COP820CJ,COP840CJ,COP880C,COP884BC, COP888CF,COP888CL,COP888EK,COP888FH, COP888GW,COP8ACC5,COP8AME9,COP8CBE9, COP8CBR9,COP8CCE9,COP8CCR9,COP8CDR9, COP8SAA7,COP8SAC7,COP8SBR9,COP8SCR9, COP8SDR9,COP8SGE5,COP8SGE7,COP8SGG5, COP8SGH5,COP8SGK5,COP8SGR5,COP8SGR7, COP912C

AN-596 COP800 Mathpak

TEXAS INSTRUMENTS

Literature Number: SNOA110

COP800 MathPak

National Semiconductor
Application Note 596
Verne H. Wilson
June 1989

OVERVIEW

This application note discusses the various arithmetic operations for National Semiconductor's COP800 family of 8-bit microcontrollers. These arithmetic operations include both binary and BCD (Binary Coded Decimal) operation. The four basic arithmetic operations (add, subtract, multiply, divide) are outlined in detail, with several examples shown for both binary and BCD addition and subtraction. Multiplication, division, and BCD conversion algorithms are also provided. Both BCD to binary and binary to BCD conversion subroutines are included, as well as the various multiplication and division subroutines.

Four sets of optimal subroutines are provided for

1. Multiplication
2. Division
3. Decimal (Packed BCD) to binary conversion
4. Binary to decimal (Packed BCD) conversion

One class of subroutines is optimized for minimal COP800 program code, while the second class is optimized for minimal execution time in order to optimize throughput time.

This application note is organized in four different sections. The first section outlines various addition and subtraction routines, including both binary and BCD (Binary Coded Decimal). The second section outlines the multiplication algorithm and provides several optimal multiply subroutines for 1, 2, 3, and 4 byte operation. The third section outlines the division algorithm and provides several optimal division subroutines for 1, 2, 3, and 4 byte operation. The fourth section outlines both the decimal (Packed BCD) to binary and binary to decimal (Packed BCD) conversion algorithms. This section provides several optimal subroutines for these BCD conversions.

The COP800 arithmetic instructions include the Add (ADD), Add with Carry (ADC), Subtract with Carry (SUBC), Increment (INCR), Decrement (DECR), Decimal Correct (DCOR), Clear Accumulator (ACC), Set Carry (SC), and Reset Carry (RC). The shift and rotate instructions, which include the Rotate Right through Carry (RRC) and the Swap Accumulator Nibbles (SWAP), may also be considered as arithmetic instruction variations. The RRC instruction is instrumental in writing a fast multiply routine.

1.0 BINARY AND BCD ADDITION AND SUBTRACTION

In subtraction, a borrow is represented by the absence of a carry and vice versa. Consequently, the carry flag needs to be set (no borrow) before a subtraction, just as the carry flag is reset before an addition. The ADD instruction does not use the carry flag as an input, nor does it change the carry flag. It should also be noted that both the carry and half carry flags (bits 6 and 7, respectively, of the PSW control register) are cleared with reset, and remain unchanged with the ADD, INC, DEC, DCOR, CLR and SWAP instructions. The DCOR instruction uses both the carry and half carry flags. The SC instruction sets both the carry and half carry flags, while the RC instruction resets both these flags.

The following program examples illustrate additions and subtractions of 4-byte data fields in both binary and BCD (Binary Coded Decimal). The four bytes from data memory locations 24 through 27 are added to or subtracted from the four bytes in data memory locations 16 through 19. The results replace the data in memory locations 24 through 27.

These operations are performed both in Binary and BCD. It should be noted that the BCD pre-conditioning of Adding (ADD) the hex 66 is only necessary with the BCD addition, not with the BCD subtraction. The (Binary Coded Decimal) DCOR (Decimal Correct) instruction uses both the carry and half carry flags as inputs, but does not change the carry and half carry flags. Also note that the #12 with the IFBNE instruction represents 28 − 16, since the IFBNE operand is modulo 16 (remainder when divided by 16).

 RRD-B30M75/Printed in U. S. A.

BINARY ADDITION:

```
        LD      X,#16       ;  NO LEADING ZERO
        LD      B,#24       ;     INDICATES DECIMAL
        RC                  ;  RESET CARRY TO START
LOOP:   LD      A,[X+]      ;  [X] TO ACC
        ADC     A,[B]       ;  ADD [B] TO ACC
        X       A,[B+]      ;  RESULT TO [B]
        IFBNE   #12         ;  IF STILL IN DATA FIELD
        JP      LOOP        ;     JUMP BACK TO REPEAT LOOP
        IFC                 ;  IF TERMINAL CARRY,
        JP      OVFLOW      ;     JUMP TO OVERFLOW
```

BINARY SUBTRACTION:

```
        LD      X,#010      ;  LEADING ZERO
        LD      B,#018      ;     INDICATES HEX
        SC                  ;  RESET BORROW TO START
LOOP:   LD      A,[X+]      ;  [X] TO ACC
        SUBC    A,[B]       ;  SUBTRACT [B] FROM ACC
        X       A,[B+]      ;  RESULT TO [B]
        IFBNE   #12         ;  IF STILL IN DATA FIELD
        JP      LOOP        ;     JUMP BACK TO REPEAT LOOP
        IFNC                ;  IF TERMINAL BORROW,
        JP      NEGRSLT     ;     JUMP TO NEGATIVE RESULT
```

BCD ADDITION:

```
        LD      X,#010      ;  LEADING ZERO
        LD      B,#018      ;     INDICATES HEX
        RC                  ;  RESET CARRY TO START
LOOP:   LD      A,[X+]      ;  [X] TO ACC
        ADD     A,#066      ;  ADD HEX 66 TO ACC
        ADC     A,[B]       ;  ADD [B] TO ACC
        DCOR    A           ;  DECIMAL CORRECT RESULT
        X       A,[B+]      ;  RESULT TO [B]
        IFBNE   #12         ;  IF STILL IN DATA FIELD
        JP      LOOP        ;     JUMP BACK TO REPEAT LOOP
        IFC                 ;  IF TERMINAL CARRY
        JP      OVFLOW      ;     JUMP TO OVERFLOW
```

BCD SUBTRACTION:

```
        LD      X,#16       ;  NO LEADING ZERO
        LD      B,#24       ;     INDICATES DECIMAL
        C
LOOP:   LD      A,[X+]      ;  [X] TO ACC
        SUBC    A,[B]       ;  SUBTRACT [B] FROM ACC
        DCOR    A           ;  DECIMAL CORRECT RESULT
        X       A,[B+]      ;  RESULT TO [B]
        IFBNE   #12         ;  IF STILL IN DATA FIELD
        JP      LOOP        ;     JUMP BACK TO REPEAT LOOP
        IFNC                ;  IF TERMINAL BORROW
        JP      NEGRSLT     ;     JUMP TO NEGATIVE RESULT
```

The astute observer will notice that these previous additions and subtractions are not ''adding machine'' type arithmetic operations in that the result replaces the second operand rather than the first. The following program examples illustrate ''adding machine'' type operation where the result replaces the first operand. With subtraction, this entails the result replacing the minuend rather than the subtrahend. Note that the B and X pointers are now reversed.

BINARY ADDITION:

```
         LD        B,#16              ;  B POINTER AT FIRST OPERAND
         LD        X,#24              ;  X POINTER AT SECOND OPERAND
         RC                           ;  RESET CARRY TO START
LOOP:    LD        A,[X+]             ;  [X] TO ACC
         ADC       A,[B]              ;  ADD [B] TO ACC
         X         A,[B+]             ;  RESULT TO [B]
         IFBNE     #4                 ;  IF STILL IN DATA FIELD
         JP        LOOP               ;     JUMP BACK TO REPEAT LOOP
         IFC                          ;  IF TERMINAL CARRY
         JP        OVFLOW             ;     JUMP TO OVERFLOW
```

BINARY SUBTRACTION:

```
         LD        B,#010             ;  B POINTER AT FIRST OPERAND
         LD        X,018              ;  X POINTER AT SECOND OPERAND
         SC                           ;  RESET BORROW TO START
LOOP:    LD        A,[X+]             ;  [X] TO ACC
         X         A,[B]              ;  EXCHANGE [B] AND ACC
         SUBC      A,[B]              ;  SUBTRACT [B] FROM ACC
         X         A,[B+]             ;  RESULT TO [B]
         IFBNE     #4                 ;  IF STILL IN DATA FIELD
         JP        LOOP               ;     JUMP BACK TO REPEAT LOOP
         IFNC                         ;  IF TERMINAL BORROW
         JP        NEGRSLT            ;     JUMP TO NEGATIVE RESULT
```

BCD ADDITION:

```
         LD        B,#010             ;  B POINTER AT FIRST OPERAND
         LD        X,#018             ;  X POINTER AT SECOND OPERAND
         RC                           ;  RESET CARRY TO START
LOOP:    LD        A,[X+]             ;  [X] TO ACC
         ADD       A,#066             ;  ADD HEX66 TO ACC
         ADC       A,[B]              ;  ADD [B] TO ACC
         DCOR      A                  ;  DECIMAL CORRECT RESULT
         X         A,[B+]             ;  RESULT TO [B]
         IFBNE     #4                 ;  IF STILL IN DATA FIELD
         JP        LOOP               ;     JUMP BACK TO REPEAT LOOP
         IFC       ;                  ;  IF TERMINAL CARRY
         JP        OVFLOW             ;     JUMP TO OVERFLOW
```

BCD SUBTRACTION:

```
         LD        B,#16              ;  B POINTER AT FIRST OPERAND
         LD        X,#24              ;  X POINTER AT SECOND OPERAND
         SC                           ;  RESET BORROW TO START
LOOP:    LD        A,[X+]             ;  [X] TO ACC
         X         A,[B]              ;  EXCHANGE [B] AND ACC
         SUBC      A,[B]              ;  SUBTRACT [B] FROM ACC
         DCOR      A                  ;  DECIMAL CORRECT RESULT
         X         A,[B+]             ;  RESULT TO [B]
         IFBNE     #4                 ;  IF STILL IN DATA FIELD
         JP        LOOP               ;     JUMP BACK TO REPEAT LOOP
         IFNC                         ;  IF TERMINAL BORROW
         JP        NEGRSLT            ;     JUMP TO NEGATIVE RESULT
```

Let us now consider a hybrid arithmetic example, where we wish to add five successive bytes of a data table in ROM program memory to a two byte sum, and then subtract the SUM result from a two byte total TOT. Let us further assume that the ROM table is located starting at program memory address 0401, while SUM and TOT are at RAM data memory locations [1, 0] and [3, 2] respectively, and that we wish to encode the program as a subroutine.

ROM Table:

```
. = 0401
. Byte 102
. Byte 41
. Byte 31
. Byte 26
. Byte 5
```

ROM Table Accessed Top Down

```
SUMLO = 0
SUMHI = 1
TOTLO = 2
TOTHI = 3
```

```
ARITH1: LD      X,#5        ; SET UP ROM TABLE POINTER
        LD      B,#0        ; SET UP SUM POINTER
LOOP:   RC                  ; RESET CARRY TO START ADDITION
        LD      A,X         ; ROM POINTER TO ACC
        LAID                ; TABLE VALUE FROM ROM TO ACC
        ADC     A,[B]       ; ADD SUMLO TO ACC
        X       A,[B+]      ; RESULT TO SUMLO
        CLR     A           ; CLEAR ACC
        ADC     A,[B]       ; ADD SUMHI TO ACC
        X       A,[B−]      ; RESULT TO SUMHI
        DRSZ    X           ; DECR AND TEST ROM PTR FOR ZERO
        JP      LOOP        ; JUMP BACK TO REPEAT LOOP
                            ;    IF X PTR NOT ZERO
        SC                  ; RESET BORROW TO START SUBTRACTION
        LD      B,#2        ; SET UP TOT POINTER
LUP:    LD      A,[X+]      ; SUBTRAHEND (SUM) TO ACC
        X       A,[B]       ; REVERSE OPERANDS
        SUBC    A,[B]       ;    FOR SUBTRACTION
        X       A,[B+]      ; RESULT TO TOT
        IFBNE   #4          ; IF STILL IN TOT FIELD
        JP      LUP         ;    JUMP BACK TO REPEAT LUP
        RET                 ; RETURN FROM SUBROUTINE
```

2.0 MULTIPLICATION

The COP800 multiplications are all based on starting the multiplier in the low order end of the double length product space. The high end of the double length product space is initially cleared, and then the double length product is shifted right one bit. The bit shifted out from the low order end represents the low order bit of the multiplier. If this bit is a "1", the multiplicand is added to the high end of the double length product space. The entire shifting process and the conditional addition of the multiplicand to the upper end of the double length product is then repeated. The number of shift cycles is equal to the number of bit positions in the multiplier plus one extra shift cycle. This extra terminal shift cycle is necessary to correctly align the resultant product.

Note that an M byte multiplicand multiplied by an N byte multiplier will result in an M + N byte double length product. However, these multiplication subroutines will only use 2M + N + 1 bytes of RAM memory space, since the multiplier initially occupies the low order end of the double length product. The one extra byte is necessary for the shift counter CNTR.

The minimal code (28 byte) general multiplication subroutine is shown with two different examples, MY2448 and MY4824. Both examples multiply 24 bits by 48 bits. The MY2448 subroutine uses the 48-bit operand as the multiplier, and consequently uses minimal RAM as well as minimal program code. The MY4824 subroutine uses the 24-bit operand as the multiplier, and consequently executes considerably faster than the minimal RAM MY2448 subroutine.

MPY88
- — 8 by 8 Multiplication Subroutine
- — 19 Bytes
- — 180 Instruction Cycles
- — Minimum Code

MLT88
- — Fast 8 by 8 Multiplication Subroutine
- — 42 Bytes
- — 145 Instruction Cycles

VFM88
- — Very Fast 8 by 8 Multiply Subroutine
- — 96 Bytes
- — 116 Instruction Cycles

MPY168
- — Fast 16 by 8 Multiplication Subroutine
- — 36 Bytes
- — 230 Instruction Cycles Average
- — 254 Instruction Cycles Maximum

MPY816 (or MPY824, MPY832)
- — 8 by 16 (or 24, 32) Multiply Subroutine
- — 22 Bytes
- — 589 (or 1065, 1669) Instruction Cycles Average
- — 597 (or 1077, 1685) Instruction Cycles Maximum
- — Minimum Code, Minimum RAM
- — Extendable Routine for MPY8XX by Changing Parameters, with Number of Bytes (22) Remaining a Constant

MPY248
- — Fast 24 by 8 Multiplication Subroutine
- — 47 Bytes
- — 289 Instruction Cycles Average
- — 333 Instruction Cycles Maximum

MX1616
- — Fast 16 by 16 Multiplication Subroutine
- — 39 Bytes
- — 498 Instruction Cycles Average
- — 546 Instruction Cycles Maximum

MP1616
- — 16 by 16 Multiplicand Subroutine
- — 29 Bytes
- — 759 Instruction Cycles Average
- — 807 Instruction Cycles Maximum
- — Almost Minimum Code

MY1616 (or MY1624, MY1632)
- — 28 Bytes
- — 16 by 16 (or 24, 32) Multiply Subroutine
- — 861 (or 1473, 2213) Inst. Cycles Average
- — 1029 (or 1725, 2549) Inst. Cycles Maximum
- — Minimum Code, Minimum RAM
- — Extendable Routne for MY16XX by Changing Parameters, with Number of Bytes (28) Remaining a Constant

Minimal general multiplication subroutine for any number of bytes in multiplicand and multiplier
- — 28 Bytes
- — Minimum Code
- — MY2448 Used as First Example, with Minimum RAM and 4713 Instruction Cycles Average 5457 Instruction Cycles Maximum
- — MY4824 Used as Second Example, with Non Minimal RAM and 2751 Instruction Cycles Average 3483 Instruction Cycles Maximum

MPY88—8 BY 8 MULTIPLICATION SUBROUTINE

```
        MINIMUM CODE
        19 BYTES
        180 INSTRUCTION CYCLES
        MULTIPLICAND IN [0]        (ICAND)
        MULTIPLIER IN [1]          (IER)
        PRODUCT IN [2,1]           (PROD)
MPY88:  LD          CNTR,#9        ;  LD CNTR WITH LENGTH OF
        RC                         ;     MULTIPLIER FIELD + 1
        LD          B,#2
        CLR         A              ;  CLEAR UPPER PRODUCT
M88LUP: RRC         A              ;  RIGHT SHIFT
        X           A,[B-]         ;     UPPER PRODUCT
        LD          A,[B]
        RRC         A              ;  RIGHT SHIFT LOWER
        X           A,[B-]         ;     PRODUCT/MULTIPLIER
        CLR         A              ;  CLR ACC AND TEST LOW
        IFC                        ;     ORDER MULTIPLER BIT
        LD          A,[B]          ;  MULTIPLICAND TO ACC IF
        RC                         ;     LOW ORDER BIT = 1
        LD          B,#2           ;  ADD MULTIPLICAND TO
        ADC         A,[B]          ;     UPPER PRODUCT
        DRSZ        CNTR           ;  DECREMENT AND TEST
        JP          M88LUP         ;     CNTR FOR ZERO
        RET                        ;  RETURN FROM SUBROUTINE
```

MLT88—FAST 8 BY 8 MULTIPLICATION SUBROUTINE

```
        42 BYTES
        145 INSTRUCTION CYCLES
        MULTIPLICAND IN [0]         (ICAND)
        MULTIPLIER IN [1]           (IER)
        PRODUCT IN [2,1]            (PROD)
MLT88:  LD          CNTR,#3         ; LOAD CNTR WITH
        RC                          ;  1/3 OF LENGTH OF
        LD          B,#2            ;  (MULTIPLIER FIELD + 1)
        CLR         A               ; CLEAR UPPER PRODUCT
;
ML88LP: RRC         A               ; RIGHT SHIFT   ***
        X           A,[B-]          ;    UPPER PRODUCT
        LD          A,[B]
        RRC         A               ; RIGHT SHIFT LOWER
        X           A,[B-]          ;    PRODUCT/MULTIPLIER
        CLR         A               ; CLR ACC AND TEST LOW
        IFC                         ;    ORDER MULTIPLIER BIT
        LD          A,[B]           ; MULTIPLICAND TO ACC IF
        RC                          ;    LOW ORDER BIT = 1
        LD          B,#2            ; ADD MULTIPLICAND TO
        ADC         A,[B]           ;    UPPER PRODUCT ***
;
        RRC         A               ; REPEAT THE ABOVE
        X           A,[B-]          ;    11 BYTE
        LD          A,[B]           ;    13 INSTRUCTION
        RRC         A               ;    CYCLE PROGRAM
        X           A,[B-]          ;    SECTION (WITH
        CLR         A               ;    THE *** DELIMITERS)
        IFC                         ;    TWICE MORE FOR A
        LD          A,[B]           ;    TOTAL OF THREE TIMES
        RC
        LD          B,#2
        ADC         A,[B]           ; END OF SECOND REPEAT
;
        RRC         A               ; START OF THIRD REPEAT
        X           A,[B-]
        LD          A,[B]
        RRC         A
        X           A,[B-]
        CLR         A
        IFC
        LD          A,[B]
        RC
        LD          B,#2
        ADC         A,[B]           ; END OF THIRD REPEAT
;
        DRSZ        CNTR            ; DECREMENT AND TEST
        JMP         ML88LP          ;  CNTR FOR ZERO
        RET                         ; RETURN FROM SUBROUTINE
```

VFM88—VERY FAST 8 BY 8 MULTIPLY SUBROUTINE

```
        96 BYTES
        116 INSTRUCTION CYCLES

        MULTIPLICAND IN [0]       (ICAND)
        MULTIPLIER IN [1]         (IER)
        PRODUCT IN [2,1]          (PROD)
VFM88:  RC
        LD          B,#2
        LD          [B-],#0       ; CLEAR UPPER PRODUCT
        LD          A,[B]
        RRC         A             ; RIGHT SHIFT LOWER
        X           A,[B-]        ;  PRODUCT/MULTIPLIER
        CLR         A             ; CLR ACC AND TEST LOW
        IFC                       ;    ORDER MULTIPLIER BIT
        LD          A,[B]         ; MULTIPLICAND TO ACC IF
        RC                        ;    LOW ORDER BIT = 1
        LD          B,#2          ; ADD MULTIPLICAND TO
        ADC         A,[B]         ;    UPPER PRODUCT
;
        RRC         A             ; RIGHT SHIFT   ***
        X           A,[B-]        ;    UPPER PRODUCT
        LD          A,[B]
        RRC         A             ; RIGHT SHIFT LOWER
        X           A,[B-]        ;    PRODUCT/MULTIPLIER
        CLR         A             ; CLR ACC AND TEST LOW
        IFC                       ;    ORDER MULTIPLIER BIT
        LD          A,[B]         ; MULTIPLICAND TO ACC IF
        RC                        ;    LOW ORDER BIT = 1
        LD          B,#2          ; ADD MULTIPLICAND TO
        ADC         A,[B]         ;    UPPER PRODUCT ***
;
;   THE ABOVE 11 BYTE, 13 INSTRUCTION CYCLE SECTION WITH THE ***
;   DELIMITERS REPRESENTS THE PROCESSING FOR ONE MULTIPLIER BIT.
;
;
;       ---                       ; REPEAT THE
;                                 ; ABOVE SECTION
;       ---                       ; SIX MORE TIMES,
;                                 ; FOR A TOTAL
;       ---                       ; OF SEVEN TIMES
;
        RRC         A             ; RIGHT SHIFT
        X           A,[B-]        ; UPPER PRODUCT
        LD          A,[B]
        RRC         A             ; RIGHT SHIFT LOWER
        X           A,[B]         ;    PRODUCT/MULTIPLIER
        RET                       ; RETURN FROM SUBROUTINE
;
;
;
```

MPY168—FAST 16 BY 8 MULTIPLICATION SUBROUTINE

```
         36 BYTES
         230 INSTRUCTION CYCLES AVERAGE
         254 INSTRUCTION CYCLES MAXIMUM

         MULTIPLICAND IN [1,0]        (ICAND)
         MULTIPLIER IN [2]            (IER)
         PRODUCT IN [4,3,2]           (PROD)
MPY168:  LD          CNTR,#9          ;  LD CNTR WITH LENGTH OF
         RC                           ;   MULTIPLIER FIELD + 1
         LD          B,#4
         LD          [B-],#0          ;  CLEAR
         LD          [B-],#0          ;     UPPER PRODUCT
         JP          MP168S
M168LP:  RRC         A                ;  RIGHT SHIFT UPPER
         X           A,[B-]           ;   BYTE OF PRODUCT
         LD          A,[B]
         RRC         A                ;  RIGHT SHIFT MIDDLE
         X           A,[B-]           ;   BYTE OF PRODUCT
MP168S:  LD          A,[B]
         RRC         A                ;  RIGHT SHIFT LOWER
         X           A,[B]            ;     PRODUCT/MULTIPLIER
         IFNC                         ;  TEST LOWER BIT
         JP          MP168T           ;     OF MULTIPLIER
         RC                           ;  CLEAR CARRY
         LD          B,#0             ;  LOWER BYTE OF
         LD          A,[B]            ;     MULTIPLICAND TO ACC
         LD          B,#3             ;  ADD LOWER BYTE OF
         ADC         A,[B]            ;     MULTIPLICAND TO
         X           A,[B]            ;     MIDDLE BYTE OF PROD
         LD          B,#1             ;  UPPER BYTE OF
         LD          A,[B]            ;     MULTIPLICAND TO ACC
         LD          B,#4             ;  ADD UPPER BYTE OF ICAND
         ADC         A,[B]            ;     TO UPPER BYTE OF PROD
         DRSZ        CNTR             ;  DECREMENT CNTR AND JUMP
         JP          M168LP           ;     BACK TO LOOP; CNTR
                                      ;     CANNOT EQUAL ZERO
MP168T:  LD          B,#4             ;  HIGH ORDER PRODUCT
         LD          A,[B]            ;     BYTE TO ACC
         DRSZ        CNTR             ;  DECREMENT AND TEST IF
         JP          M168LP           ;     CNTR EQUAL TO ZERO
         RET                          ;  RETURN FROM SUBROUTINE
```

MPY816—(OR MPY824, MPY832) 8 BY 16 (OR 24, 32) MULTIPLY SUBROUTINE

```
         MINIMUM CODE, MINIMUM RAM
         22 BYTES
         589 (OR 1065, 1669) INSTR. CYCLES AVERAGE
         597 (OR 1077, 1685) INSTR. CYCLES MAXIMUM
         EXTENDABLE ROUTINE FOR MPY8XX BY CHANGING
          PARAMETERS, WITH NUMBER OF BYTES (22)
          REMAINING A CONSTANT.

         MULTIPLICAND IN [0]              (ICAND)
         MULTIPLIER IN [2,1] FOR 16 BIT (IER)
                     OR [3,2,1] for 24 BIT
                     OR [4,3,2,1] for 32 BIT
         PRODUCT IN [3,2,1] FOR 16 BIT    (PROD)
                     OR [4,3,2,1] FOR 24 BIT
                     OR [5,4,3,2,1] FOR 32 BIT

MPY816:  LD          CNTR,#17             ; LD CNTR WITH LENGTH OF
                                          ;  MULTIPLIER FIELD + 1
                                          ;  #17 FOR MPY816 16 BIT
                                          ;  (#25 FOR MPY824 24 BIT)
                                          ;  (#33 FOR MPY832 32 BIT)
         RC
         LD          B,#3                 ; #3 FOR MPY816
                                          ; (#4 FOR MPY824)
                                          ; (#5 FOR MPY832)
         LD          [B-],#0              ; CLEAR UPPER PRODUCT
M8XXLP:  LD          A,[B]                ; FIVE INSTRUCTION
M8XXL:   RRC         A                    ;    PROGRAM LOOP TO
         X           A,[B-]               ;    RIGHT SHIFT
         IFBNE       #0                   ;    PRODUCT/MULTIPLIER
         JP          M8XXLP               ; LOOP JUMP BACK
         CLR         A                    ; CLR ACC AND TEST LOW
         IFNC                             ;    ORDER MULTIPLIER BIT
         JP          M8XXT                ; JP IF LOW ORDER BIT = 0
         RC
         LD          B,#0
         LD          A,[B]                ; MULTIPLICAND TO ACC
M8XXT:   LD          B,#3                 ; #3 FOR MPY816
                                          ; (#4 FOR MPY824)
                                          ; (#5 FOR MPY832)
         ADC         A,[B]                ; ADD MULTIPLICAND TO
                                          ;    UPPER BYTE OF PRODUCT
         DRSZ        CNTR                 ; DECREMENT AND TEST
         JP          M8XXL                ;    CNTR FOR ZERO
         RET                              ; RETURN FROM SUBROUTINE
```

MPY248—FAST 24 BY 8 MULTIPLICATION SUBROUTINE

```
        47 BYTES
        289 INSTRUCTION CYCLES AVERAGE
        333 INSTRUCTION CYCLES MAXIMUM
        MULTIPLICAND IN [2,1,0]     (ICAND)
        MULTIPLIER IN [3]           (IER)
        PRODUCT IN [6,5,4,3]        (PROD)

MPY248: LD      CNTR,#9     ; LD CNTR WITH LENGTH OF
        RC                  ;  MULTIPLIER FIELD + 1
        LD      B,#6
        LD      [B-],#0     ; CLEAR THREE
        LD      [B-],#0     ;    UPPER BYTES
        LD      [B-],#0     ;    OF PRODUCT
        JP      MP248S      ; JUMP TO START
M248LP: RRC     A           ; RIGHT SHIFT HIGH
        X       A,[B-]      ;    ORDER PRODUCT BYTE
        LD      A,[B]
        RRC     A           ; RIGHT SHIFT NEXT LOWER
        X       A,[B-]      ;    ORDER PRODUCT BYTE
        LD      A,[B]
        RRC     A           ; RIGHT SHIFT NEXT LOWER
        X       A,[B-]      ;    ORDER PRODUCT BYTE
MP248S: LD      A,[B]
        RRC     A           ; RIGHT SHIFT LOW ORDER
        X       A,[B]       ;    PRODUCT/MULTIPLIER
        IFNC                ; TEST LOW ORDER
        JP      MP248T      ;    MULTIPLIER BIT
        RC
        LD      B,#0        ; LOAD ACC WITH LOW ORDER
        LD      A,[B]       ;    MULTIPLICAND BYTE
        LD      B,#4        ; ADD LOW ORDER ICAND
        ADC     A,[B]       ;    BYTE TO NEXT TO LOW
        X       A,[B]       ;    ORDER PRODUCT BYTE
        LD      B,#1        ; LOAD ACC WTIH MIDDLE
        LD      A,[B]       ;    MULTIPLICAND BYTE
        LD      B,#5        ; ADD MIDDLE ICAND BYTE
        ADC     A,[B]       ;    TO NEXT TO HIGH ORDER
        X       A,[B]       ;    MULTIPLICAND BYTE
        LD      B,#2        ; LOAD ACC WITH HIGH ORDER
        LD      A,[B]       ;    MULTIPLICAND BYTE
        LD      B,#6        ; ADD HIGH ORDER ICAND BYTE
        ADC     A,[B]       ;    TO HIGH ORDER PROD BYTE
        DRSZ    CNTR        ; DECREMENT CNTR AND JUMP
        JP      M248LP      ;    BACK TO LOOP; CNTR
                            ;    CANNOT EQUAL ZERO
MP248T: LD      B,#6        ; HIGH ORDER PRODUCT
        LD      A,[B]       ;    BYTE TO ACC
        DRSZ    CNTR        ; DECREMENT AND TEST
        JMP     M248LP      ;    CNTR FOR ZERO
        RET                 ; RETURN FROM SUBROUTINE
```

MX1616—FAST 16 BY 16 MULTIPLICATION SUBROUTINE

```
         39 BYTES
         498 INSTRUCTION CYCLES AVERAGE
         546 INSTRUCTION CYCLES AVERAGE

         MULTIPLICAND IN [1,0]     (ICAND)
         MULTIPLIER IN [3,2]       (IER)
         PRODUCT IN [5,4,3,2]      (PROD)

MX1616:  LD          CNTR,#17      ; LD CNTR WITH LENGTH OF
         RC                        ;    MULTIPLIER FIELD + 1
         LD          B,#5
         LD          [B-],#0       ; CLEAR UPPER TWO
         LD          [B-],#0       ;    PRODUCT BYTES
         JP          MXSTRT        ; JUMP TO START
MX1616L: RRC         A             ; RIGHT SHIFT
         X           A,[B-]        ;  UPPER PRODUCT BYTE
         LD          A,[B]
         RRC         A             ; RIGHT SHIFT NEXT LOWER
         X           A,[B-]        ;    PRODUCT BYTE
MXSTRT:  LD          A,[B]
         RRC         A             ; RIGHT SHIFT PRODUCT
         X           A,[B-]        ;    UPPER MULTIPLIER BYTE
         LD          A,[B]
         RRC         A             ; RIGHT SHIFT PRODUCT
         X           A,[B]         ;    LOWER MULTIPLIER BYTE
         IFNC                      ; TEST LOW ORDER
         JP          MX1616T       ;    MULTIPLIER BIT
         RC
         LD          B,#0          ; LOAD ACC WITH LOWER
         LD          A,[B]         ;    MULTIPLICAND BYTE
         LD          B,#4          ; ADD LOWER ICAND BYTE
         ADC         A,[B]         ;    TO NEXT TO HIGH
         X           A,[B]         ;    ORDER PRODUCT BYTE
         LD          B,#1          ; LOAD ACC WITH UPPER
         LD          A,[B]         ;    MULTIPLICAND BYTE
         LD          B,#5          ; ADD UPPER ICAND BYTE TO
         ADC         A,[B]         ;    HIGH ORDER PRODUCT
         DRSZ        CNTR          ; DECREMENT CNTR AND JUMP
         JP          MX1616L       ;    BACK TO LOOP; CNTR
                                   ;    CANNOT EQUAL ZERO
MX1616T: LD          B,#5          ; HIGH ORDER PRODUCT
         LD          A,[B]         ;    BYTE TO ACC
         DRSZ        CNTR          ; DECREMENT AND TEST
         JP          MX1616L       ;  CNTR FOR ZERO
         RET                       ; RETURN FROM SUBROUTINE
```

MP1616—16 BY 16 MULTIPLICATION SUBROUTINE

```
         MINIMUM CODE
         29 BYTES
         759 INSTRUCTION CYCLES AVERAGE
         807 INSTRUCTION CYCLES MAXIMUM]
         MULTIPLICAND IN [1,0]     (ICAND)
         MULTIPLIER IN [3,2]       (IER)
         PRODUCT IN [5,4,3,2]      (PROD)

MP1616:  LD          CNTR,#17      ;  LD CNTR WITH LENGTH OF
         RC                        ;     MULTIPLIER FIELD + 1
         LD          B,#5
         LD          [B-],#0       ;  CLEAR UPPER TWO
         LD          [B-],#0       ;     PRODUCT BYTES
M1616X:  LD          A,[B]         ;  FIVE INSTRUCTION
M1616L:  RRC         A             ;     PROGRAM LOOP TO
         X           A,[B-]        ;     RIGHT SHIFT
         IFBNE       #1            ;     PRODUCT/MULTIPLIER.
         JP          M1616X        ;     LOOP JUMP BACK
         CLR         A             ;  CLEAR ACC
         IFNC                      ;  TEST LOW ORDER
         JP          M1616T        ;     MULTIPLIER BIT
         RC
         LD          B,#0          ;  LOAD ACC WITH LOWER
         LD          A,[B]         ;     MULTIPLICAND BYTE
         LD          B,#4          ;  ADD LOWER ICAND BYTE
         ADC         A,[B]         ;     TO NEXT TO LOW
         X           A,[B]         ;     ORDER PRODUCT BYTE
         LD          B,#1          ;  LOAD ACC WITH UPPER
         LD          A,[B]         ;     MULTIPLICAND BYTE
M1616T:  LD          B,#5          ;  ADD UPPER ICAND BYTE TO
         ADC         A,[B]         ;     HIGH ORDER PRODUCT
         DRSZ        CNTR          ;  DECREMENT AND TEST
         JP          M1616L        ;     CNTR EQUAL TO ZERO
         RET                       ;  RETURN FROM SUBROUTINE
```

MY1616 (OR MY1624, MY1632)—16 BY 16 (OR 24, 32) MULTIPLY SUBROUTINE

```
          MINIMUM CODE, MINIMUM RAM
          28 BYTES
          861 (OR 1473, 2213) INST. CYCLES AVERAGE
          1029 (OR 1725,1473) INST. CYCLES MAXIMUM
          EXTENDABLE ROUTINE FOR MY16XX BY CHANGING
            PARAMETERS, WITH NUMBER OF BYTES (28)
            REMAINING A CONSTANT
          MULTIPLICAND IN [1,0]                (ICAND)
          MULTIPLIER IN [3,2] FOR 16 BIT       (IER)
                     OR [4,3,2] FOR 24 BIT
                     OR [5,4,3,2] FOR 32 BIT
          PRODUCT IN [5,4,3,2] FOR 16 BIT      (PROD)
                  OR [6,5,4,3,2] FOR 24 BIT
                  OR [7,6,5,4,3,2] FOR 32 BIT

MY1616:   LD          CNTR,#17      ; LD CNTR WITH LENGTH OF
                                    ;    MULTIPLIER FIELD + 1
                                    ; #17 FOR MY1616
                                    ; (#25 FOR MY1624)
                                    ; (#33 FOR MY1632)
          LD          B,#5          ; #5 FOR MY1616
                                    ; (#6 FOR MY1624)
                                    ; (#7 FOR MY1632)
          LD          [B-],#0       ; CLEAR UPPER TWO
          LD          [B-],#0       ;    PRODUCT BYTES
          RC
MY16XS:   LD          A,[B]         ; FIVE INSTRUCTION
          RRC         A             ;    PROGRAM LOOP TO
          X           A,[B-]        ;    RIGHT SHIFT
          IFBNE       #1            ;    PRODUCT/MULTIPLIER
          JP          M16XS         ;    LOOP JUMP BACK
          IFNC                      ; TEST LOW ORDER
          JP          MY16XT        ;    MULTIPLIER BIT
          RC
          LD          B,#4          ; #4 FOR MY1616
                                    ; (#5 FOR MY1624)
                                    ; (#6 FOR MY1632)
          LD          X,#0          ; LOAD ACC WITH
MY16XL:   LD          A,[X+]        ;    MULTIPLICAND BYTES
          ADC         A,[B]         ; ADD MULTIPLICAND TO
          X           A,[B+]        ;    HI TWO PROD. BYTES
          IFBNE       #2            ; LOOP BACK FOR SECOND
          JP          MY16XL        ;    MULTIPLICAND BYTE
MY16XT:   LD          B,#5          ; #5 FOR MY1616
                                    ; (#6 FOR MY1624)
                                    ; (#7 FOR MY1632)
          DRSZ        CNTR          ; DECREMENT AND TEST
          JP          MY16XS        ;    CNTR EQUAL TO ZERO
          RET                       ; RETURN FROM INTERRUPT
;
```

MY2448—MINIMAL GENERAL MULTIPLICATION SUBROUTINE (28 BYTES)

```
    ANY NUMBER OF BYTES IN MULTIPLICAND
    AND MULTIPLIER
 FIRST EXAMPLE:     (MY2448)
    24 BY 48 MULTIPLICATION SUBROUTINE
          --28 BYTES
          --MINIMAL CODE, MINIMAL RAM
          --4713 INSTRUCTION CYCLES AVERAGE
          --5457 INSTRUCTION CYCLES MAXIMUM
    MULTIPLICAND IN [2,1,0]                    (ICAND)
    MULTIPLIER IN [8,7,6,5,4,3]                (IER)
    PRODUCT IN [11,10,9,8,7,6,5,4,3]           (PROD)

 SECOND EXAMPLE: (MY4824)
    48 BY 24 MULTIPLICATION SUBROUTINE
          --28 BYTES
          --MINIMAL CODE, NON MINIMAL RAM
          --2751 INSTRUCTION CYCLES AVERAGE
          --3483 INSTRUCTION CYCLES MAXIMUM
    MULTIPLICAND IN [5,4,3,2,1,0]              (ICAND)
    MULTIPLIER IN [8,7,6]                      (IER)
    PRODUCT IN [14,13,12,11,10,9,8,7,6]        (PROD)
```

```
MY2448:   ; (OR MY4824)
          LD          CNTR, #49   ; LD CNTR WITH LENGTH OF
                                  ;    MULTIPLIER FIELD + 1
                                  ; #49 FOR MY2448
                                  ; (#25 FOR MY4824)
          LD          B,#11       ; TOP OF PROD TO B PTR
                                  ; #11 FOR MY2448
                                  ; (#14 FOR MY4824)
CLRLUP:   LD          [B-],#0     ; CLR UNTIL TOP OF IER
          IFBNE       #8          ; #8 FOR BOTH MY2448
          JP          CLRLUP      ;    AND MY4824
          RC                      ; INITIALIZE CARRY
SHFTLP:   LD          A,[B]       ; RIGHT SHIFT PRODUCT
          ADC         A,[B]       ;    AND MULTIPLIER
          X           A,[B-]      ;    UNTIL TOP OF ICAND
          IFBNE       #2          ; #2 FOR MY2448
          JP          SHFTLP      ; (#5 FOR MY4824)
          IFNC                    ; TEST LOW ORDER
          JP          MYTEST      ;    MULTIPLIER BIT
          LD          B,#9        ; TOP OF IER + 1 TO B PTR
          LD          X,#0        ; START OF ICAND TO X PTR
          RC
ADDLUP:   LD          A,[X+]      ; ADD MULTIPLICAND TO TOP
          ADC         A,[B]       ;    OF PRODUCT ABOVE
          X           A,[B+]      ;    MULTIPLIER UNTIL TOP
          IFBNE       #12         ;    OF PRODUCT + 1
          JP          ADDLUP      ; #12 FOR MY2448
                                  ; (#15 FOR MY4824)
MYTEST:   LD          B,#11       ; TOP OF PROD TO B PTR
                                  ; #11 FOR MY2448
                                  ; (#14 FOR MY4824)
          DRSZ        CNTR        ; DECREMENT AND TEST
          JP          SHFTLP      ;    CNTR FOR ZERO
          RET                     ; RETURN FROM SUBROUTINE
```

3.0 DIVISION

The COP 800 divisions are all based on shifting the dividend left up into a test field equal in length to the number of bytes in the divisor. The divisor is resident immediately above this test field. After each shift cycle of the dividend into the test field, a trial subtraction is made of the test field minus the divisor. If the divisor is found equal to or less than the contents of the test field, then the divisor is subtracted from the test field and a 1's quotient digit is recorded by setting the low order bit of the dividend field. The dividend and test field left shift cycle is then repeated. The number of left shift cycles is equal to the number of bit positions in the dividend. The quotient from the division is formed in the dividend field, while the remainder from the division is resident in the test field.

Note that an M byte dividend divided by an N byte divisor will result in an M byte quotient and an N byte remainder.

These division algorithms will use M + 2N + 1 bytes of RAM memory space, since the test field is equal to the length of the divisor. The one extra byte is necessary for the shift counter CNTR.

In special cases where the dividend has an upper bound and the divisor has a lower bound, the upper bytes of the dividend may be used as the test field. One example is shown (DV2815), where a 28 bit dividend is divided by a 15-bit divisor. The dividend is less than 2**28 (upper nibble of high order byte is zero), while the divisor is greater than 2**12 (4096) and less than 2**15 (32768). In this case, the upper limit for the quotient is 2**28/2**12, which indicates a 16-bit quotient (2**16) and a 15-bit remainder. Consequently, the upper two bytes of the dividend may be used as the test field for the remainder, since the divisor is greater than the test field (upper two bytes of the 28-bit dividend) initially.

The minimal code (40 byte) general division subroutine is shown with the example DV3224, which divides a 32 bit dividend by a 24 bit divisor.

DIV88 — 8 by 8 Division Subroutine
— 24 Bytes
— 201 Instruction Cycles Average
— 209 Instruction Cycles Maximum
Minimum code

DV88 — Fast 8 by 8 Division Subroutine
— 28 Bytes
— 194 Instruction Cycles Average
— 202 Instruction Cycles Maximum

FDV88 — Very Fast 8 by 8 Division Subroutine
— 131 Bytes
— 146 Instruction Cycles Average
— 159 Instruction Cycles Maximum

DIV168 (or DIV248, DIV328)
— 16 (or 24, 32) by 8 Division Subroutine
— 26 Bytes
— 649 (or 1161, 1801) Instruction Cycles Average
— 681 (or 1209,1865) Instruction Cycles Maximum
— Minimum Code
— Extendable Routine for DIVXX8 by Changing Parameters, with Number of Bytes (26) Remaining a Constant

FDV168 — Fast 16 by 8 Division Subroutine
— 35 Bytes
— 481 Instruction Cycles Average
— 490 Instruction Cycles Maximum

FDV248 — Fast 24 by 8 Division Subroutine
— 38 Bytes
— 813 Instruction Cycles Average
— 826 Instruction Cycles Maximum

FDV328 — Fast 32 by 8 Division Subroutine
— 42 Bytes
— 1209 Instruction Cycles Average
— 1226 Instructions Maximum

Divide by 16 Subroutines:

DV1616 — 16 by 16 Division Subroutine
— 34 Bytes
— 979 Instruction Cycles Average
— 1067 Instruction Cycles Maximum
— Minimum Code

DV2416 (or DV3216)
— 24 (or 32) by 16 Division Subroutine
— 39 Bytes
— 1694 (or 2410) Inst. Cycles Average
— 1886 (or 2766) Inst. Cycles Maximum
— Minimum code
— Extendable Routine for DVXX16 by Changing Parameters, with Number of Bytes (39) Remaining a Constant

DX1616 — Fast 16 by 16 Division Subroutine
— 53 Bytes
— 638 Instruction Cycles Average
— 678 Instruction Cycles Maximum

DV2815 — Fast 28 by 15 Division Subroutine, Where the Dividend is Less Than 2**28 and the Divisor
is Greater than 2**12 (4096)
and Less than 2**15 (32768)
— 43 Bytes
— 640 Instruction Cycles Average
— 696 Instruction Cycles Maximum

DX3216 — Fast 32 by 16 Division Subroutine
— 70 Bytes
— 1511 Instruction Cycles Average
— 1591 Instruction Cycles Maximum

Minimal General Division Subroutine for any Number of Bytes in Dividend and Divisor
— 40 Bytes
— Minimal Code
— DV3224 Used as Example, with
3879 Instruction Cycles Average
4535 Instruction Cycles Maximum

DIV88—8 BY 8 DIVISION SUBROUTINE

```
        MINIMUM CODE
        24 BYTES
        201 INSTRUCTION CYCLES AVERAGE
        209 INSTRUCTION CYCLES MAXIMUM
        DIVIDEND     IN [0]       (DD)
        DIVISOR IN [2]            (DR)
        QUOTIENT IN [0]           (QUOT)
        REMAINDER IN [1]          (TEST FIELD)

DIV88:  LD           CNTR,#8      ;  LOAD CNTR WITH LENGTH
        LD           B,#1         ;     OF DIVIDEND FIELD
        LD           [B],#0       ;  CLEAR TEST FIELD
DIV88S  RC
        LD           B,#0
        LD           A,[B]
        ADC          A,[B]        ;  LEFT SHIFT DIVIDEND
        X            A,[B+]
        LD           A,[B]
        ADC          A,[B]        ;  LEFT SHIFT TEST FIELD
        X            A,[B]
        LD           A,[B+]       ;  TEST FIELD TO ACC
        SC                        ;  TEST SUBTRACT DIVISOR
        SUBC         A,[B]        ;     FROM TEST FIELD
        IFNC                      ;  TEST IF BORROW
        JP           DIV88B       ;     FROM SUBTRACTION
        LD           B,#1         ;  SUBTRACTION RESULT
        X            A,[B-]       ;     TO TEST FIELD
        SBIT         0,[B]        ;  SET QUOTIENT BIT
DIV88B: DRSZ         CNTR         ;  DECREMENT AND TEST
        JP           DIV88S       ;     CNTR FOR ZERO
        RET                       ;  RETURN FROM SUBROUTINE
```

DV88—FAST 8 BY 8 DIVISION SUBROUTINE

```
        28 BYTES
        194 INSTRUCTION CYCLES AVERAGE
        202 INSTRUCTION CYCLES MAXIMUM
        DIVIDEND     IN [0]      (DD)
        DIVISOR IN [2]           (DR)
        QUOTIENT IN [0]          (QUOT)
        REMAINDER IN [1]         (TEST FIELD)
DV88:   LD           CNTR,#8     ;  LOAD CNTR WITH LENGTH
        LD           B,#1        ;   OF DIVIDEND FIELD
        LD           [B-],#0     ;  CLEAR TEST FIELD
        RC
DV88S:  LD           A,[B]
        ADC          A,[B]       ;  LEFT SHIFT DIVIDEND
        X            A,[B+]
        LD           A,[B]
        ADC          A,[B]       ;  LEFT SHIFT TEST FIELD
        X            A,[B]
        LD           A,[B+]      ;  TEST FIELD TO ACC
        SC                       ;  TEST SUBTRACT DIVISOR
        SUBC         A,[B]       ;     FROM TEST FIELD
        IFNC                     ;  TEST IF BORROW
        JP           DV88B       ;     FROM SUBTRACTION
        LD           B,#1        ;  SUBTRACTION RESULT
        X            A,[B-]      ;  TO TEST FIELD
        SBIT         0,[B]       ;  SET QUOTIENT BIT
        RC
        DRSZ         CNTR        ;  DECREMENT AND TEST
        JP           DV88S       ;     CNTR FOR ZERO
        RET                      ;  RETURN FROM SUBROUTINE
DV88B:  LD           B,#0
        DRSZ         CNTR        ;  DECREMENT AND TEST
        JP           DV88S       ;     CNTR FOR ZERO
        RET                      ;  RETURN FROM SUBROUTINE
```

FDV88—VERY FAST 8 BY 8 DIVISION SUBROUTINE

```
         131 BYTES
         146 INSTRUCTION CYCLES AVERAGE
         159 INSTRUCTION CYCLES MAXIMUM

         DIVIDEND IN [0]            (DD)
         DIVISOR IN [2]             (DR)
         QUOTIENT IN [0]            (QUOT)
         REMAINDER IN [1]           (TEST FIELD)

FDV88:   LD          B,#1
         LD          [B-],#0        ;  CLEAR TEST FIELD
         RC
         LD          A,[B]
         ADC         A,[B]          ;  LEFT SHIFT DIVIDEND
         X           A,[B+]
         LD          A,[B]
         ADC         A,[B]          ;  LEFT SHIFT TEST FIELD
         X           A,[B]
         LD          A,[B+]         ;  TEST FIELD TO ACC
         SC                         ;  TEST SUBTRACT DIVISOR
         SUBC        A,[B]          ;     FROM TEST FIELD
         IFNC                       ;  TEST IF BORROW
         JP          DVBP1          ;     FROM SUBTRACTION
         LD          B,#1           ;  SUBTRACTION RESULT
         X           A,[B-]         ;     TO TEST FIELD
         SBIT        0,[B]          ;  SET QUOTIENT BIT
         RC
DVBP1:   LD          B,#0           ;  THIS 16 BYTE SECTION
         LD          A,[B]          ;  OF PROGRAM CODE
         ADC         A,[B]          ;  CONTAINS
         X           A,[B+]         ;  16 INSTRUCTIONS,
         LD          A,[B]          ;  AND REPRESENTS THE
         ADC         A,[B]          ;  PROCESSING FOR THE
         X           A,[B]          ;  GENERATION OF
         LD          A,[B+]         ;  1 QUOTIENT BIT.
         SC                         ;
         SUBC        A,[B]          ;  THE PROGRAM CODE
         IFNC                       ;  EXECUTION TIMES IS 16
         JP          DVBP2          ;  INSTRUCTION CYCLES
         LD          B,#1           ;  FOR A 0'S QUOTIENT BIT
         X           A,[B-]         ;  AND 19 INSTRUCTION
         SBIT        0,[B]          ;  CYCLES FOR A 1'S
         RC                         ;  QUOTIENT BIT.
;        ---                        ;
DVBP2:   LD          B,#0           ;  REPEAT THE ABOVE
;        ---                        ;
;DVBP3:
;        ---                        ;SECTION OF CODE FIVE
;DVBP4:
;        ---                        ;MORE TIMES FOR A
;DVBP5:
;        ---                        ;TOTAL OF SIX TIMES
;DVBP6:
;        ---                        ;
;
DVBP7:   LD          B,#0
         LD          A,[B]
         ADC         A,[B]          ;  LEFT SHIFT DIVIDEND
         X           A,[B+]
         LD          A,[B]
         ADC         A,[B]          ;  LEFT SHIFT TEST FIELD
         X           A,[B]
         LD          A,[B+]         ;  TEST FIELD TO ACC
         SC                         ;  TEST SUBTRACT DIVISOR
         SUBC        A,[B]          ;     FROM TEST FIELD
         IFNC                       ;  TEST BORROW FROM SUBC
         RET                        ;  RETURN FROM SUBROUTINE
         LD          B,#1           ;  SUBTRACTION RESULT
         X           A,[B-]         ;     TO TEST FIELD
         SBIT        0,[B]          ;  SET QUOTIENT BIT
         RET                        ;  RETURN FROM SUBROUTINE
```

DIV168—16 (OR 24, 32) BY 8 DIVISION SUBROUTINE

```
        MINIMUM CODE
        26 BYTES
        649 (or 1161,1801) INST. CYCLES AVERAGE
        681 (or 1209,1865) INST. CYCLES MAXIMUM
        EXTENDABLE ROUTINE FOR DIVXX8 BY CHANGING
        PARAMETERS, WITH NUMBER OF BYTES (26)
        REMAINING A CONSTANT
        DIVIDEND IN [1,0] FOR 16 BIT            (DD)
                 OR [2,1,0] FOR 24 BIT
                 OR [3,2,1,0] FOR 32 BIT
        DIVISOR IN [3] FOR 16 BIT               (DR)
                 OR [4] FOR 24 BIT
                 OR [5] FOR 32 BIT
        QUOTIENT IN [1,0] FOR 16 BIT            (QUOT)
                 OR [2,1,0] FOR 24 BIT
                 OR [3,2,1,0] FOR 32 BIT
        REMAINDER IN [2] FOR 16 BIT             (TEST FIELD)
                 OR [3] FOR 24 BIT
                 OR [4] FOR 32 BIT

DIV168:  LD       CNTR,#16      ; LOAD CNTR WITH LENGTH
                                ;    OF DIVIDEND FIELD
                                ; #16 FOR DIV168
                                ; (#24 FOR DIV248)
                                ; (#32 FOR DIV328)
         LD       B,#2          ; (#3 FOR DIV168)
                                ; (#3 FOR DIV248)
                                ; (#4 FOR DIV328)
         LD       [B],#0        ; CLEAR TEST FIELD
DVXX8L:  RC
         LD       B,#0
DXX8LP:  LD       A,[B]         ; LEFT SHIFT DIVIDEND
         ADC      A,[B]         ;    AND TEST FIELD
         X        A,[B+]
         IFBNE    #3            ; #3 FOR DIV168
         JP       DXX8LP        ; (#4 FOR DIV248)
                                ; (#5 FOR DIV328)
         LD       A,[B-]        ; DIVISOR TO ACCUMULATOR
         IFC                    ; TEST IF BIT SHIFTED OUT
         JP       DVXX8S        ;    OF TEST FIELD***
         IFGT     A,[B]         ; TEST DIVISOR GREATER
         JP       DVXX8T        ;    THAN REMAINDER
         SC                     ;
DVXX8S:  X        A,[B]         ; REMAINDER TO ACC
         SUBC     A,[B]         ; SUBTRACT DIVISOR
         X        A,[B]         ;    FROM REMAINDER
         LD       B,#0
         SBIT     0,[B]         ; SET QUOTIENT BIT
DVXX8T:  DRSZ     CNTR          ; DECREMENT AND TEST
         JP       DVXX8L        ;    CNTR FOR ZERO
         RET                    ; RETURN FROM SUBROUTINE

;
;
;   ***   SPECIAL CASE FOR DIVISION WHERE NUMBER OF BYTES
;         IN DIVIDEND IS GREATER THAN NUMBER OF BYTES IN DIVISOR, AND
;         DIVISOR CONTAINS A HIGH ORDER 1'S BIT. THE SHIFTED DIVIDEND
;         MAY CONTAIN A HIGH ORDER 1'S BIT IN THE TEST FIELD AND
;         YET BE SMALLER THAN THE DIVISOR SO THAT NO SUBTRACTION
;         OCCURS. iN THIS CASE A 1'S BIT WILL BE SHIFTED OUT OF
;         THE TEST FIELD AND AN OVERRIDE SUBTRACTION MUST BE PERFORMED
```

FDV168—FAST 16 BY 8 DIVISION SUBROUTINE

```
        35 BYTES
        481 INSTRUCTION CYCLES AVERAGE
        490 INSTRUCTION CYCLES MAXIMUM
        DIVIDEND IN [1,0]           (DD)
        DIVISOR IN [3]              (DR)
        QUOTIENT IN [1,0]           (QUOT)
        REMAINDER IN [2]            (TEST FIELD)

FDV168: LD      CNTR,#16    ;  LOAD CNTR WITH LENGTH
        LD      B,#3        ;     OF DIVIDEND FIELD
        LD      [B],#0      ;  CLEAR TEST FIELD
FD168S: LD      B,#0
FD168L: RC
        LD      A,[B]
        ADC     A,[B]       ;  LEFT SHIFT DIVIDEND LO
        X       A,[B+]
        LD      A,[B]
        ADC     A,[B]       ;  LEFT SHIFT DIVIDEND HI
        X       A,[B+]
        LD      A,[B]
        ADC     A,[B]       ;  LEFT SHIFT TEST FIELD
        X       A,[B]
        LD      A,[B+]      ;  TEST FIELD TO ACC
        IFC                 ;  TEST IF BIT SHIFTED OUT
        JP      FD168B      ;     OF TEST FIELD***
        SC                  ;  TEST SUBTRACT DIVISOR
        SUBC    A,[B]       ;     FROM TEST FIELD
        IFNC                ;  TEST IF BORROW
        JP      FD168T      ;     FROM SUBTRACTION
FD168R: LD      B,#2        ;  SUBTRACTION RESULT
        X       A,[B]       ;     TO TEST FIELD
        LD      B,#0
        SBIT    0,[B]       ;  SET QUOTIENT BIT
        DRSZ    CNTR        ;  DECREMENT AND TEST
        JP      FD168L      ;     CNTR FOR ZERO
        RET                 ;  RETURN FROM SUBROUTINE
FD168T: DRSZ    CNTR        ;  DECREMENT AND TEST
        JP      FD168S      ;     CNTR FOR ZERO
        RET                 ;  RETURN FROM SUBROUTINE
FD168B: SUBC    A,[B]       ;  SUBTRACT DIVISOR FROM
        JP      FD168R      ;     TEST FIELD***
```

FDV248—FAST 24 BY 8 DIVISION SUBROUTINE

```
         38 BYTES
         813 INSTRUCTION CYCLES AVERAGE
         826 INSTRUCTION CYCLES MAXIMUM
         DIVIDEND IN [2,1,0]        (DD)
         DIVISOR IN [4]             (DR)
         QUOTIENT IN [2,1,0]        (QUOT)
         REMAINDER IN [3]           (TEST FIELD)

FDV248:  LD        CNTR,#24     ; LOAD CNTR WITH LENGTH
         LD        B,#4         ;    OF DIVIDEND FIELD
         LD        [B],#0       ; CLEAR TEST FIELD
FD248S:  LD        B,#0
FD248L:  RC
         LD        A,[B]
         ADC       A,[B]        ; LEFT SHIFT DIVIDEND LO
         X         A,[B+]
         LD        A,[B]
         ADC       A,[B]        ; LEFT SHIFT DIVIDEND MID
         X         A,[B+]
         LD        A,[B]
         ADC       A,[B]        ; LEFT SHIFT DIVIDEND HI
         X         A,[B+]
         LD        A,[B]
         ADC       A,[B]        ; LEFT SHIFT TEST FIELD
         X         A,[B]
         LD        A,[B+]
         IFC                    ; TEST IF BIT SHIFTED OUT
         JP        FD248B       ;    OF TEST FIELD ***
         SC                     ; TEST SUBTRACT DIVISOR
         SUBC      A,[B]        ;    FROM TEST FIELD
         IFNC                   ; TEST IF BORROW
         JP        FD248T       ;    FROM SUBTRACTION
FD248R:  LD        B,#3         ; SUBTRACTION RESULT
         X         A,[B]        ;    TO TEST FIELD
         LD        B,#0
         SBIT      0,[B]        ; SET QUOTIENT BIT
         DRSZ      CNTR         ; DECREMENT AND TEST
         JP        FD248L       ;    CNTR FOR ZERO
         RET                    ; RETURN FROM SUBROUTINE
FD248T:  DRSZ      CNTR         ; DECREMENT AND TEST
         JP        FD248S       ;    CNTR FOR ZERO
         RET                    ; RETURN FROM SUBROUTINE
FD248B:  SUBC      A,[B]        ; SUBTRACT DIVISOR FROM
         JP        FD248R       ;    TEST FIELD ***
```

DV1616—16 (OR 24, 32) BY 16 DIVISION SUBROUTINE

```
        MINIMUM CODE
        34 BYTES
        979 (OR 1655,2459) INSTRUCTION CYCLES AVERAGE
        1067 (OR 1787,2635) INSTRUCTION CYCLES MAXIMUM
        DIVIDEND IN [1,0]          (DD)
        DIVISOR IN [5,4]           (DR)
        QUOTIENT IN [1,0]          (QUOT)
        REMAINDER IN [3,2]         (TEST FIELD)

DV1616: LD          CNTR,#16       ; LOAD CNTR WITH LENGTH
                                   ;    OF DIVIDEND FIELD
        LD          B,#3
        LD          [B-],#0        ; CLEAR
        LD          [B],#0         ;    TEST FIELD
DV616S: RC
        LD          X,#2           ; INITIALIZE X POINTER
        LD          B,#0           ; INITIALIZE B POINTER
DV616L: LD          A,[B]          ; LEFT SHIFT DIVIDEND
        ADC         A,[B]          ;    AND TEST FIELD
        X           A,[B+]
        IFBNE       #4
        JP          DV616L
        SC                         ; RESET BORROW
        LD          A,[X+]         ; TEST FIELD LO TO ACC
        SUBC        A,[B]          ; SUBT DR LO FROM REM LO
        LD          A,[X]          ; TEST FIELD HI TO ACC
        LD          B,#5
        SUBC        A,[B]          ; SUBT DR HI FROM REM HI
        IFNC                       ; TEST IF BORROW
        JP          DV616T         ;    FROM SUBTRACTION
        X           A,[X-]         ; SUBT RESULT HI TO REM HI
        LD          A,[X]          ; TEST FIELD LO TO ACC
        LD          B,#4
        SUBC        A,[B]          ; SUBT DR LO FROM REM LO
        X           A,[X]          ; RESULT LO TO REM LO
        LD          B,#0
        SBIT        0,[B]          ; SET QUOTIENT BIT
DV616T: DRSZ        CNTR           ; DECREMENT AND TEST
        JP          DV616S         ;    CNTR FOR ZERO
        RET                        ; RETURN FROM SUBROUTINE
```

DX1616—FAST 16 BY 16 DIVISION SUBROUTINE

```
        53 BYTES
        638 INSTRUCTION CYCLES AVERAGE
        678 INSTRUCTION CYCLES MAXIMUM
        DIVIDEND IN [1,0]           (DD)
        DIVISOR IN [5,4]            (DR)
        QUOTIENT IN [1,0]           (QUOT)
        REMAINDER IN [3,2]          (TEST FIELD)

DX1616: LD      CNTR,#16    ; LOAD CNTR WITH LENGTH
        LD      B,#5        ;    OF DIVIDEND FIELD
        LD      A,[B]       ; REPLACE DIVISOR WITH
        XOR     A,#0FF      ;    1'S COMPLEMENT OF
        X       A,[B-]      ;    DIVISOR TO ALLOW
        LD      A,[B]       ;    OPTIONAL ADDITION OF
        XOR     A,#0FF      ;    DIVISOR'S COMPLEMENT
        X       A,[B-]      ;    IN MAIN PROG. LOOP
        LD      [B-],#0     ; CLEAR
        LD      [B],#0      ;    TEST FIELD
DX616S: LD      B,#0
DX616L: RC
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT DIVIDEND LO
        X       A,[B+]
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT DIVIDEND HI
        X       A,[B+]
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT TEST FIELD LO
        X       A,[B+]
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT TEST FIELD HI
        X       A,[B+]
        SC
        LD      A,[B]       ; DIVISORX (DRX) LO TO ACC
        LD      B,#2        ;    (1'S COMPLEMENT)
        ADC     A,[B]       ; ADD REM LO TO DRX LO
        LD      B,#5
        LD      A,[B]       ; DIVISORX (DRX) HI TO ACC
        LD      B,#3        ;    (1'S COMPLEMENT)
        ADC     A,[B]       ; ADD REM HI TO DRX HI
        IFNC                ; TEST IF NO CARRY FROM
        JP      DX616T      ;    1'S COMPL.ADDITION
        X       A,[B+]      ; RESULT TO REM HI
        LD      A,[B]       ; DRX LO TO ACCUMULATOR
        LD      B,#2
        ADC     A,[B]       ; ADD REM LO TO DRX LO
        X       A,[B]       ; RESULT TO REM LO
        LD      B,#0
        SBIT    0,[B]       ; SET QUOTIENT BIT
        DRSZ    CNTR        ; DECREMENT AND TEST
        JP      DX616L      ;    CNTR FOR ZERO
        RET                 ; RETURN FROM SUBROUTINE
DX616T: DRSZ    CNTR        ; DECREMENT AND TEST
        JMP     DX616S      ;    CNTR FOR ZERO
        RET                 ; RETURN FROM SUBROUTINE
```

DV2815—FAST 28 BY 15 DIVISION SUBROUTINE

```
        WHERE THE DIVIDEND IS LESS THAN 2**28
        AND THE DIVISOR IS GREATER THAN 2**12 (4096) AND LESS THAN 2**15 (32768)
        43 BYTES
        640 INSTRUCTION CYCLES AVERAGE
        696 INSTRUCTION CYCLES MAXIMUM
        DIVIDEND IN [3,2,1,0]       (DD)
        DIVISOR IN [5,4]            (DR)
        QUOTIENT IN [1,0]           (QUOT)
        REMAINDER IN [3,2]          (TEST FIELD)

DV2815: LD      CNTR,#16    ; LOAD CNTR WITH LENGTH OF QUOTIENT FIELD
D2815S: LD      B,#0
D2815L: RC
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT LOWER
        X       A,[B+]      ;    BYTE OF DIVIDEND
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT NEXT HIGHER
        X       A,[B+]      ;    BYTE OF DIVIDEND
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT NEXT HIGHER
        X       A,[B+]      ;    BYTE OF DIVIDEND
        LD      A,[B]
        ADC     A,[B]       ; LEFT SHIFT UPPER
        X       A,[B-]      ;    BYTE OF DIVIDEND
        ***
        NOTE THAT WITH A 16 BIT DIVISOR (DIV 2816) SUBROUTINE, A TEST FOR A HIGH
        ORDER BIT SHIFTED OUT OF THE TEST FIELD WOULD BE NECESSARY AT THIS POINT.
        IFC
        JP      SUBTRMD     ; SUBTRACT REM MINUS DR
        THE PRESENCE OF THIS CARRY WOULD REQUIRE THAT THE DIVISOR BE SUBTRACTED
        FROM THE REMAINDER AS SHOWN WITH THE DIV168*** SUBROUTINE.
        LD      A,[B]       ; REM LOWER BYTE TO ACC
        SC                  ; TEST SUBTRACT LOWER
        LD      B,#4        ;    BYTE OF DR FROM
        SUBC    A,[B]       ;    LOWER BYTE OF REM
        LD      B,#3        ; TEST SUBTRACT UPPER
        LD      A,[B]       ;    BYTE OF DIVISOR
        LD      B,#5        ;    FROM UPPER BYTE
        SUBC    A,[B]       ;    OF REMAINDER
        IFNC                ; TEST IF BORROW
        JP      D2815T      ;    FROM SUBTRACTION
        LD      B,#3        ; UPPER BYTE OF RESULT
        X       A,[B+]      ;    TO UPPER BYTE OF REM
        LD      A,[B]       ; DR LOWER BYTE TO ACC
        LD      B,#2        ; SUBTRACT LOWER BYTE
        X       A,[B]       ;    OF DIVISOR FROM
        SUBC    A,[B]       ;    LOWER BYTE OF
        X       A,[B]       ;    REMAINDER
        LD      B,#0
        SBIT    0,[B]       ; SET QUOTIENT BIT
        DRSZ    CNTR        ; DECREMENT AND TEST
        JMP     D2815L      ;    CNTR FOR ZERO
        RET                 ; RETURN FROM SUBROUTINE
D2815T: DRSZ    CNTR        ; DECREMENT AND TEST
        JMP     D2815S      ;    CNTR FOR ZERO
        RET                 ; RETURN FROM SUBROUTINE
```

DX3216—FAST 32 BY 16 DIVISION SUBROUTINE

```
         70 BYTES
         1510 INSTRUCTION CYCLES AVERAGE
         1590 INSTRUCTION CYCLES MAXIMUM
         DIVIDEND IN [3,2,1,0]       (DD)
         DIVISOR IN [7,6]            (DR)
         QUOTIENT IN [3,2,1,0]       (QUOT)
         REMAINDER IN [5,4]          (TEST FIELD)

DX3216:  LD         CNTR,#32        ; LOAD CNTR WITH LENGTH
         LD         B,#7            ;    OF DIVIDEND FIELD
         LD         A,[B]           ; REPLACE DIVISOR WITH
         XOR        A,#0FF          ;    1'S COMPLEMENT OF
         X          A,[B-]          ;    DIVISOR TO ALLOW
         LD         A,[B]           ;    OPTIONAL ADDITION OF
         XOR        A,#0FF          ;    DIVISOR'S COMPLEMENT
         X          A,[B-]          ; IN MAIN PROG. LOOP
         LD         [B-],#0         ; CLEAR
         LD         [B],#0          ;   TEST FIELD
DX326S:  LD         B,#0
DX326L:  RC
         LD         A,[B]
         ADC        A,[B]           ; LEFT SHIFT DIVIDEND LO
         X          A,[B+]
         LD         A,[B]
         ADC        A,[B]           ; LEFT SHIFT NEXT HIGHER
         X          A,[B+]          ;    DIVIDEND BYTE
         LD         A,[B]
         ADC        A,[B+]          ; LEFT SHIFT NEXT HIGHER
         X          A,[B+]          ;    DIVIDEND BYTE
         LD         A,[B]
         ADC        A,[B]           ; LEFT SHIFT DIVIDEND HI
         X          A,[B+]
         LD         A,[B]
         ADC        A,[B]           ; LEFT SHIFT TST FIELD LO
         X          A,[B+]
         LD         A,[B]
         ADC        A,[B]           ; LEFT SHIFT TST FIELD HI
         X          A,[B+]
         IFC                        ; **TEST IF BIT SHIFTED
         JP         DX326B          ; ** OUT OF TEST FIELD
         SC
         LD         A,[B]           ; DVSORX (DRX) LO TO ACC
         LD         B,#4            ;    (1'S COMPLEMENT)
         ADC        A,[B]           ; ADD REM LO TO DRX LO
         LD         B,#7
         LD         A,[B]           ; DVSORX (DRX) HI TO ACC
         LD         B,#5            ;    (1'S COMPLEMENT)
         ADC        A,[B]           ; ADD REM HI TO DRX HI
         IFNC                       ; TEST IF NO CARRY FROM
         JP         DX326T          ;    1'S COMPL. ADDITION
         X          A,[B+]          ; RESULT TO REM NI
         LD         A,[B]           ; DRX LO TO ACCUMULATOR
         LD         B,#4
DX326R:  ADC        A,[B]           ; ADD REM LO TO DRX LO
                                    ; ** ADD REM HI TO DRX HI
         X          A,[B]           ; RESULT TO REM LO
                                    ; ** RESULT TO REM HI
LD                  B,#0
         SBIT       0,[B]           ; SET QUOTIENT BIT
         DRSZ       CNTR            ; DECREMENT AND TEST
         JMP        DX326L          ;    CNTR FOR ZERO
         RET                        ; RETURN FROM SUBROUTINE
DX326T:  DRSZ       CNTR            ; DECREMENT AND TEST
         JMP        DX326S          ;    CNTR FOR ZERO
         RET                        ; RETURN FROM SUBROUTINE
DX326B:  LD         A,[B]           ; ** REM LO TO ACC
         LD         B,#6            ; ** B PTR TO DRX LO
         ADC        A,[B]           ; ** ADD DRX LO TO REM LO
         X          A,[B]           ; ** RESULT TO REM LO
         LD         B,#7            ; **
         LD         A,[B]           ; ** DRX HI TO ACC
         LD         B,#5            ; ** B PTR TO REM HI
         JP         DX36R           ; **
   **    THESE INSTRUCTIONS UNNECESSARY IF DIVISOR
         LESS THAN 2**15 (DX3215 SUBROUTINE)
```

MINIMAL GENERAL DIVISION SUBROUTINE (40 BYTES)

```
          ANY NUMBER OF BYTES IN DIVIDEND AND DIVISOR
          DV3224 SERVES AS EXAMPLE
          32 BY 24 DIVISION SUBROUTINE
                       --40 BYTES
                       --MINIMAL CODE
                       --3879 INSTRUCTION CYCLES AVERAGE
                       --4535 INSTRUCTION CYCLES MAXIMUM

          DIVIDEND IN [3,2,1,0]        (DD)
          DIVISOR IN [9,8,7]           (DR)
          QUOTIENT IN [3,2,1,0]        (QUOT)
          REMAINDER IN [6,5,4]         (TEST FIELD)

DV3224:   LD        CNTR,#32      ; LOAD CNTR WITH LENGTH
          LD        B,#6          ;    OF DIVIDEND FIELD
CLRLUP:   LD        [B-],#0       ; CLEAR TEST FIELD
          IFBNE     #3            ; TOP OF DIVIDEND FIELD
          JP        CLRLUP
DVSHFT:   RC
          LD        B,#0
SHFTLP:   LD        A,[B]
          ADC       A,[B]         ; LEFT SHIFT DIVIDEND
          X         A,[B+]        ;    AND TEST FIELD
          IFBNE     #7            ; BOTTOM OF DR FIELD
          JP        SHFTLP
          IFC                     ; TEST IF BIT SHIFTED
          JP        DVSUBT        ; *** OUT OF TEST FIELD
          SC                      ; RESET BORROW
          LD        X,#4
TSTLUP:   LD        A,[X+]        ; TEST SUBTRACT DIVISOR
          SUBC      A,[B]         ;    FROM TEST FIELD
          LD        A,[B+]        ; INCREMENT B POINTER
          IFBNE     #10           ; TOP OF DIVISOR + 1
          JP        TSTLUP
          IFNC                    ; TEST IF BORROW
          JP        DVTEST        ;    FROM SUBTRACTION
          LD        B,#7
DVSUBT:   LD        X,#4
SUBTLP:   LD        A,[X]         ; SUBTRACT DIVISOR
          SUBC      A,[B]         ;    FROM REMAINDER
          X         A,[X+]        ;    IN TEST FIELD
          LD        A,[B+]        ; INCREMENT B POINTER
          IFBNE     #10           ; TOP OF DIVISOR + 1
          JP        SUBTLP
          LD        B,#0
          SBIT      0,[B]         ; SET QUOTIENT BIT
DVTEST:   DRSZ      CNTR          ; DECREMENT AND TEST
          JP        DVSHFT        ;    CNTR FOR ZERO
          RET                     ; RETURN FROM SUBROUTINE
```

4.0 DECIMAL (PACKED BCD)/BINARY CONVERSION

Subroutines For Two Byte Conversion:

DECBIN — Decimal (Packed BCD) to Binary
— 24 Bytes ***
— 1030 Instruction Cycles

FDTOB — Fast Decimal (Packaged BCD) to Binary
— 76 Bytes
— 92 Instruction Cycles

BINDEC — Binary to Decimal (Packed BCD)
— 25 Bytes ***
— 856 Instruction Cycles

FBTOD — Fast Binary to Decimal (Packed BCD)
— 59 Bytes
— 334 Instruction Cycles

VFBTOD — Very Fast Binary to Decimal (Packed BCD)
— 189 Bytes
— 144 Instruction Cycles Average
— 208 Instruction Cycles Maximum

***These subroutines extendable to multiple byte conversion by simply changing parameters within subroutine as shown, with number of bytes in subroutine remaining constant.

DECBIN—Decimal (Packed BCD) to Binary

This 24 byte subroutine represents very minimal code for translating a packed BCD decimal number of any length to binary.

ALGORITHM:

The binary result is resident just below the packed BCD decimal number. During each cycle of the algorithm, the decimal operand and the binary result are shifted right one bit position, with the low order bit of the decimal operand shifting down into the high order bit position of the binary field. The residual decimal operand is then tested for a high order bit in each of its nibbles. A three is subtracted from each nibble in the BCD operand space that is found to contain a high order bit equal to one. (This process effectively right shifts the BCD operand one bit position, and then corrects the result to BCD format.) The entire cycle is then repeated, with the total number of cycles being equal to the number of bit positions in the decimal field.

16 Bit: Binary IN [1,0]
Packed BCD in [3, 2]

24 Bit: Binary in [2, 1, 0]
Packed BCD in [5, 4, 3]

32 Bit: Binary in [3, 2, 1, 0]
Packed BCD in [7, 6, 5, 4]

24 Bytes
1030 Instruction Cycles (16 Bit)

```
DECBIN:    LD          CNTR,#16          ; LOAD CNTR WITH NUMBER
                                         ;    OF BIT POSITIONS
                                         ;    IN BCD FIELD
                                         ; #16 FOR 16 BIT (2 BYTE)
                                         ; #'S 24/32 FOR 24/32 BIT
DB1:       LD          B,#3              ; #'S 5/7 FOR 24/32 BIT
           RC
DB2:       LD          A,[B]             ; PROGRAM LOOP TO
           RRC         A                 ;    RIGHT SHIFT
           X           A,[B-]            ;    DECIMAL (BCD) AND
           IFBNE       #0F               ;    BINARY FIELDS.
           JP          DB2               ;    LOOP JUMP BACK
           LD          B,#3              ; #'S 5/7 FOR 24/32 BIT
           SC                            ; SET CARRY FOR SUBTRACT
DB3:       LD          A,[B]             ; TEST HIGH ORDER BITS
           IFBIT       7,[B]             ;    OF BCD NIBBLES, AND
           SUBC        A,#030            ;    SUBTRACT A THREE
           IFBIT       3,[B]             ;    FROM EACH NIBBLE IF
           SUBC        A,#3              ;    HIGH ORDER BIT OF
           X           A,[B-]            ;    NIBBLE IS A ONE
           IFBNE       #1                ; #'S 2/3 FOR 24/32 BIT
           JP          DB3               ; LOOP BACK FOR MORE BCD BYTES
           DRSZ        CNTR              ; DECREMENT AND TEST IF
           JP          DB1               ; CNTR EQUAL TO ZERO
           RET                           ; RETURN FROM SUBROUTINE
```

FDTOB—FAST DECIMAL (PACKED BCD) TO BINARY

BCD Format: Four Nibbles − W, X, Y, Z, with W = Hi Order Nibble

*** [1] = 16W + X

*** [0] = 16Y + Z

Algorithm: Binary Result is equal to 100(10W + X) + (10Y + Z)

BCD IN [1, 0]***

Temp in [2]

Binary in [4, 3]

76 Bytes

92 Instruction Cycles

```
FDTOB:  RC
        LD      B,#1
        LD      A,[B+]      ;  16W + X
        AND     A,#0F0      ;  EXTRACT 16W
        RRC     A           ;  8W
        X       A,[B]       ;  8W TO TEMP
        RRC     A           ;  4W
        RRC     A           ;  2W
        ADD     A,[B]       ;  2W + 8W = 10W
        X       A,[B−]      ;  10W TO TEMP
        LD      A,[B+]      ;  16W + X
        AND     A,#0F       ;  EXTRACT X
        ADC     A,[B]       ;  10W + X
        X       A,[B]       ;  10W + X TO TEMP
        LD      A,[B]
        ADC     A,[B]       ;  2.(10W + X)
        X       A,[B]       ;  2.(10W + X) TO TEMP
        ADC     A,[B]       ;  3.(10W + X)
        LD      B,#3        ;     = 16P + Q
        X       A,[B+]      ;  16P + Q TO [3]
        CLR     A
        IFC
        LD      A,#010      ;  16C TO A (C = CARRY)
        X       A,[B−]      ;  16C TO [4]
        LD      A,[B]       ;  16P + Q
        SWAP    A           ;  16Q + P
        X       A,[B]       ;  16Q + P TO [3]
        LD      A,[B+]      ;  16Q + P
        AND     A,#0F       ;  EXTRACT P
        ADD     A,[B]       ;  16C + P
        X       A,[B−]      ;  16C + P TO [4]**
        LD      A,[B]       ;  16Q + P
        AND     A,#0F0      ;  EXTRACT 16Q
        X       A,[B−]      ;  16Q TO [3]**
        LD      A,[B+]      ;  2.(10W + X)
        ADC     A,[B]       ;  2.(10W + X) + 16Q
```

```
        X           A,[B+]          ;  2 BYTE 2.(10W + X)
        CLR         A,[B−]          ;     ADD: + 48.**(10W + X)
        ADC         A,[B]           ;  16C + P + NU C
        X           A,[B−]          ;  50.(10W + X)
        LD          A,[B]
        ADC         A,[B]           ;  DOUBLE
        X           A,[B+]          ;     50.(10W + X)
        LD          A,[B]           ;     TO FORM
        ADC         A,[B]           ;     100.(10W + X)
        X           A,[B]           ;     IN [3,4]
        LD          B,#0
        LD          A,[B]           ;  16Y + Z
        AND         A,#0F0          ;  EXTRACT 16Y
        LD          B,#2
        RRC         A               ;  8Y
        X           A,[B]           ;  8Y TO TEMP
        LD          A,[B]
        RRC         A               ;  4Y
        RRC         A               ;  2Y
        ADC         A,[B]           ;  2Y + 8Y = 10Y
        X           A,[B]           ;  10Y TO TEMP
        LD          B,#0
        LD          A,[B]           ;  16Y + Z
        AND         A,#0F           ;  EXTRACT Z
        LD          B,#2
        ADD         A,[B]           ;  10Y + Z
        LD          B, #3
        ADC         A,[B]           ;  TWO BYTE ADD
        X           A,[B+]          ;     100.(10W + X)
        CLR         A               ;     + (10Y + Z)
        ADC         A,[B]           ;     WITH BINARY
        X           A,[B]           ;     RESULT TO [3,4]
        RET
```

BINDEC—Binary to Decimal (Packed BCD)

This 25 byte subroutine represents very minimal code for translating a binary number of any length to packed BCD decimal.

ALGORITHM:

The packed BCD decimal result is resident just above the binary number. A sufficient number of bytes must be allowed for the BCD result. During each cycle of the algorithm the binary number is shifted left one bit position. The packed BCD decimal result is also shifted left one bit position, with the high order bit of the binary field being shifted up into the low order bit position of the BCD field. The shifted result in the BCD field is decimal corrected by using the DCOR instruction. Note that for addition an ''ADD A, #066'' instruction must be used in conjunction with the DCOR (Decimal Correct) instruction. The entire cycle is then repeated, with the total number of cycles being equal to the number of bit positions in the binary field.

16 Bit:	Binary in [1, 0]
	Packed BCD in [4, 3, 2]
24 Bit:	Binary in [2, 1, 0]
	Packed BCD in [6, 5, 4, 3]
32 Bit:	Binary in [3, 2, 1, 0]
	Packed BCD in [8, 7, 6, 5, 4]

25 Bytes
856 Instructions Cycles (16 Bit)

```
BINDEC:   LD       CNTR,#16       ;  LOAD CNTR WITH NUMBER OF BIT POSITIONS
                                  ;    IN BINARY FIELD
                                  ; #16 FOR 16 BIT (2 BYTE)
                                  ; #'S 24/32 FOR 24/32 BIT
          RC
          LD       B,#2           ; #'S 3/4 FOR 24/32 BIT
BD1:      LD       [B+],#0        ; CLEAR BCD FIELD
          IFBNE    #5             ; #'S 7/9 FOR 24/32 BIT
          JP       BD1            ; JUMP BACK FOR CLR LOOP
BD2:      LD       B,#0
BD3:      LD       A,[B]          ; PROGRAM LOOP TO
          ADC      A,[B]          ;    LEFT SHIFT
          X        A,[B+]         ;    BINARY FIELD
          IFBNE    #2             ; #'S 3/4 FOR 24/32 BIT
          JP       BD3            ; JUMP BACK FOR SHIFT LOOP1
BD4:      LD       A,[B]          ; PROGRAM LOOP TO
          ADD      A,#066         ;    LEFT SHIFT AND
          ADC      A,[B]          ;    DECIMAL CORRECT
          DCOR     A              ;    RESULT OF SHIFT
          X        A,[B+]         ;    IN BCD FIELD
          IFBNE    #5             ; #'S 7/9 FOR 24/32 BIT
          JP       BD4            ; JUMP BACK FOR SHIFT LOOP2
          DRSZ     CNTR           ; DECREMENT AND TEST IF
          JP       BD2            ;  CNTR EQUAL TO ZERO
          RET                     ; RETURN FROM SUBROUTINE
```

FBTOD—FAST BINARY TO DECIMAL (PACKED BCD)

Algorithm: This algorithm is based on the BINDEC algorithm, except that it is optimized for speed of execution.

Binary in [1, 0]
Packed BCD in [4, 3, 2]

59 Bytes
334 Instruction Cycles

```
FBTOD:   RC
         LD        B,#1
         LD        A,[B]
         SWAP      A                ;  REVERSE NIBBLES IN
         X         A,[B]            ;     UPPER BINARY BYTE
         LD        A,[B+]           ;  EXTRACT ORIGINAL UPPER
         AND       A,#0F            ;     NIBBLE OF HI BYTE
         IFGT      A,#9             ;  IF NIBBLE GREATER THAN
         ADD       A,#06            ;     NINE, THEN ADD SIX TO CORRECT BCD NIBBLE
         X         A,[B+]           ;  NIBBLE TO LOWER BCD BYTE
         LD        [B+],#0          ;  CLEAR UPPER BCD BYTES
         LD        [B],#0           ;  INITIALIZE CNTR TO COVER
         LD        CNTR,#4          ;     REMAINING HI NIBBLE (ORIGINALLY LO NIBBLE)
                                    ;  IN UPPER BINARY BYTE
FBD1:    LD        B,#1             ;  PROGRAM LOOP TO
         LD        A,[B]            ;     LEFT SHIFT A BIT
         ADC       A,[B]            ;     OUT OF UPPER BINARY
         X         A,[B+]           ;     BYTE INTO LOW ORDER
         LD        A,[B]            ;     BIT POSITION OF BCD
         ADD       A,#066           ;     FIELD, AS LOWER TWO
         ADC       A,[B]            ;     BYTES OF BCD FIELD
         DCOR      A                ;     ARE LEFT SHIFTED WITH
         X         A,[B+]           ;     THE LOWER BYTE BEING
         LD        A,[B]            ;     DECIMAL CORRECTED
         ADC       A,[B]            ;  MIDDLE BYTE OF BCD FIELD
         X         A,[B]            ;     NEED NOT BE DECIMAL CORRECTED, SINCE
                                    ;     MAX VALUE IS 2 (256)
         DRSZ      CNTR             ;  DECREMENT AND TEST IF
         JP        FBD1             ;     CNTR EQUAL TO ZERO
         LD        CNTR,#8          ;  INITIALIZE CNTR TO COVER
FBD2:    LD        B,#0             ;     LOWER BINARY BYTE
         LD        A,[B]            ;  PROGRAM LOOP TO
         ADC       A,[B]            ;     LEFT SHIFT A BIT
         X         A,[B]            ;     OUT OF LOWER BINARY
         LD        B,#2             ;     BYTE INTO LOW ORDER
         LD        A,[B]            ;     BIT POSITION OF BCD
         ADD       A,#066           ;     FIELD, AS BCD FIELD
         ADC       A,[B]            ;     IS LEFT SHIFTED WITH
         DCOR      A                ;     THE LOWER TWO BYTES
         X         A,[B+]           ;     OF THE FIELD BEING
         LD        A,[B]            ;     DECIMAL CORRECTED
         ADD       A,#066           ;  ADD (NOT ADC) HEX 66
         ADC       A,[B]            ;     TO SET UP "ADD" DCOR
         DCOR      A                ;  DECIMAL CORRECT MIDDLE
         X         A,[B+]           ;     BYTE OF BCD FIELD
         LD        A,[B]            ;  UPPER BYTE OF BCD FIELD
         ADC       A,[B]            ;     NEED NOT BE DECIMAL
         X         A,[B]            ;     CORRECTED, SINCE MAX
                                    ;     VALUE IS 6 (65535)
         DRSZ      CNTR             ;  DECREMENT AND TEST IF
         JP        FBD2             ;     CNTR EQUAL TO ZERO
         RET                        ;  RETURN FROM SUBROUTINE
```

VFBTOD—VERY FAST BINARY TO DECIMAL (PACKED BCD)

Algorithm: Decimal (Packed BCD) result is equal to summation in BCD of powers of two corresponding to 1's bits present in binary number.

Note that binary field (2 bytes) is initially one's complemented by program, in order to facilitate bypass branching when a tested bit in the binary field is found equal to zero.

Binary in [1, 0]
BCD in [4, 3, 2]

189 Bytes
144 Instruction Cycles Average
208 Instruction Cycles Maximum

```
VFBTOD:   RC
          LD        B,#0
          LD        A,[B]
          AND       A,#0F          ;  EXTRACT LO NIBBLE
          IFGT      A,#9           ;  TEST NIBBLE 9
          ADD       A,#6           ;  ADD 6 FOR CORRECTION
          LD        B,#2
          X         A,[B+]         ;  STORE IN LO BCD NIBBLE
          LD        [B+],#0        ;  CLEAR UPPER
          LD        [B],#0         ;     BCD NIBBLES
          LD        B,#1
          LD        A,[B]
          XOR       A,#0FF         ;  COMPLEMENT HI BYTE
          X         A,[B-]         ;     FOR REVERSE TESTING
          LD        A,[B]          ;     OF BINARY NUMBER
          XOR       A,#0FF         ;  COMPLEMENT LO BYTE
          X         A,[B]          ;     FOR REVERSE TESTING
          IFBIT     4,[B]          ;  TEST BINARY BIT 4
          JP        VFB1           ;     TO CONDITIONALLY
          LD        B,#2           ;     ADD BCD 16
          LD        A,#07C         ;  16 + 66
          ADC       A,[B]          ;  ADD BCD 16
          DCOR      A
          X         A,[B]
          LD        B,#0
VFB1:     IFBIT     5,[B]          ;  TEST BINARY BIT 5
          JP        VFB2           ;     TO CONDITIONALLY
          LD        B,#2           ;     ADD BCD 32
          LD        A,#098         ;  32 + 66
          ADC       A,[B]          ;  ADD BCD 32
          DCOR      A
          X         A,[B]
          LD        B,#0
VFB2:     IFBIT     6,[B]          ;  TEST BINARY BIT 6
          JP        VFB3           ;     TO CONDITIONALLY
          LD        B,#2           ;     ADD BCD 64
          LD        A,#0CA         ;  64 + 66
          ADC       A,[B]          ;  ADD BCD 64
          DCOR      A
          X         A,[B+]
          CLR       A
          ADC       A,[B]          ;  ADD CARRY
          X         A,[B]
          LD        B,#0
```

```
VFB3:   IFBIT   7,[B]       ; TEST BINARY BIT 7
        JP      VFB4        ;    TO CONDITIONALLY
        LD      B,#2        ;    ADD BCD 128
        LD      A,#08E      ; 28 + 66
        ADC     A,[B]       ; ADD BCD 28
        DCOR    A
        X       A,[B+]
        LD      A,#1
        ADC     A,[B]       ; ADD BCD 1
        X       A,[B]
VFB4:   LD      B,#1        ; HI BINARY BYTE
        IFBIT   0,[B]       ; TEST BINARY BIT 8
        JP      VFB5        ;    TO CONDITIONALLY
        LD      B,#2        ;    ADD BCD 256
        LD      A,#0BC      ; 56 + 66
        ADC     A,[B]       ; ADD BCD 56
        DCOR    A
        X       A,[B+]
        LD      A,#2
        ADC     A,[B]       ; ADD BCD 2
        X       A,[B]
        LD      B,#1
VFB5:   IFBIT   1,[B]       ; TEST BINARY BIT 9
        JP      VFB6        ;    TO CONDITIONALLY
        LD      B,#2        ;    ADD BCD 512
        LD      A,#078      ; 12 + 66
        ADC     A,[B]       ; ADD BCD 12
        DCOR    A
        X       A,[B+]
        LD      A,#06B      ; 5 + 66
        ADC     A,[B]       ; ADD BCD 5
        DCOR    A
        X       A,[B]
        LD      B,#1
VFB6:   IFBIT   2,[B]       ; TEST BINARY BIT 10
        JP      VFB7        ;    TO CONDITIONALLY
        LD      B,#2        ;    ADD BCD 1024
        LD      A,#08A      ; 24 + 66
        ADC     A,[B]       ; ADD BCD 24
        DCOR    A
        X       A,[B+]
        LD      A,#076      ; 10 + 66
        ADC     A,[B]       ; ADD BCD 10
        DCOR    A
        X       A,[B]
        LD      B,#1
VFB7:   IFBIT   3,[B]       ; TEST BINARY BIT 11
        JP      VFB8        ;    TO CONDITIONALLY
        LD      B,#2        ;    ADD BCD 2048
        LD      A,#0AE      ; 48 + 66
        ADC     A,[B]       ; ADD BCD 48
        DCOR    A
        X       A,[B+]
        LD      A,#086      ; 20 + 66
        ADC     A,[B]       ; ADD BCD 20
        DCOR    A
        X       A,[B]
        LD      B,#1
```

```
VFB8:    IFBIT   4,[B]        ; TEST BINARY BIT 12
         JP      VFB9         ;    TO CONDITIONALLY
         LD      B,#2         ;    ADD BCD 4096
         LD      A,#0FC       ; 96 + 66
         ADC     A,[B]        ; ADD BCD 96
         DCOR    A
         X       A,[B+]
         LD      A,#0A6       ; 40 + 66
         ADC     A,[B]        ; ADD BCD 40
         DCOR    A
         X       A,[B]
         LD      B,#1
VFB9:    IFBIT   5,[B]        ; TEST BINARY BIT 13
         JP      VFB10        ;    TO CONDITIONALLY
         LD      B,#2         ;    ADD BCD 8192
         LD      A,#0F8       ; 92 + 66
         ADC     A,[B]        ; ADD BCD 92
         DCOR    A
         X       A,[B+]
         LD      A,#0E7       ; 81 + 66
         ADC     A,[B]        ; ADD BCD 81
         DCOR    A
         X       A,[B]
         CLR     A
         ADC     A,[B]        ; ADD CARRY
         X       A,[B]
         LD      B,#1
VFB10:   IFBIT   6,[B]        ; TEST BINARY BIT 14
         JP      VFB11        ;    TO CONDITIONALLY
         LD      B,#2         ;    ADD BCD 16384
         LD      A,#0EA       ; 84 + 66
         ADC     A,[B]        ; ADD BCD 84
         DCOR    A
         X       A,[B+]
         LD      A,#0C9       ; 63 + 66
         ADC     A,[B]        ; ADD BCD 63
         DCOR    A
         X       A,[B+]
         LD      A,#1
         ADC     A,[B]        ; ADD BCD 1
         X       A,[B]
         LD      B,#1
VFB11:   IFBIT   7,[B]        ; TEST BINARY BIT 15
         RET                  ;    TO CONDITIONALLY
         LD      B,#2         ;    ADD BCD 32768
         LD      A,#0CE       ; 68 + 66
         ADC     A,[B]        ; ADD BCD 68
         DCOR    A
         X       A,[B+]
         LD      A,#08D       ; 27 + 66
         ADC     A,[B]        ; ADD BCD 27
         DCOR    A
         X       A,[B+]
         LD      A,#3
         ADC     A,[B]        ; ADD BCD 3
         X       A,[B]
         RET
```

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998