
0139 0590 40         OVFL HALT               ARITHMETIC OVERFLOW
0140                 *
0141 0000                 END     0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 18

Program Title
FIXED-POINT DECIMAL DIVISION FOR SIGNED, PACKED BCD NUMBERS

Function
Division of 2 decimal numbers (fixed point).

Dividend, divisor, and quotient are of equal length as defined by LENG.

DIVIDEND : DIVISOR ➜ DIVIDEND.

Parameters

Input:
Length of numbers (in bytes) is defined by LENG.

DVDN, DVDN+1, DVDN+2, etc., contain dividend.

DVSR, DVSR+1, DVSR+2, etc., contain divisor.

Output:
DVDN, DVDN+1, DVDN+2, etc., contain quotient.

Dividend is destroyed after division.

Overflow is detected.

Special Requirements
Software: Fixed-point alignment subroutine ALGN.

Operation
Prior to the division algorithm (which is an unsigned operation), the sign of the quotient is determined. To obtain maximum precision, the division procedure is continued until either a non-zero most-significant digit is detected or the maximum allowed decimal point position is reached. Then the resulting quotient is aligned with a minus zero result suppressed. Overflow is detected if the divisor is zero.

Refer to Figures 19 and 20 for flowchart and program listing.

FLOWCHART FOR DIVISION OF FIXED-POINT NUMBERS

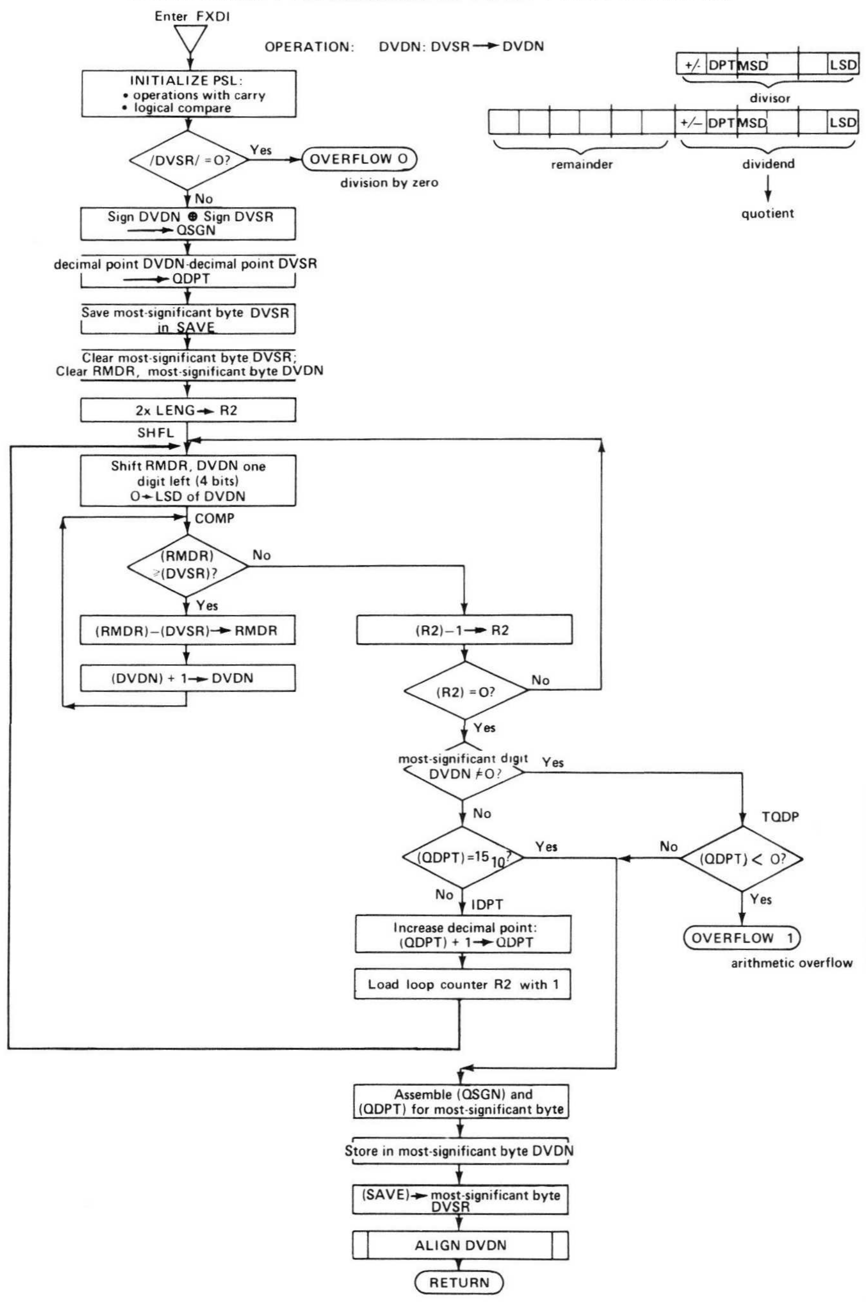

Enter FXDI
OPERATION: DVDN: DVSR ➜ DVDN
+/- DPT MSD LSD
divisor
+/- DPT MSD LSD
remainder
dividend
quotient
INITIALIZE PSL:
• operations with carry
• logical compare
/DVSR/ = O?
Yes
OVERFLOW O
division by zero
No
Sign DVDN ⊕ Sign DVSR ➜ QSGN
decimal point DVDN-decimal point DVSR ➜ QDPT
Save most-significant byte DVSR in SAVE
Clear most-significant byte DVSR; Clear RMDR, most-significant byte DVDN
2x LENG ➜ R2
SHFL
Shift RMDR, DVDN one digit left (4 bits) O ➜ LSD of DVDN
COMP
(RMDR) ≥ (DVSR)?
No
Yes
(RMDR)−(DVSR) ➜ RMDR
(DVDN) + 1 ➜ DVDN
(R2)−1 ➜ R2
(R2) = O?
No
Yes
most-significant digit DVDN ≠ O?
Yes
No
TQDP
(QDPT) = 15₁₀?
Yes
No
(QDPT) < O?
Yes
OVERFLOW 1
arithmetic overflow
No
IDPT
Increase decimal point: (QDPT) + 1 ➜ QDPT
Load loop counter R2 with 1
Assemble (QSGN) and (QDPT) for most-significant byte
Store in most-significant byte DVDN
(SAVE) ➜ most-significant byte DVSR
ALIGN DVDN
RETURN

Figure 19

HARDWARE AFFECTED							
REGISTERS	R0 X	R1 X	R2 X	R3 X	R1′	R2′	R3′
PSU	F	II	SP				
PSL	CC X	IDC X	RS	WC X	OVF X	COM X	C X

RAM REQUIRED (BYTES): (3 x LENG) +5

ROM REQUIRED (BYTES): 166

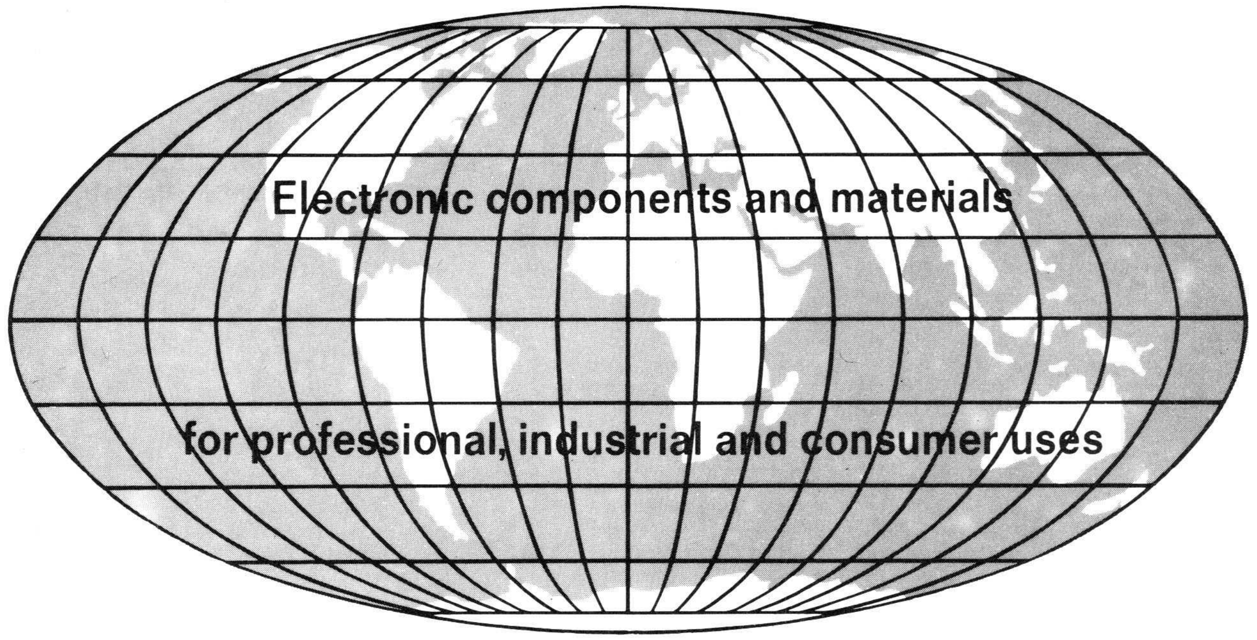
MAXIMUM SUBROUTINE NESTING LEVELS: 1

ASSEMBLER/COMPILER USED: TWIN VER 1.0

FIXED-POINT DECIMAL DIVISION FOR SIGNED, PACKED BCD NUMBERS

```
TWIN ASSEMBLER VER 1.0                                PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                 * FD760081
0002                 ***********************************
0003                 * FIXED-POINT DECIMAL DIVISION    *
0004                 * FOR SIGNED, PACKED-BCD NUMBERS *
0005                 ***********************************
0006                 * OPERATION:  DIVIDEND : DIVISOR --> DIVIDEND.
0007                 * DIVIDEND  IS IN: DVDN,DVDN+1,DVDN+2, ETC.
0008                 * DIVISOR   IS IN: DVSR,DVSR+1,DVSR+2, ETC.
0009                 * QUOTIENT  IS IN: DVDN,DVDN+1,DVDN+2, ETC.
0010                 * DIVIDEND IS DESTROYED AFTER DIVISION.
0011                 * DVDN AND DVSR ARE MOST-SIGNIFICANT BYTES.
0012                 * LENGTH OF NUMBERS (IN BYTES) IS DEFINED BY: LENG.
0013                 *    ALLOWED RANGE:  1 < LENG < 65.
0014                 * NUMBERS ARE IN SIGN-MAGNITUDE NOTATION.
0015                 * MS-BYTE HOLDS SIGN AND DECIMAL POINT INFORMATION:
0016                 *    SIGN IS IN MS 4 BITS:  H'0' IS +, H'F' IS -
0017                 *    DECIMAL POINT IS IN LS 4 BITS: BINARY CODED,
0018                 *       RANGE (0 THRU 15) EQUALS NUMBER OF DECIMALS.
0019                 *
0020                 * DEFINITIONS OF SYMBOLS:
0021                 *
0022 0000            R0   EQU     0        PROCESSOR REGISTERS
0023 0001            R1   EQU     1
0024 0002            R2   EQU     2
0025 0003            R3   EQU     3
0026 0008            WC   EQU     H'08'    PSL:  1=WITH, 0=WITHOUT CARRY
0027 0002            COM  EQU     H'02'          1=LOGIC, 0=ARITH.COMPARE
0028 0001            C    EQU     H'01'          CARRY/BORROW
0029 0000            Z    EQU     0        BRANCH COND.:  ZERO
0030 0001            P    EQU     1                       POSITIVE
0031 0002            N    EQU     2                       NEGATIVE
0032 0000            EQ   EQU     0                       EQUAL
0033 0002            LT   EQU     2                       LESS THAN
0034 0003            UN   EQU     3                       UNCONDITIONAL
0035                 *
0036                 * PARAMETERS *
0037                 *
0038 0450            ALGN EQU     H'450'   ADDRESS OF ALIGNMENT SUBROUTINE
0039 0005            LENG EQU     5        LENGTH OF OPERANDS (IN BYTES)
0040                 *
0041 0000                 ORG     H'700'
0042                 *
0043 0700            RMDR RES     LENG     REMAINDER
0044 0705            DVDN RES     LENG     DIVIDEND
0045                 * NOTE: RMDR AND DVDN MUST BE IN SUCCESSIVE
0046                 *       RAM LOCATIONS, BECAUSE OF DOUBLE-LENGTH SHIFT
0047 070A            DVSR RES     LENG     DIVISOR
0048 070F            TEMP RES     2        TEMPORARY STORAGE FOR ADDRESS
0049 0711            QSGN RES     1        QUOTIENT SIGN
0050 0712            QDPT RES     1        QUOTIENT DECIMAL POINT
0051 0713            SAVE RES     1        TEMPORARY STORAGE
0052                 *


TWIN ASSEMBLER VER 1.0                                PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0054 0714                 ORG     H'500'
0055                 *
0056 0500 7708       FXDI PPSL    WC+COM+C  OPERATIONS WITH CARRY,
0057                 *                         LOGICAL COMPARISON, CLEAR BORROW
0058 0502 0704            LODI,R3 LENG-1    LOAD INDEX REGISTER FOR ZERO TEST
0059 0504 0F670A     TZER LODA,R0 DVSR,R3   FETCH BYTE OF DIVISOR
0060 0507 9805            BCFR,Z  NZER      BRANCH IF NON-ZERO
0061 0509 FB79            BDRR,R3 TZER      BRANCH IF ALL BYTES NOT RDY
0062 050B 1C05A6          BCTA,Z  OVF0      BRANCH IF ZERO
0063 050E 0C0705     NZER LODA,R0 DVDN      FETCH MS-BYTE DIVIDEND
0064 0511 C1              STRZ    R1        SAVE IN R1
0065 0512 0E070A          LODA,R2 DVSR      FETCH MS-BYTE DIVISOR
0066 0515 CE0713          STRA,R2 SAVE      SAVE MS-BYTE DIVISOR
0067 0518 22              EORZ    R2        EX-OR SIGN DVDN AND DVSR
0068 0519 44F0            ANDI,R0 H'F0'     REMOVE DECIMAL POINT DIGIT
0069 051B CC0711          STRA,R0 QSGN      SAVE QUOTIENT SIGN
0070 051E 01              LODZ    R1        FETCH MS-BYTE DIVIDEND
0071 051F 440F            ANDI,R0 H'0F'     REMOVE SIGN
0072 0521 460F            ANDI,R2 H'0F'     REMOVE SIGN MS-BYTE DIVISOR
0073 0523 A2              SUBZ    R2        SUBTRACT DECIMAL POINTS: DVDN - DVSR
0074 0524 CC0712          STRA,R0 QDPT      SAVE DECIMAL POINT QUOTIENT
0075                 *
0076 0527 20              EORZ    R0        CLEAR R0
0077 0528 CC070A          STRA,R0 DVSR      CLEAR MS-BYTE DIVISOR
0078 052B 0706            LODI,R3 LENG+1    LOAD INDEX REGISTER
0079 052D CF4700     CLRM STRA,R0 RMDR,R3,- CLEAR REMAINDER AND SIGN DVDN
0080 0530 5B7B            BRNR,R3 CLRM      BRANCH IF NOT DONE
0081                 *
0082 0532 060A            LODI,R2 LENG+LENG NUMBER OF DIGITS TO LOOP COUNTER
0083                 *
0084                 *                      SHIFT RMDR/DVDN 4 BITS LEFT
0085                 *                         INSERTING ZEROES IN LS-BITS
0086 0534 0504       SHFL LODI,R1 4         LOAD BIT COUNTER
0087 0536 7501       SHF0 CPSL    C         CLEAR CARRY
0088 0538 070A            LODI,R3 LENG+LENG LOAD INDEX REGISTER
0089 053A 0F4700     SHF1 LODA,R0 RMDR,R3,- FETCH BYTE OF RMDR/DVDN
0090 053D D0              RRL,R0            ROTATE LEFT WITH CARRY
0091 053E CF6700          STRA,R0 RMDR,R3   RESTORE SHIFTED BYTE
0092 0541 5B77            BRNR,R3 SHF1      BRANCH IF ALL NOT SHIFTED
0093 0543 F971            BDRR,R1 SHF0      BRANCH IF 4 BITS NOT SHIFTED
0094                 *
0095                 *                      COMPARE RMDR AND DVSR TO TEST
0096                 *                         IF SUBTRACTION IS POSSIBLE.
0097 0545 0500       COMP LODI,R1 0         CLEAR R1; MS-BIT OF R1 BECOMES
0098                 *                         1 FOR RMDR < DVSR.


TWIN ASSEMBLER VER 1.0                                PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0100 0547 0705            LODI,R3 LENG      LOAD INDEX REGISTER
0101 0549 0F4700     COM0 LODA,R0 RMDR,R3,- FETCH BYTE OF REMAINDER
0102 054C EF670A          COMA,R0 DVSR,R3   COMPARE WITH BYTE OF DIVISOR
0103 054F 1802            BCTR,EQ COM1      BRANCH IF EQUAL
0104 0551 13              SPSL              PSL TO R0
0105 0552 C1              STRZ    R1        SAVE PSL IN R1
0106 0553 5B74       COM1 BRNR,R3 COM0      BRANCH IF ALL BYTES NOT TESTED
0107 0555 01              LODZ    R1        FETCH STATUS OF COMPARISON
0108 0556 1A1A            BCTR,LT NXDG      BRANCH IF RMDR < DVSR
0109                 *
0110                 *                      SUBTRACT DIVISOR FROM REMAINDER
0111 0558 7701            PPSL    C         CLEAR BORROW
0112 055A 0705            LODI,R3 LENG      LOAD INDEX REGISTER
0113 055C 0F4700     SURD LODA,R0 RMDR,R3,- FETCH BYTE OF REMAINDER
0114 055F AF670A          SUBA,R0 DVSR,R3   SUBTRACT BYTE OF DIVISOR
0115 0562 94              DAR,R0            DECIMAL ADJUST RESULT
0116 0563 CF6700          STRA,R0 RMDR,R3   RESTORE IN REMAINDER
0117 0566 5B74            BRNR,R3 SURD      BRANCH IF NOT READY
0118                 *
0119 0568 0C0709          LODA,R0 DVDN+LENG-1  FETCH LS-BYTE QUOTIENT
0120 056B D800            BIRR,R0 $+2       INCREASE R0
0121 056D CC0709          STRA,R0 DVDN+LENG-1 RESTORE INCREMENTED QUOTIENT
0122 0570 1B53            BCTR,UN COMP      BRANCH FOR NEXT COMPARISON
0123                 *
0124 0572 FA40       NXDG BDRR,R2 SHFL      BRANCH IF DIVISION NOT READY
0125 0574 20              EORZ    R0        CLEAR INDEX REGISTER
0126 0575 0C2705          LODA,R0 DVDN,R0,+ FETCH MS-DIGITS QUOTIENT
0127 0578 44F0            ANDI,R0 H'F0'     TAKE MSD ONLY
0128 057A 9811            BCFR,Z  TQDP      BRANCH IF MSD NOT ZERO
0129 057C 0E0712          LODA,R2 QDPT      FETCH DECIMAL POINT QUOTIENT
0130 057F E60F            COMI,R2 15
0131 0581 980F            BCFR,EQ ASQU      BRANCH IF DECIMAL POINT=MAX.
0132 0583 DA00       IDPT BIRR,R2 $+2       INCREASE DECIMAL POINT QUOTIENT
0133 0585 CE0712          STRA,R2 QDPT      RESTORE
0134 0588 0601            LODI,R2 1         LOAD LOOP COUNTER
0135 058A 1F0534          BCTA,UN SHFL      BRANCH FOR NEXT DIVIDE LOOP
0136                 *
0137 058D 0E0712     TQDP LODA,R2 QDPT      FETCH DECIMAL POINT QUOTIENT
0138 0590 1A15            BCTR,N  OVF1      BRANCH IF NEGATIVE
0139 0592 6E0711     ASQU IORA,R2 QSGN      ASSEMBLE SIGN+DECIMAL POINT QUOTIENT
0140 0595 CE0705          STRA,R2 DVDN      STORE SIGN IN MS-BYTE DVDN
0141 0598 0C0713          LODA,R0 SAVE      FETCH SIGN+DECIMAL POINT DIVISOR
0142 059B CC070A          STRA,R0 DVSR      RESTORE MS-BYTE DIVISOR
0143 059E 0407            LODI,R0 <DVDN     HIGH-ADDRESS QUOTIENT TO R0
0144 05A0 0505            LODI,R1 >DVDN     LOW- ADDRESS QUOTIENT TO R1
0145 05A2 3F0450          BSTA,UN ALGN      ALIGN QUOTIENT; SET + SIGN IF
0146                 *                         QUOTIENT IS ZERO
0147 05A5 17              RETC,UN           RETURN
0148                 *
0149 05A6 40         OVF0 HALT              OVERFLOW, DIVISION BY ZERO
0150 05A7 40         OVF1 HALT              ARITHMETIC OVERFLOW
0151                 *
0152 0000                 END  0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 20


Electronic components and materials
for professional, industrial and consumer uses

from the world-wide Philips Group of Companies

Argentina: FAPESA I.y.C., Av. Crovara 2550, Tablada, Prov. de BUENOS AIRES, Tel. 652-7438/7478.
Australia: PHILIPS INDUSTRIES HOLDINGS LTD., Elcoma Division, 67 Mars Road, LANE COVE, 2066, N.S.W., Tel. 42 1261.
Austria: ÖSTERREICHISCHE PHILIPS BAUELEMENTE Industrie G.m.b.H., Triester Str. 64, A-1101 WIEN, Tel. 62 91 11.
Belgium: M.B.L.E., 80, rue des Deux Gares, B-1070 BRUXELLES, Tel 523 00 00.
Brazil: IBRAPE, Caixa Postal 7383, Av. Paulista 2073-S/Loja, SAO PAULO, SP, Tel. 287-7144.
Canada: PHILIPS ELECTRONICS LTD., Electron Devices Div., 601 Milner Ave., SCARBOROUGH, Ontario, M1B 1M8, Tel. 292-5161.
Chile: PHILIPS CHILENA S.A., Av. Santa Maria 0760, SANTIAGO, Tel. 39-40 01.
Colombia: SADAPE S.A., P.O. Box 9805, Calle 13, No. 51 + 39, BOGOTA D.E. 1., Tel. 600 600.
Denmark: MINIWATT A/S, Emdrupvej 115A, DK-2400 KØBENHAVN NV., Tel. (01) 69 16 22.
Finland: OY PHILIPS AB, Elcoma Division, Kaivokatu 8, SF-00100 HELSINKI 10, Tel. 1 72 71.
France: R.T.C. LA RADIOTECHNIQUE-COMPELEC, 130 Avenue Ledru Rollin, F-75540 PARIS 11, Tel. 355-44-99.
Germany: VALVO, UB Bauelemente der Philips G.m.b.H., Valvo Haus, Burchardstrasse 19, D-2 HAMBURG 1, Tel. (040) 3296-1.
Greece: PHILIPS S.A. HELLENIQUE, Elcoma Division, 52, Av. Syngrou, ATHENS, Tel. 915 311.
Hong Kong: PHILIPS HONG KONG LTD., Comp. Dept., Philips Ind. Bldg., Kung Yip St., K.C.T.L. 289, KWAI CHUNG, N.T. Tel. 12-24 51 21.
India: PHILIPS INDIA LTD., Elcoma Div., Band Box House, 254-D, Dr. Annie Besant Rd., Prabhadevi, BOMBAY-25-DD, Tel. 457 311-5.
Indonesia: P.T. PHILIPS-RALIN ELECTRONICS, Elcoma Division, 'Timah' Building, Jl. Jen. Gatot Subroto, JAKARTA, Tel. 44 163.
Ireland: PHILIPS ELECTRICAL (IRELAND) LTD., Newstead, Clonskeagh, DUBLIN 14, Tel. 69 33 55.
Italy: PHILIPS S.P.A., Sezione Elcoma, Piazza IV Novembre 3, I-20124 MILANO, Tel. 2-6994.
Japan: NIHON PHILIPS CORP., Shuwa Shinagawa Bldg., 26-33 Takanawa 3-chome, Minato-ku, TOKYO (108), Tel. 448-5611.
(IC Products) SIGNETICS JAPAN, LTD., TOKYO, Tel. (03) 230-1521.
Korea: PHILIPS ELECTRONICS (KOREA) LTD., Philips House, 260-199 Itaewon-dong, Yongsan-ku, C.P.O. Box 3680, SEOUL, Tel. 44-4202.
Mexico: ELECTRONICA S.A. de C.V., Varsovia No. 36, MEXICO 6, D.F., Tel. 5-33-11-80.
Netherlands: PHILIPS NEDERLAND B.V., Afd. Elonco, Boschdijk 525, NL-4510 EINDHOVEN, Tel. (040) 79 33 33.
New Zealand: Philips Electrical Ind. Ltd., Elcoma Division, 2 Wagener Place, St. Lukes, AUCKLAND, Tel. 867 119.
Norway: ELECTRONICA A/S., Vitaminveien 11, P.O. Box 29, Grefsen, OSLO 4, Tel. (02) 15 05 90.
Peru: CADESA, Jr. Ilo, No. 216, Apartado 10132, LIMA, Tel. 27 73 17.
Philippines: ELDAC, Philips Industrial Dev. Inc., 2246 Pasong Tamo, MAKATI-RIZAL, Tel. 86-89-51 to 59.
Portugal PHILIPS PORTUGESA S.A.R.L., Av. Eng. Duharte Pacheco 6, LISBOA 1, Tel. 68 31 21.
Singapore: PHILIPS SINGAPORE PTE LTD., Elcoma Div., POB 340, Toa Payoh CPO, Lorong 1, Toa Payoh, SINGAPORE 12, Tel. 53 88 11.
South Africa: EDAC (Pty.) Ltd., South Park Lane, New Doornfontein, JOHANNESBURG 2001, Tel. 24/6701.
Spain: COPRESA S.A., Balmes 22, BARCELONA 7, Tel. 301 63 12.
Sweden: A.B. ELCOMA, Lidingövägen 50, S-10 250 STOCKHOLM 27, Tel. 08/67 97 80.
Switzerland: PHILIPS A.G., Elcoma Dept., Edenstrasse 20, CH-8027 ZÜRICH, Tel. 01/44 22 11.
Taiwan: PHILIPS TAIWAN LTD., 3rd Fl., San Min Building, 57-1, Chung Shan N. Rd, Section 2, P.O. Box 22978, TAIPEI, Tel. 5513101-5.
Turkey: TÜRK PHILIPS TICARET A.S., EMET Department, Inonu Cad. No. 78-80, ISTANBUL, Tel. 43 59 10.
United Kingdom: MULLARD LTD., Mullard House, Torrington Place, LONDON WC1E 7HD, Tel. 01-580 6633.
United States: (Active devices & Materials) AMPEREX SALES CORP., 230, Duffy Avenue, HICKSVILLE, N.Y. 11802, Tel. (516) 931-6200.
(Passive devices) MEPCO/ELECTRA INC., Columbia Rd., MORRISTOWN, N.J. 07960, Tel. (201) 539-2000.
(IC Products) SIGNETICS CORPORATION, 811 East Arques Avenue, SUNNYVALE, California 94086, Tel. (408) 739-7700.
Uruguay: LUZILECTRON S.A., Rondeau 1567, piso 5, MONTEVIDEO, Tel. 9 43 21.
Venezuela: IND. VENEZOLANAS PHILIPS S.A., Elcoma Dept., A. Ppal de los Ruices, Edif. Centro Colgate, Apdo 1167, CARACAS, Tel. 36 05 11.

A3

Printed in The Netherlands 3-77 9399 509 58301