```
DDF1 CAFEDD               JZ     :DDFE       And abort

                  * Entry if from edit buffer:

DDF4 E5           EFC10   PUSH   H
DDF5 2A3201               LHLD   :0132       Get EFEPT
DDF8 79                   MOV    A,C
DDF9 CD30DE               CALL   :DE30       Add curr.line pos to EFEPT
DDFC 7E                   MOV    A,M         Get character
DDFD E1                   POP    H
DDFE C9           EFC20   RET

                  * If from screen:

DDFF EF           EFC30   RST    5           Get character from line
DE00 15                   DATA   :15
DE01 C9                   RET
                  *
                  *
                  *  ==================================
                  *** SINGLE AND DOUBLE BYTE UTILITIES ***
                  *  ==================================
                  *
                  *
                  *********************************
                  * CHECK IF UPPER CASE CHARACTER *
                  *********************************
                  *
                  * Entry: A: Character to be checked.
                  * Exit:  CY=0: Not upper case.
                  *        CY=1: Upper case.
                  *        ABCDEHL preserved, F corrupted.
                  *
DE02 FE41         ALPHA   CPI    :41         Lowest upper case char
DE04 3F                   CMC
DE05 D0                   RNC
DE06 FE5B                 CPI    :5B         First lower case char
DE08 C9                   RET
                  *
                  *********************************************
                  * CHECK IF CHARACTER IS NUMBER OR UPPER CASE *
                  *********************************************
                  *
                  * Entry: A: Character to be checked.
                  * Exit:  CY=0: Not number, not upper case.
                  *        CY=1: Number or upper case.
                  *        ABCDEHL preserved. F corrupted.
                  *
DE09 CD02DE       ALNUM   CALL   :DE02       Check if upper case
DE0C D8                   RC
DE0D FE30         NUMER   CPI    :30         Lowest number
DE0F 3F                   CMC
DE10 D0                   RNC
DE11 FE3A                 CPI    :3A         No number anymore
DE13 C9                   RET
                  *
                  **********************
                  * COMPARE HL AND DE *
                  **********************
                  *
                  * Compares HL with DE (HL-DE).
                  *
```

```
                      * Exit: DE=HL: Z=1, CY=0.
                      *       DE<HL: Z=0, CY=0.
                      *       DE>HL: Z=0, CY=1.
                      *       BCDEHL preserved, AF corrupted.
                      *
DE14 7C               COMP    MOV   A,H
DE15 BA                       CMP   D
DE16 C0                       RNZ
DE17 7D                       MOV   A,L
DE18 BB                       CMP   E
DE19 C9                       RET
                      *
                      ***************************
                      * CALCULATE LENGTH OF BLOCK *
                      ***************************
                      *
                      * Sets HL=HL-DE.
                      *
                      * Entry: Startaddress in DE, 1st address after
                      *        block in HL.
                      * Exit:  Length in HL, startaddress in DE.
                      *        If DE>HL, length in 2-complement.
                      *        ABCDE preserved, F as in COMP.
                      *
DE1A C5               SUBDE   PUSH  B
DE1B F5                       PUSH  PSW
DE1C 7D                       MOV   A,L
DE1D 93                       SUB   E         Calc. difference lowest byte
DE1E 6F                       MOV   L,A
DE1F 7C                       MOV   A,H
DE20 9A                       SBB   D         Calc. diff. highest byte
DE21 67                       MOV   H,A
DE22 C1                       POP   B
DE23 78                       MOV   A,B
DE24 C1                       POP   B
DE25 C9                       RET
                      *
                      *****************************
                      * DOUBLE BYTE TWO COMPLEMENT *
                      *****************************
                      *
                      * Sets HL=-HL.
                      *
                      * Entry: Double byte to be converted in HL.
                      * Exit:  Two complement in HL. ABCDEF preserved.
                      *
DE26 F5               CMPHL   PUSH  PSW
DE27 7C                       MOV   A,H
DE28 2F                       CMA             Complement H
DE29 67                       MOV   H,A
DE2A 7D                       MOV   A,L
DE2B 2F                       CMA             Complement L
DE2C 6F                       MOV   L,A
DE2D 23                       INX   H         Add 1
DE2E F1                       POP   PSW
DE2F C9                       RET
                      *
                      *************************
                      * ADD OFF-SET TO ADDRESS *
                      *************************
                      *
                      * Adds a given offset to a base address (HL=HL+A).
```

```
                    *
                    * Entry: Base in HL, offset in A.
                    * Exit:  HL=HL+A. ABCDE preserved, F corrupted.
                    *
DE30 F5             DADA     PUSH   PSW
DE31 85                      ADD    L
DE32 6F                      MOV    L,A          L=L+A
DE33 7C                      MOV    A,H
DE34 CE00                    ACI    :00          Add carry if overflow
DE36 67                      MOV    H,A
DE37 F1                      POP    PSW
DE38 C9                      RET
                    *
                    **********************************
                    * CALCULATE ADDRESS AFTER STRING *
                    **********************************
                    *
                    * Sets HL=HL+M+1.
                    *
                    * Entry: HL points to 1st byte of string (length
                    *        byte).
                    * Exit:  HL points to first byte after string.
                    *        AFBCDE preserved.
                    *
DE39 F5             DADM     PUSH   PSW
DE3A 7E                      MOV    A,M          Get length of string
DE3B 23                      INX    H            HL: addr. 1st char. byte
DE3C CD30DE                  CALL   :DE30        Calc addr after string
DE3F F1                      POP    PSW
DE40 C9                      RET
                    *
                    *****************
                    * DELAY ROUTINE *
                    *****************
                    *
                    * Runs a fixed delay loop of 665 msec. If
                    * interrupts are enabled, the delay will be
                    * approx. 750 msec.
                    * HL is loaded with FFFF, and then decremented.
                    *
                    * Exit: ABCDEHL preserved; F corrupted.
                    *
DE41 E5             DELAY    PUSH   H
DE42 D5                      PUSH   D
DE43 21FFFF                  LXI    H,:FFFF      Init. delay value
DE46 54                      MOV    D,H
DE47 5D                      MOV    E,L
DE48 19             DLY10    DAD    D
DE49 DA48DE                  JC     :DE48        Repeat if not ready
DE4C D1                      POP    D
DE4D E1                      POP    H
DE4E C9                   →  RET
                    *
                    ************************
                    * DATA BLOCK TRANSFER *
                    ************************
                    *
                    * Moves a block of data starting at (DE) and
                    * ending at (HL)-1 to (BC).
                    *
                    * Entry: DE: Startaddr. source bank.
                    *        BC: Startaddr. destination bank.
```

```
                *       HL: Points after end source bank.
                * Exit: AF preserved, BCDEHL corrupted.
                *
DE4F F5         MOVE    PUSH  PSW
DE50 E5                 PUSH  H
DE51 CD1ADE             CALL  :DE1A     Calc. length source bank
DE54 79                 MOV   A,C
DE55 93                 SUB   E
DE56 78                 MOV   A,B
DE57 9A                 SBB   D
DE58 DA6CDE             JC    :DE6C     If destination addr. is
                                        lower than source addr.

                * Destination address > source address:

DE5B 54                 MOV   D,H
DE5C 5D                 MOV   E,L       Save length in DE
DE5D 09                 DAD   B         Highest dest.addr. in HL
DE5E C1                 POP   B
DE5F 7A         MOV10   MOV   A,D       Check if ready
DE60 B3                 ORA   E
DE61 CA7ADE             JZ    :DE7A     Then abort
DE64 1B                 DCX   D
DE65 2B                 DCX   H
DE66 0B                 DCX   B
DE67 0A                 LDAX  B         Get byte to be transferred
DE68 77                 MOV   M,A       Transfer it
DE69 C35FDE             JMP   :DE5F     Next one

                * Destination address < source address:

DE6C 7C         MOV20   MOV   A,H
DE6D B5                 ORA   L
DE6E CA79DE             JZ    :DE79     Abort if ready
DE71 2B                 DCX   H
DE72 1A                 LDAX  D         Get byte to be transferred
DE73 02                 STAX  B         Transfer it
DE74 13                 INX   D
DE75 03                 INX   B
DE76 C36CDE             JMP   :DE6C     Next byte

                * If ready:

DE79 E1         MOV30   POP   H
DE7A F1         MOV40   POP   PSW
DE7B C9                 RET
                *
                **********************************
                * FILL BANK WITH IDENTICAL DATA *
                **********************************
                *
                * Fills an area of memory with a constant.
                *
                * Entry: DE: Startaddr. of bank.
                *        HL: Points after bank.
                *        A:  Data to be loaded into bank.
                * Exit:  DE: Points after bank.
                *        BCHL preserved, AF corrupted.
                *
DE7C C5         FILL    PUSH  B
DE7D 47                 MOV   B,A       Save data in B
DE7E CD14DE     FIL10   CALL  :DE14     Check if bank full
```

```
DEB1 CA8DDE                JZ     :DE8D      Abort if ready
DEB4 DA8DDE                JC     :DE8D      Abort if DE>HL
DEB7 78                    MOV    A,B        Get data
DEB8 12                    STAX   D          and store it
DEB9 13                    INX    D
DEBA C37EDE                JMP    :DE7E      Next addr.
DEBD C1           FIL20    POP    B
DEBE C9                    RET
                  *
                  ********************
                  * MULTIPLY HL BY A *
                  ********************
                  *
                  * Multiplies a 16-bit value by a 8-bit value.
                  *
                  * Entry: HL: 16-bit value.
                  *         A:  8-bit value.
                  * Exit:   CY=0: Result in HL.
                  *         CY=1: Overflow.
                  *         ABCDE preserved.
                  *
DE8F 37           HLMUL    STC
DE90 F5                    PUSH   PSW
DE91 D5                    PUSH   D
DE92 EB                    XCHG
DE93 210000                LXI    H,:0000    Init. result
DE96 B7           HLM10    ORA    A
DE97 1F                    RAR               Next bit of multiplier
DE98 D29FDE                JNC    :DE9F      Jump if bit is 0
DE9B 19                    DAD    D          Add 1* HL if bit is 1
DE9C DAADDE                JC     :DEAD      Abort if overflow
DE9F B7           HLM20    ORA    A
DEA0 CAB1DE                JZ     :DEB1      Abort if ready
DEA3 EB                    XCHG
DEA4 29                    DAD    H          Multiply *2
DEA5 EB                    XCHG
DEA6 D296DE                JNC    :DE96      Again if no overflow
DEA9 00                    NOP
DEAA 00                    NOP
DEAB 00                    NOP
DEAC 00                    NOP
DEAD 00           HLM90    NOP
DEAE D1                    POP    D
DEAF F1                    POP    PSW        Error exit if overflow
DEB0 C9                    RET
DEB1 D1           HLM99    POP    D
DEB2 F1                    POP    PSW
DEB3 3F                    CMC               No error exit
DEB4 C9                    RET
                  *
                  *
                  *
DEB5                       END

***************************
* S Y M B O L   T A B L E *
***************************

ALNUM   DE09    ALPHA   DE02    BRS10   DDC5    BRSER   DDC0
CINC    DDB4    CMPHL   DE26    COLO    DD55    COMP    DE14
COUTC   DD6A    CRLF    DD5E    DADA    DE30    DADM    DE39
DELAY   DE41    DLY10   DE48    EFC10   DDF4    EFC20   DDFE
```

EFC30	DDFF	EFETCH	DDE0	EXIT1	DD45	FIL10	DE7E
FIL20	DE8D	FILL	DE7C	HLM10	DE96	HLM20	DE9F
HLM90	DEAD	HLM99	DEB1	HLMUL	DE8F	IGNB	DDD2
IGNBR	DDD1	INPLO	DD1A	INPLN	DD1F	INSER	DDB4
IPL10	DD22	IPL20	DD2A	IPL30	DD49	MOV10	DE5F
MOV20	DE6C	MOV30	DE79	MOV40	DE7A	MOVE	DE4F
NUMER	DE0D	OTBIN	DD75	OTC10	DD70	OTC20	DD8E
OTC99	DD8C	OTS10	DD95	OTS20	DD9D	OUTC	DD60
OUTSE	DD94	SCCHR	DD60	SUBDE	DE1A		