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

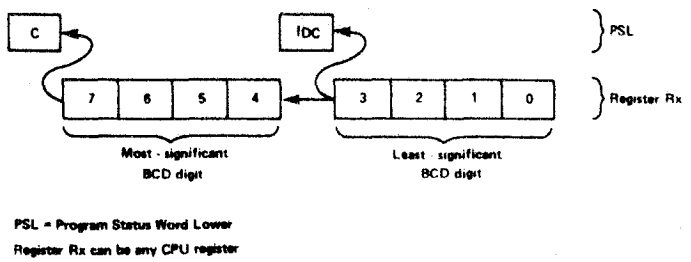

Figure 1

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

Figure 2

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

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

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.

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.

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.

```
        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

```
        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

```
        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

```
        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

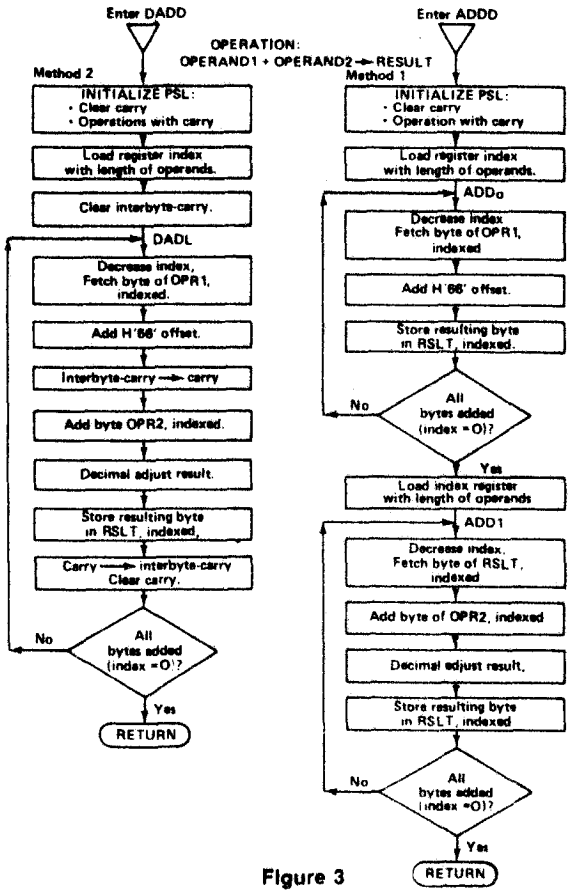

Figure 3

SIGN-MAGNITUDE NOTATION

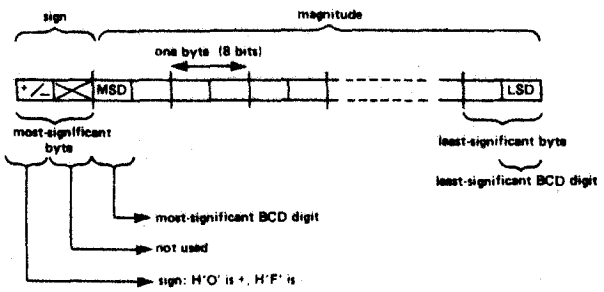

Figure 4

BCD ADDITION AND SUBTRACTION ROUTINES

```
TWIN ASSEMBLER VER 1.0                                  PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                * PD760887
0002                ********************************************
0003                * DECIMAL ADDITION/SUBTRACTION FOR PACKED-BCD *
0004                ********************************************
0005                * OPERATION  OPERAND1 +/- OPERAND2 --> RESULT
0006                * OPERAND1 IS IN  OPR1,OPR1+1,OPR1+2,ETC
0007                * OPERAND2 IS IN  OPR2,OPR2+1,OPR2+2,ETC
0008                * RESULT   IS IN  RSLT,RSLT+1,RSLT+2,ETC
0009                * OPR1,OPR2 AND RSLT ARE MOST-SIGNIFICANT BYTES.
0010                * ALL NUMBERS ARE OF EQUAL LENGTH (IN BYTES)
0011                * LENGTH IS DEFINED BY   LENG
0012                *
0013                * DEFINITIONS OF SYMBOLS
0014                *
0015 0000          R0   EQU    0         PROCESSOR REGISTERS
0016 0001          R1   EQU    1
0017 0002          R2   EQU    2
0018 0003          R3   EQU    3
0019 0000          WC   EQU    H'00'     PSL   1=WITH, 0=WITHOUT CARRY
0020 0001          C    EQU    H'01'           CARRY/BORROW
0021 0003          UN   EQU    3         BRANCH CONDITION  UNCONDITIONAL
0022                *
0023 0005          LENG EQU    5         LENGTH OF OPERANDS/RESULT IN BYTES
0024                *
0025 0000               ORG    H'700'    PARAMETERS
0026                *
0027 0700          OPR1 RES    LENG      OPERAND1
0028 0705          OPR2 RES    LENG      OPERAND2
0029 070A          RSLT RES    LENG      RESULT
0030                *
0031 070F               ORG    H'450'
0032                *
0033                ********************************************
0034                * ADDITION OF UNSIGNED, SINGLE-BYTE BCD NUMBERS *
0035                ********************************************
0036                * OPERATION  OPERAND1 + OPERAND2 --> RESULT
0037                *
0038 0450 0F0700   ADD  LODA,R3 OPR1      FETCH FIRST OPERAND
0039 0453 0766          ADDI,R3 H'66'     ADD OFFSET FOR BCD ADD
0040 0455 0F0705        ADDA,R3 OPR2      ADD SECOND OPERAND
0041 0458 97            DAR,R3            DECIMAL ADJUST RESULT
0042 0459 CF070A        STRA,R3 RSLT      STORE RESULT
0043                *
0044                ********************************************
0045                * SUBTRACTION OF UNSIGNED, SINGLE-BYTE BCD NUMBERS *
0046                ********************************************
0047                * OPERATION  OPERAND1 - OPERAND2 --> RESULT
0048                *
0049 045C 0F0700   SUBT LODA,R3 OPR1      FETCH FIRST OPERAND
0050 045F AF0705        SUBA,R3 OPR2      SUBTRACT SECOND OPERAND
0051 0462 97            DAR,R3            DECIMAL ADJUST RESULT
0052 0463 CF070A        STRA,R3 RSLT      STORE RESULT
0053                *
```

```
TWIN ASSEMBLER VER 1.0                                  PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0055                ********************************************
0056                * ADDITION OF UNSIGNED MULTIPLE-BYTE BCD NUMBERS *
0057                ********************************************
0058                * OPERATION  OPERAND1 + OPERAND2 --> RESULT
0059                *
0060 0466 7501     DADD CPSL   C         CLEAR CARRY
0061 0468 7708          PPSL   WC        ARITHMETIC/ROTATE WITH CARRY
0062 046A 0705          LODI,R3 LENG     LOAD INDEX REGISTER
0063 046C 0500          LODI,R1 0        CLEAR INTERBYTE-CARRY
0064 046E 0F4700   DADL LODA,R0 OPR1,R3,- FETCH BYTE OF OPERAND1
0065 0471 0466          ADDI,R0 H'66'    ADD OFFSET FOR BCD ADD
0066 0473 51            RRR,R1           RESTORE INTERBYTE-CARRY TO C
0067 0474 0F6705        ADDA,R0 OPR2,R3  ADD BYTE OF OPERAND2
0068 0477 94            DAR,R0           DECIMAL ADJUST RESULT
0069 0478 CF670A        STRA,R0 RSLT,R3  STORE RESULTING BYTE
0070 047B D1            RRL,R1           SAVE INTERBYTE-CARRY IN R1, CLEAR C
0071 047C 5B70          BRNR,R3 DADL     BRANCH IF NOT READY
0072                *
0073                ********************************************
0074                * ADDITION OF UNSIGNED MULTIPLE-BYTE BCD NUMBERS *
0075                *       ALTERNATE METHOD                   *
0076                ********************************************
0077                * OPERATION  OPERAND1 + OPERAND2 --> RESULT
0078                *
0079 047E 7501     ADDD CPSL   C         CLEAR CARRY
0080 0480 7708          PPSL   WC        ARITHMETIC WITH CARRY
0081 0482 0705          LODI,R3 LENG     LOAD INDEX REGISTER
0082 0484 0F4700   ADD0 LODA,R0 OPR1,R3,- FETCH BYTE OF OPERAND1
0083 0487 0466          ADDI,R0 H'66'    ADD OFFSET FOR BCD-ADD
0084 0489 CF670A        STRA,R0 RSLT,R3  STORE INTERMEDIATE RESULT
0085 048C 5B76          BRNR,R3 ADD0     BRANCH IF ALL BYTES NOT READY
0086 048E 0705          LODI,R3 LENG     LOAD INDEX REGISTER
0087 0490 0F470A   ADD1 LODA,R0 RSLT,R3,- FETCH BYTE OF INTERMEDIATE SUM
0088 0493 0F6705        ADDA,R0 OPR2,R3  ADD BYTE OF OPERAND2
0089 0496 94            DAR,R0           DECIMAL ADJUST RESULT
0090 0497 CF670A        STRA,R0 RSLT,R3  STORE RESULT
0091 049A 5B74          BRNR,R3 ADD1     BRANCH IF ALL BYTES NOT READY
0092                *
0093                *
0094                ********************************************
0095                * SUBTRACTION OF UNSIGNED MULTIPLE-BYTE BCD NUMBERS *
0096                ********************************************
0097                * OPERATION  OPERAND1 - OPERAND2 --> RESULT
0098                *
0099 049C 7709     DSUB PPSL   WC+C      ARITHMETIC WITH CARRY, CLEAR BORROW
0100 049E 0705          LODI,R3 LENG     LOAD INDEX REGISTER
0101 04A0 0F4700   DSUL LODA,R0 OPR1,R3,- FETCH BYTE OF OPERAND1
0102 04A3 AF6705        SUBA,R0 OPR2,R3  SUBTRACT BYTE OF OPERAND2
0103 04A6 94            DAR,R0           DECIMAL ADJUST RESULT
0104 04A7 CF670A        STRA,R0 RSLT,R3  STORE RESULTING BYTE
0105 04AA 5B74          BRNR,R3 DSUL     BRANCH IF NOT READY
0106                *
0107 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

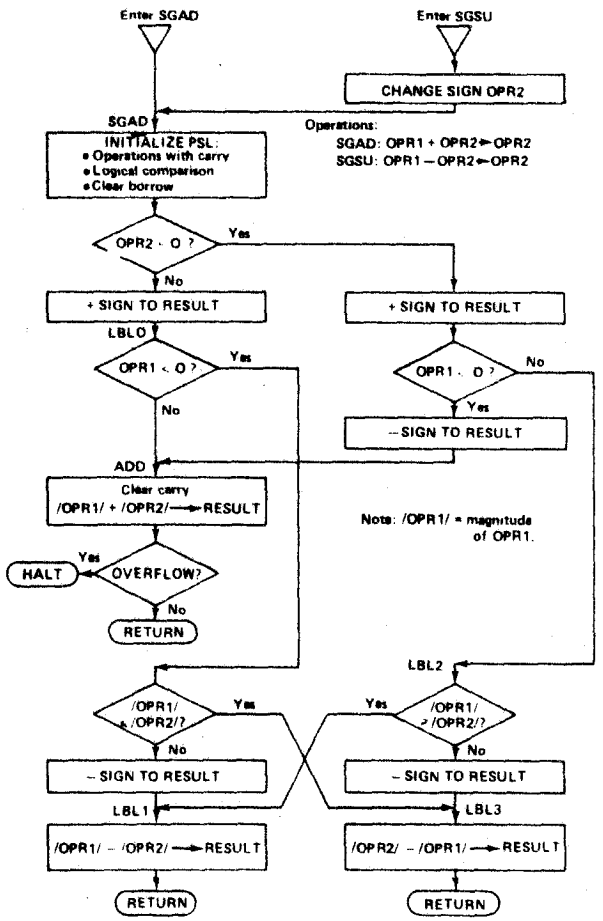

Figure 6

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               * PD760806
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 0000          WC   EQU   H'00'    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     COM2 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
004A 050A 1882          BCTR,EQ COM1    BRANCH IF EQUAL
004B 050C 13            SPSL            PSL TO R0
004C 050D C1            STR2    R1      SAVE PSL IN R1
004D 050E FB74     COM1 BDRR,R3 COM0    BRANCH IF ALL BYTES NOT TESTED
004F 0510 01            LOD2    R1      UPDATE CC WITH STATUS COMPARE
0050 0511 17            RETC,UN         RETURN
0052               *


TWIN ASSEMBLER VER 1.0                                  PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0053               *
0054               ************************************
0055               * SUBTRACTION FOR SIGNED INTEGERS *
0056               ************************************
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 7708     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 9A25          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 0466          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 COM2     COMPARE OPR1 WITH OPR2,
0098               *                       (MAGNITUDES ONLY)
0099 0550 991E          BCFR,GT LBL1     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 COM2     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,R2 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 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

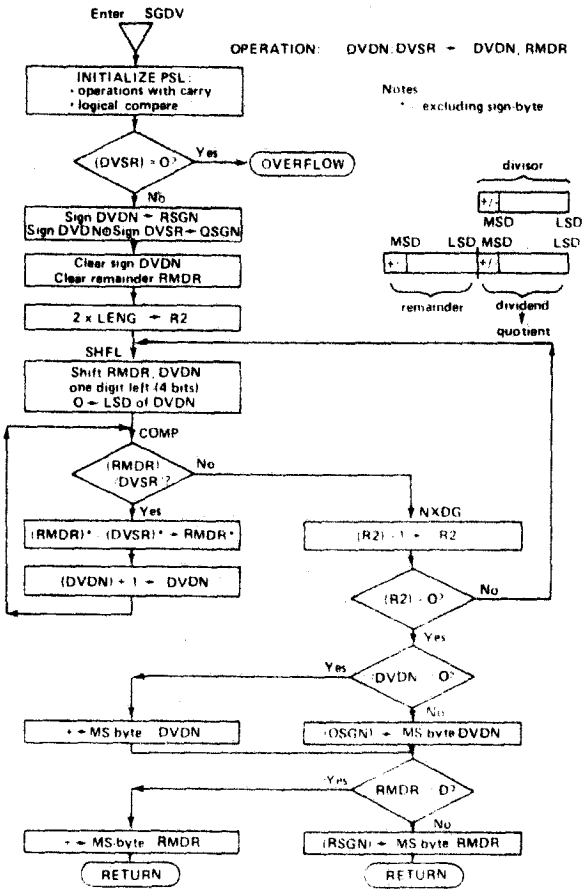

Figure 10

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

0001                * P0760004
0002                ********************************************
0003                * DECIMAL DIVISION FOR SIGNED INTEGERS       *
0004                * NUMBERS ARE IN PACKED BCD. SIGN-MAGNITUDE NOTATION *
0005                ********************************************
0006                * OPERATION
0007                *   DIVIDEND  DIVISOR --> DIVIDEND, REMAINDER
0008                * DIVIDEND  IS IN  DVDN,DVDN+1,DVDN+2, ETC.
0009                * DIVISOR   IS IN  DVSR,DVSR+1,DVSR+2, ETC.
0010                * QUOTIENT  IS IN  DVDN,DVDN+1,DVDN+2, ETC.
0011                * REMAINDER IS IN  RMDR,RMDR+1,RMDR+2, ETC.
0012                * DIVIDEND IS DESTROYED AFTER DIVISION
0013                * DVDN,DVSR AND RMDR ARE MOST-SIGNIFICANT BYTES
0014                * LENGTH OF NUMBERS (IN BYTES) IS DEFINED BY  LENG
0015                *   ALLOWED RANGE   1 < LENG < 65
0016                * MS-BYTE HOLDS SIGN INFORMATION  H'00' FOR +, H'F0' FOR -
0017                *
0018                * DEFINITIONS OF SYMBOLS
0019                *
0020 0000           R0   EQU   0        PROCESSOR REGISTERS
0021 0001           R1   EQU   1
0022 0002           R2   EQU   2
0023 0003           R3   EQU   3
0024 0000           WC   EQU   H'00'    PSL   1=WITH, 0=WITHOUT CARRY
0025 0002           COM  EQU   H'02'          1=LOGIC, 0=ARITH COMPARE
0026 0001           C    EQU   H'01'          CARRY/BORROW
0027 0000           Z    EQU   0        BRANCH COND    ZERO
0028 0001           P    EQU   1                       POSITIVE
0029 0002           N    EQU   2                       NEGATIVE
0030 0000           EQ   EQU   0                       EQUAL
0031 0002           LT   EQU   2                       LESS THAN
0032 0003           UN   EQU   3                       UNCONDITIONAL
0033                *
0034                * PARAMETERS *
0035                *
0036 0005           LENG EQU   5        LENGTH OF OPERANDS (IN BYTES)
0037                *
0038 0000                ORG   H'700'
0039                *
0040 0700           RMDR RES   LENG     REMAINDER
0041 0705           DVDN RES   LENG     DIVIDEND
0042                * NOTE  RMDR AND DVDN MUST BE IN SUCCESSIVE
0043                *       RAM LOCATIONS, BECAUSE OF DOUBLE-LENGTH SHIFT
0044 070A           DVSR RES   LENG     DIVISOR
0045 070F           TEMP RES   2        TEMPORARY STORAGE FOR ADDRESS
0046 0711           QSGN RES   1        QUOTIENT SIGN
0047 0712           RSGN RES   1        REMAINDER SIGN
0048                *


TWIN ASSEMBLER VER 1.0                                  PAGE 0002

LINE ADDR  OBJECT  E SOURCE

0050 0713                ORG   H'500'
0051                *
0052                ****************************************
0053                * SUBROUTINE TO TEST OPERAND FOR ZERO *
0054                ****************************************
0055                * OPERAND ADDRESS MUST BE IN R0,R1 (HIGH,LOW ADDR.)
0056                * ALL BYTES  EXCEPT MS-BYTE (=SIGN) ARE TESTED
0057                * CONDITION CODE BECOMES 00 IF OPERAND WAS ZERO.
0058                *
0059 0500 CC070F    TZER STRA,R0 TEMP       SAVE OPERAND ADDRESS
0060 0503 CC0710         STRA,R1 TEMP+1
0061 0506 0704           LODI,R3 LENG-1     LOAD INDEX REGISTER
0062 0508 0FE70F    TZE0 LODA,R0 *TEMP,R3   FETCH BYTE OF OPERAND
0063 050B 15             RETC,P             RETURN IF POSITIVE (CC=01)
0064 050C 16             RETC,N             RETURN IF NEGATIVE (CC=10)
0065 050D FB79           BDRR,R3 TZE0       BRANCH IF ALL NOT TESTED
0066 050F 17             RETC,UN            RETURN WITH CC=00
0067                *
0068                *                       ********************
0069                *                       * DIVISION PROGRAM *
0070                *                       ********************
0071                *
0072 0510 770A      SGDV PPSL   WC+COM      OPERATIONS WITH CARRY,
0073                *                         LOGICAL COMPARISON
0074 0512 0407           LODI,R0 <DVSR      HIGH-ADDRESS DIVISOR TO R0
0075 0514 050A           LODI,R1 >DVSR      LOW- ADDRESS DIVISOR TO R1
0076 0516 3F0500         BSTA,UN TZER       TEST DIVISOR FOR ZERO
0077 0519 1C0595         BCTA,Z  OVFL       BRANCH IF ZERO
0078 051C 0C0705         LODA,R0 DVDN       FETCH SIGN DIVIDEND
0079 051F CC0712         STRA,R0 RSGN       SAVE IN REMAINDER SIGN
0080 0522 2C070A         EORA,R0 DVSR       TAKE EX-OR WITH DVSR SIGN
0081 0525 CC0711         STRA,R0 QSGN       SAVE IN QUOTIENT SIGN
0082 0528 20             EORZ    R0         CLEAR R0
0083 0529 0706           LODI,R3 LENG+1     LOAD INDEX REGISTER
0084 052B CF4700    CLRM STRA,R0 RMDR,R3,-  CLEAR REMAINDER AND SIGN DVDN
0085 052E 5B7B           BRNR,R3 CLRM       BRANCH IF NOT DONE
0086                *
0087 0530 060A           LODI,R2 LENG+LENG  NUMBER OF DIGITS TO LOOP COUNTER
0088                *
0089                *                       SHIFT RMDR/DVDN 4 BITS LEFT
0090                *                         INSERTING ZEROES IN LS-BITS
0091 0532 0504      SHFL LODI,R1 4          LOAD BIT COUNTER
0092 0534 7501      SHF0 CPSL   C           CLEAR CARRY
0093 0536 070A           LODI,R3 LENG+LENG  LOAD INDEX REGISTER
0094 0538 0F4700    SHF1 LODA,R0 RMDR,R3,-  FETCH BYTE OF RMDR/DVDN
0095 053B D0             RPL,R0             ROTATE LEFT WITH CARRY
0096 053C CF6700         STRA,R0 RMDR,R3    RESTORE SHIFTED BYTE
0097 053F 5B77           BRNR,R3 SHF1       BRANCH IF ALL NOT SHIFTED
0098 0541 F971           BDRR,R1 SHF0       BRANCH IF 4 BITS NOT SHIFTED


TWIN ASSEMBLER VER 1.0                                  PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0100                *                       COMPARE RMDR AND DVSR TO TEST
0101                *                         IF SUBTRACTION IS POSSIBLE
0102 0543 0500      COMP LODI,R1 0          CLEAR R1, MS-BIT OF R1 BECOMES
0103                *                         1 FOR RMDR < DVSR
0104 0545 0704           LODI,R3 LENG-1     LOAD INDEX REGISTER
0105 0547 0F6700    COMB LODA,R0 RMDR,R3    FETCH BYTE OF REMAINDER
0106 054A 076700         COMA,R0 DVSR,R3    COMPARE WITH BYTE OF DIVISOR
0107 054D 1B02           BCTR,EQ COM1       BRANCH IF EQUAL
0108 054F 13             SPSL               PSL TO R0
0109 0550 C1             STRZ    R1         SAVE PSL IN R1
0110 0551 FB74      COM1 BDRR,R3 COMB       BRANCH IF ALL BYTES NOT TESTED
0111 0553 01             LODZ    R1         FETCH STATUS OF COMPARISON
0112 0554 1A1A           BCTR,LT NXDG       BRANCH IF RMDR < DVSR
0113                *
0114                *                       SUBTRACT DIVISOR FROM
0115                *                         REMAINDER WITHOUT MS-BYTES.
0116 0556 7701           PPSL    C          CLEAR BORROW
0117 0558 0704           LODI,R3 LENG-1     LOAD INDEX REGISTER
0118 055A 0F6700    SUBD LODA,R0 RMDR,R3    FETCH BYTE OF REMAINDER
0119 055D 0F670A         SUBA,R0 DVSR,R3    SUBTRACT BYTE OF DIVISOR
0120 0560 34             DAR,R0             DECIMAL ADJUST RESULT
0121 0561 CF6700         STRA,R0 RMDR,R3    RESTORE IN REMAINDER
0122 0564 FB74           BDRR,R3 SUBD       BRANCH IF NOT READY
0123                *
0124 0566 0C0709         LODA,R0 DVDN+LENG-1  FETCH LS-BYTE QUOTIENT
0125 0569 D000           BIRR,R0 $+2        INCREASE R0
0126 056B CC0709         STRA,R0 DVDN+LENG-1 RESTORE INCREMENTED QUOTIENT
0127 056E 1B53           BCTR,UN COMP       BRANCH FOR NEXT COMPARISON
0128                *
0129 0570 FA40      NXDG BRNR,R2 SHFL       BRANCH IF DIVISION NOT READY
0130 0572 0E0711         LODA,R2 QSGN       FETCH SIGN QUOTIENT
0131 0575 0407           LODI,R0 <DVDN      HIGH-ADDRESS QUOTIENT TO R0
0132 0577 0505           LODI,R1 >DVDN      LOW- ADDRESS QUOTIENT TO R1
0133 0579 3F0500         BSTA,UN TZER       TEST IF QUOTIENT IS ZERO
0134 057C 9802           BCFR,Z  STQS       BRANCH IF NOT ZERO
0135 057E 0600           LODI,R2 0          CLEAR R0
0136 0580 CE0705    STQS STRA,R2 DVDN       STORE SIGN IN MS-BYTE DVDN
0137 0583 0E0712         LODA,R2 RSGN       FETCH REMAINDER SIGN
0138 0586 0407           LODI,R0 <RMDR      HIGH-ADDRESS REMAINDER TO R0
0139 0588 0500           LODI,R1 >RMDR      LOW- ADDRESS REMAINDER TO R1
0140 058A 3F0500         BSTA,UN TZER       TEST IF REMAINDER IS ZERO
0141 058D 9802           BCFR,Z  STRS       BRANCH IF NOT ZERO
0142 058F 0600           LODI,R2 0          CLEAR R2
0143 0591 CE070A    STRS STRA,R2 RMDR       STORE SIGN IN MS-BYTE RMDR
0144 0594 17             RETC,UN            RETURN
0145                *
0146 0595 40        OVFL HALT               OVERFLOW LOCATION
0147                *
0148 0000                END   0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 11