```
                            ORG   :E602
                   *
                   *
                   *
                   ********************
                   * RUN SCREEN ERROR *
                   ********************
                   *
                   * Entry: A: Error code: 01: Off screen.
                   *                       02: Colour not available.
E602 FE01          SCRER    CPI   :01         Code is 1?
E604 3E11                   MVI   A,:11
E606 CAF5D9                 JZ    :D9F5       Then run error 'OFF SCREEN'
E609 3E10                   MVI   A,:10       Else: run error
E60B C3F5D9                 JMP   :D9F5       'COLOUR NOT AVAILABLE'
                   *
                   ************************
                   * RUN basiccmd COLORT *
                   ************************
                   *
E60E CD1CE6        RCOLT    CALL  :E61C       Get colours in scratch area
E611 EF                     RST   5           Set text colours
E612 06                     DATA  :06
E613 B7                     ORA   A           No special action
E614 C9                     RET
                   *
                   ************************
                   * RUN basiccmd COLORG *
                   ************************
                   *
E615 CD1CE6        RCOLG    CALL  :E61C       Get colours in scratch area
E618 EF                     RST   5           Set graphic colours
E619 1B                     DATA  :1B
E61A B7                     ORA   A           No special action
E61B C9                     RET
                   *
                   ************************************
                   * GET 4 COLOURS INTO SCRATCH AREA *
                   ************************************
                   *
                   * Colour data from a program line are stored in
                   * scratch area SCOLT/SCOLG (#0119-011C).
                   *
                   * Exit: HL: Points to start scratch area.
                   *
E61C 211901        R4COL    LXI   H,:0119     Startaddr SCOLT/SCOLG area
E61F E5                     PUSH  H
E620 3E0F          R4C10    MVI   A,:0F
E622 CD43E7                 CALL  :E743       Get one colour (0-15)
E625 77                     MOV   M,A         Store it in scratch area
E626 23                     INX   H
E627 7D                     MOV   A,L
E628 FE1D                   CPI   :1D         4 colours done?
E62A C220E6                 JNZ   :E620       Next if not
E62D E1                     POP   H
E62E C9                     RET
                   *
                   ********************
                   * RUN basiccmd DIM *
                   ********************

                     Entry: BC: points to program line.
```

```
                  *
E62F 0A           RDIM    LDAX  B          Get nr of items
E630 03                   INX   B
E631 B7           RDM05   ORA   A
E632 C8                   RZ               Abort if no items or ready
E633 3D                   DCR   A          Item count
E634 F5                   PUSH  PSW        Preserve count
E635 CD5AE9               CALL  :E95A      Get pnr to array in HL;
                                           type in A
E638 E5                   PUSH  H          Preserve pntr
E639 CD5CCE               CALL  :CE5C      Erase array if existing
E63C E630                 ANI   :30        Get type only
E63E 110400               LXI   D,:0004    Length array element if
                                           FPT/INT
E641 FE20                 CPI   :20        String type?
E643 C248E6               JNZ   :E648      Jump if not
E646 1E02                 MVI   E,:02      Length STR array element
E648 0A           RDM10   LDAX  B          Get number of elements
E649 03                   INX   B
E64A 67                   MOV   H,A        ) In H and in L
E64B 6F                   MOV   L,A        )
E64C EB                   XCHG

                  * Calculate total required length:

E64D CD4FE7       RDM20   CALL  :E74F      Get length next dimension
E650 F5                   PUSH  PSW        Remember it
E651 3C                   INR   A          Length +1
E652 CDF0ED               CALL  :EDF0      Calc reqd space
E655 DA15DA               JC    :DA15      Run error 'NUMBER OUT
                                           OF RANGE' if total space
                                           > 64K
E658 15                   DCR   D          nr of elements -1
E659 C24DE6               JNZ   :E64D      Next element if not ready

                  * Find space in heap:

E65C 25                   DCR   H
E65D 24                   INR   H
E65E FA15DA               JM    :DA15      Run error 'NUMBER OUT OF
                                           RANGE' if > 32K reserved
E661 19                   DAD   D
E662 23                   INX   H          Size of space reqd in HL
E663 D5                   PUSH  D
E664 EB                   XCHG
E665 CD8BE6               CALL  :E68B      Get space of size needed
E668 D1                   POP   D
E669 73                   MOV   M,E        Store nr of elements
E66A 19                   DAD   D          Last element

                  * Elements into heap:

E66B F1           RDM30   POP   PSW        Get length 1 element
E66C 77                   MOV   M,A        Store it in memory
E66D 2B                   DCX   H
E66E 1D                   DCR   E
E66F C26BE6               JNZ   :E66B      Next element to memory
                  *
E672 EB                   XCHG
E673 E1                   POP   H          Get pntr to array
E674 73                   MOV   M,E        )
E675 23                   INX   H          ) Set pointer
```

```
E676 72                  MOV    M,D        )
E677 F1                  POP    PSW        Get item count in A
E678 C331E6              JMP    :E631      Next item
                 *
                 ***************************
                 * part of RUN TALK (0EE94) *
                 ***************************
                 *
                 * Entry: A: Code for osc.channel SHR 1.
                 *
E67B 29          MPT47   DAD    H
E67C 119402      RTK50   LXI    D,:0294    Addr volumes osc. 0,1
E67F E601                ANI    :01        Code SHR 1 only
E681 83                  ADD    E
E682 5F                  MOV    E,A        DE=#0294 for osc.0,1;
                                           DE=#0295 for osc.2,N
E683 7C                  MOV    A,H        Get mask
E684 2F                  CMA               Complement it
E685 EB                  XCHG              Mask + vol in DE, addr
                                           POROM/POR1M in HL
E686 A6                  ANA    M          Part to be preserved from
                                           old POROM/POR1M
E687 B3                  ORA    E          Add new volume
E688 C340EA              JMP    :EA40      Continu
                 *
                 **********************
                 * REQUEST HEAP SPACE *
                 **********************
                 *
                 * Part of Run 'DIM' (0E665).
                 * Requests space from Heap and fills it with
                 * zeroes.
                 *
                 * Entry: DE: Size needed.
                 * Exit:  HL: Points to data area (after length
                 *                bytes).
                 *            AFDE corrupted.
                 *
E68B 15          ZHREQ   DCR    D
E68C 14                  INR    D
E68D FA15DA              JM     :DA15      Run error 'NUMBER OUT OF
                                           RANGE' if >32K reqd
E690 CDC5D1              CALL   :D1C5      Run Heap request
E693 23                  INX    H
E694 23                  INX    H          HL pnts after length byte
E695 E5                  PUSH   H
E696 EB                  XCHG              Start data area in DE
E697 19                  DAD    D          End area in HL
E698 AF                  XRA    A
E699 CD7CDE              CALL   :DE7C      Load bank with '0'
E69C E1                  POP    H
E69D C9                  RET
                 *
                 ******************
                 * RUN basiccmd UT *
                 ******************
                 *
                 * Valid as direct command only.
                 *
E69E AF          RUT     XRA    A
E69F 32B902              STA    :02B9      Enable complete keyb scan
E6A2 CF                  RST    1          Go to utility
```

```
E6A3 09                    DATA  :09
                  *
                  **********************
                  * RUN basiccmd CALLM *
                  **********************
                  *
E6A4 21B3E6       RCALM    LXI   H,:E6B3     Returnaddr from Utility
E6A7 E5                    PUSH  H           on stack
E6A8 CDF8E6                CALL  :E6F8       Get UT addr in HL
E6AB E5                    PUSH  H           UT addr on stack
E6AC 0A                    LDAX  B           Get next expr
E6AD FEFF                  CPI   :FF         End marker?
E6AF C263E9                JNZ   :E963       If not: Get varptr of given
                                             variable in HL, T/L in A
E6B2 03                    INX   B
E6B3 B7           RCM10    ORA   A           No special action
E6B4 C9                    RET               On entry: Goto UT addr
                                             On exit: Back to Basic
                                             monitor
                  *
                  **********************
                  * RUN basiccmd CLEAR *
                  **********************
                  *
E6B5 CD7FD8       RCLEAR   CALL  :D87F       Get space reqd in HL
                                             (CY=1 if > 32K)
E6B8 CDBBCE                CALL  :CEBB       Must be >=4 bytes, else run
                                             error 'NUMBER OUT OF RANGE'
E6BB EB                    XCHG
E6BC 229D02                SHLD  :029D       Set Heap size
E6BF 2A9F02                LHLD  :029F       Get startaddr. textbuf
E6C2 E5                    PUSH  H
E6C3 CD23CB                CALL  :CB23       Empty Heap + symtab
E6C6 D5                    PUSH  D
E6C7 CD95D1                CALL  :D195       Set Heap to all available
E6CA E1                    POP   H
E6CB C314D2                JMP   :D214       Continu
                  *
                  ******************************
                  * Run basiccmds TRON - TROFF *
                  ******************************
                  *
                  * Sets or resets the trace flag.
                  *
                  * RTRON: Set trace flag.
                  * RTROF: Reset trace flag.
                  *
                  * Entry: none.
                  * Exit:   Z=1: Flag reset.
                  *         Z=0: Flag set.
                  *
E6CE 3EFF         RTRON    MVI   A,:FF
E6D0 321501       RTR10    STA   :0115       Set trace flag
E6D3 B7                    ORA   A           No special action
E6D4 C9                    RET
                  *
E6D5 AF           RTROF    XRA   A
E6D6 C3D0E6                JMP   :E6D0       Reset trace flag
                  *
                  *******************
                  * READ LINENUMBER *
                  *******************
                  *
                  * Entry: BC: Points to linenumber.
                  * Exit:  Z=0: Linenumber in HL.
                  *         Z=1: HL preserved.
                  *         BC updated, DE preserved, AF corrupted.
                  *
E6D9 E5           RLN     PUSH   H
E6DA 0A                   LDAX   B
E6DB 03                   INX    B
E6DC 67                   MOV    H,A       )
E6DD 0A                   LDAX   B         ) Get linenr in HL
E6DE 03                   INX    B         )
E6DF 6F                   MOV    L,A       )
E6E0 B4                   ORA    H
E6E1 CAE5E6               JZ     :E6E5     Abort if linenr is 0
E6E4 E3                   XTHL             Linenr on stack
E6E5 E1           RLN10   POP    H         Old HL or linenr in HL
E6E6 C9                   RET
                  *
                  ********************************************
                  * READ LINENUMBER AND FIND IT IN TEXTBUFFER *
                  ********************************************
                  *
                  * Entry: BC: Points to linenumber.
                  * Exit:  BC updated, DE preserved, AF corrupted
                  *         (RLNF) or preserved (RLNFI).
                  *         HL: Points to 1st linenr >= reqd. number.
                  *         CY=1: Linenumber found.
                  *         CY=0: Not found.
                  *
E6E7 CDD9E6       RLNF    CALL   :E6D9     Get linenr in HL
E6EA C3F6CA               JMP    :CAF6     Find it in textbuffer

                  * Idem as RLNF, but with error reporting:

E6ED F5           RLNFI   PUSH   PSW
E6EE CDE7E6               CALL   :E6E7     Read linenr and find it
E6F1 3E04                 MVI    A,:04
E6F3 D2F5D9               JNC    :D9F5     Run error 'UNDEFINED NUMBER'
                                           if not found
E6F6 F1                   POP    PSW
E6F7 C9                   RET
                  *
                  ******************************************
                  * RUN A INT EXPRESSION WITH 2-BYTE RESULT *
                  ******************************************
                  *
                  * Evaluates a 16-bit INT expression (in range 0-
                  * FFFF). The result is in HL.
                  *
                  * Entry: BC: Points to expression.
                  * Exit:  HL: Result.
                  *         BC updated, AFDE corrupted.
                  *
E6F8 F5           REXI2   PUSH   PSW
E6F9 D5                   PUSH   D
E6FA CD19E8               CALL   :E819     Eval arguments in num expr
                                           Result in MACC or in WORKE
E6FD 7C                   MOV    A,H
E6FE B5                   ORA    L
E6FF CA10E7               JZ     :E710     Jump if result in MACC
```

```
                * If result in WORKE:

E702 7E                 MOV    A,M         )
E703 23                 INX    H           ) Check if > 2 bytes
E704 B6                 ORA    M           )
E705 C215DA             JNZ    :DA15       Then run error 'NUMBER OUT
                                           OF RANGE'
E708 23                 INX    H
E709 7E                 MOV    A,M         )
E70A 23                 INX    H           ) Get result in HL
E70B 6E                 MOV    L,M         )
E70C 67                 MOV    H,A         )
E70D D1                 POP    D
E70E F1                 POP    PSW
E70F C9                 RET

                * If result in MACC:

E710 C5         RX210   PUSH   B
E711 E7                 RST    4           Copy MACC to reg A,B,C,D
E712 15                 DATA   :15
E713 B0                 ORA    B           Check if > 2 bytes
E714 C215DA             JNZ    :DA15       Then run error 'NUMBER OUT
                                           OF RANGE'
E717 6A                 MOV    L,D         ) Result in HL
E718 61                 MOV    H,C         )
E719 C1                 POP    B
E71A D1                 POP    D
E71B F1                 POP    PSW
E71C C9                 RET
                *
                ******************************
                * RUN A 1-BYTE INT EXPRESSION *
                ******************************
                *
                * Evaluates a 8-bit INT expression (range 0-
                * FF). Result in A.
                *
                * Entry: BC: Points to expression.
                * Exit:  A:  Result.
                *            BC updated, DEHL preserved.
                *
E71D D5         REXI1   PUSH   D
E71E E5                 PUSH   H
E71F CD19E8             CALL   :E819       Eval arguments in num expr
                                           Result in MACC or WORKE
E722 7C                 MOV    A,H
E723 B5                 ORA    L
E724 CA36E7             JZ     :E736       If HL=0: Get result frm MACC

                * Result in WORKE:

E727 7E                 MOV    A,M         )
E728 23                 INX    H           )
E729 B6                 ORA    M           ) Check if > 1 byte
E72A 23                 INX    H           )
E72B B6                 ORA    M           )
E72C C215DA             JNZ    :DA15       Then run error 'NUMBER OUT
                                           OF RANGE'
E72F 23                 INX    H
E730 7E                 MOV    A,M         Get result in A
E731 E1                 POP    H
```

```
E732 D1                    POP    D
E733 C9                    RET

                   * If result in MACC (also entry from REXF1):

E734 D5            RX110   PUSH   D
E735 E5                    PUSH   H
E736 E1            RX120   POP    H
E737 C5                    PUSH   B
E738 E7                    RST    4         Copy MACC to reg A,B,C,D
E739 15                    DATA   :15
E73A B0                    ORA    B         ) Check if > 1 byte
E73B B1                    ORA    C         )
E73C C215DA                JNZ    :DA15     Then run error 'NUMBER OUT
                                            OF RANGE'
E73F 7A                    MOV    A,D       Get result in A
E740 C1                    POP    B
E741 D1                    POP    D
E742 C9                    RET
                   *
                   ***********************************************
                   * RUN 1-BYTE INT EXPRESSION WITH LIMITED RANGE *
                   ***********************************************
                   *
                   * Entry: BC: Points to expression.
                   *        A:  Range of arguments (<=FE).
                   * Exit:  BC updated, DEHL preserved, F corrupted.
                   *        A:  Result.
                   *
E743 D5            REXIL   PUSH   D
E744 57                    MOV    D,A       Argument range in D
E745 CD1DE7                CALL   :E71D     Get value of argument in A
E748 14                    INR    D
E749 BA                    CMP    D         Out of range ? Then run
E74A D215DA                JNC    :DA15     error 'NUMBER OUT OF RANGE'
E74D D1                    POP    D
E74E C9                    RET
                   *
                   ********************************************
                   * CHECK VARIABLE TYPE AND GET ITS INT VALUE *
                   ********************************************
                   *
                   * Entry: BC: Points to expression.
                   * Exit:  Error: If string type.
                   *        If OK: Value in A (FPT: converted to INT).
                   *               BC updated, DEHL preserved.
                   *
E74F 0A            REX1    LDAX   B         Get var. type byte
E750 03                    INX    B
E751 FE20                  CPI    :20       String type?
E753 CA1ADA                JZ     :DA1A     Then run error 'TYPE
                                            MISMATCH'
E756 FE10                  CPI    :10       INT type?
E758 CA1DE7                JZ     :E71D     Then get value in A

                   * If FPT:

E75B CD08E8        REXF1   CALL   :E808     Get value in MACC
E75E E7                    RST    4         Change it to INT
E75F 48                    DATA   :48
E760 C334E7                JMP    :E734     Get value in A
                   *
```

```
                    *====================================*
                    * RUN EXPRESSIONS WITH OPERATOR PREFIX *
                    *====================================*
                    *
                    * #E763-E8ED evaluate logical, FPT, INT or STR
                    * expressions in 'operator prefix' format.
                    *
                    * Register allocation during operation:
                    *     INT/FPT: D=0:    MACC empty.
                    *              E:      Operator.
                    *              HL=0:   Result in MACC.
                    *              HL<>0:  HL points to result.
                    *     STR:     HL:     Points to string.
                    *              E:      Type of string (constant [0],
                    *                      variable [1], temporary [2]).
                    *
                    ********************************
                    * EVALUATE A LOGICAL EXPRESSION *
                    ********************************
                    *
E763 1600           REXPL   MVI   D,:00      MACC free
E765 0A                     LDAX  B          Get byte
E766 E660                   ANI   :60
E768 FE40                   CPI   :40        String ?
E76A CABDE7                 JZ    :E7BD      Then jump
E76D 0A                     LDAX  B          Get byte
E76E E61F                   ANI   :1F
E770 FE18                   CPI   :18        Relational operator?
E772 DA50E8                 JC    :E850      If not: eval expr which
                                             begins with num operator
E775 03                     INX   B
E776 FE1A                   CPI   :1A        Bracket?
E778 CA63E7                 JZ    :E763      Then ignore it

                    * Logical AND or OR:

E77B F5                     PUSH  PSW        Preserve type of operation
E77C CD63E7                 CALL  :E763      Get 1st operand
E77F F5                     PUSH  PSW        Preserve it
E780 CD63E7                 CALL  :E763      Get 2nd operand
E783 D1                     POP   D          1st operand in D
E784 F5                     PUSH  PSW        Preserve 2nd operand
E785 A2                     ANA   D          AND operation
E786 5F                     MOV   E,A        Result in E
E787 F1                     POP   PSW        2nd operand in A
E788 B2                     ORA   D          OR operation
E789 57                     MOV   D,A        Result in D
E78A F1                     POP   PSW        Type of operation in F
E78B 7A                     MOV   A,D        Result OR in A
E78C EA90E7                 JPE   :E790      Quit if OR
E78F 7B                     MOV   A,E        Result AND in A
E790 C9             RXL10   RET
                    *
                    ******************************
                    * EVALUATE STRING EXPRESSION *
                    ******************************
                    *
                    * This routine returns temporary strings before
                    * they are really free.
                    * The heap is cleared if it is a temporary string.
                    *
                    * Entry: BC: Points to expression.
```

```
                   * Exit:  BC updated, AFD corrupted.
                   *        HL: Points to string.
                   *        E:  Status.
                   *
E791 CD9DE7        REXSR    CALL   :E79D     Evaluate string expr
E794 7B                     MOV    A,E       Get status
E795 FE02                   CPI    :02       Temporary ?
E797 E5                     PUSH   H
E798 CC87D1                 CZ     :D187     Then clear heap entry
E79B E1                     POP    H
E79C C9                     RET
                   *
                   *
                   *
E79D                        END
```

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

MPT47   E67B    R4C10   E620    R4COL   E61C    RCALM   E6A4
RCLEAR  E6B5    RCM10   E6B3    RCOLG   E615    RCOLT   E60E
RDIM    E62F    RDM05   E631    RDM10   E648    RDM20   E64D
RDM30   E66B    REX1    E74F    REXF1   E75B    REXI1   E71D
REXI2   E6F8    REXIL   E743    REXPL   E763    REXSR   E791
RLN     E6D9    RLN10   E6E5    RLNF    E6E7    RLNFI   E6ED
RTK50   E67C    RTR10   E6D0    RTROF   E6D5    RTRON   E6CE
RUT     E69E    RX110   E734    RX120   E736    RX210   E710
RXL10   E790    SCRER   E602    ZHREQ   E68B
```

```
                                ORG    :E79D
                        *
                        *
                        *
                        ****************************************
                        * EVALUATE ARGUMENTS IN STRING EXPRESSION *
                        ****************************************
                        *
                        * Only '+' or compare with logical result is
                        * allowed. The right-hand side of a string expres-
                        * sion is evaluated. If it is not status 02, it is
                        * moved into the Heap. The stringpointer is saved
                        * at the varptr location.
                        * If the variable had already an old value on the
                        * heap, it is cleared, see further exit conditions.
                        *
                        * Entry: (BC): 1.....  Expr. begins with operator.
                        *              01....  Variable reference.
                        *              001...  Function call.
                        *              else    Constant.
                        * Exit:  BC updated, DEHL corrupted.
                        *        A: Type (#20).
                        *
E79D 0A                 REXPS   LDAX   B          get 1st byte
E79E 07                         RLC
E79F DABDE7                     JC     :E7BD      Jump if 1st byte is operator
E7A2 07                         RLC
E7A3 DAB3E7                     JC     :E7B3      Jump if string variable
E7A6 07                         RLC
E7A7 DAD9E9                     JC     :E9D9      Jump if string function

                        * If string constant:

E7AA 1E00                       MVI    E,:00      Status: constant
E7AC 03                         INX    B
E7AD 0A                         LDAX   B          Get string length
E7AE 60                         MOV    H,B        ) Stringpntr in HL
E7AF 69                         MOV    L,C        )
E7B0 C390E1                     JMP    :E190      Abort with BC pnts after STR

                        * If string variable:

E7B3 CD63E9             RXS10   CALL   :E963      Get varptr in HL, T/L in A
E7B6 5E                         MOV    E,M        )
E7B7 23                         INX    H          ) Stringaddr in DE
E7B8 56                         MOV    D,M        )
E7B9 EB                         XCHG              and in HL
E7BA 1E01                       MVI    E,:01      Status: variable
E7BC C9                         RET

                        * If string operation:

E7BD 0A                 ROSTR   LDAX   B          Get 1st byte
E7BE 03                         INX    B
E7BF F5                         PUSH   PSW
E7C0 D5                         PUSH   D
E7C1 CD9DE7                     CALL   :E79D      Evaluate string expression
E7C4 F1                         POP    PSW
E7C5 57                         MOV    D,A
E7C6 F1                         POP    PSW
E7C7 E5                         PUSH   H          Remember it
E7C8 53                         MOV    D,E        Type in D
```

```
E7C9 F5                 PUSH  PSW
E7CA D5                 PUSH  D
E7CB CD9DE7             CALL  :E79D     Eval 2nd string expr
E7CE F1                 POP   PSW
E7CF 57                 MOV   D,A
E7D0 F1                 POP   PSW
E7D1 C5                 PUSH  B         Save it
E7D2 44                 MOV   B,H
E7D3 4D                 MOV   C,L
E7D4 E1                 POP   H
E7D5 E3                 XTHL            Save program pointer
E7D6 C5                 PUSH  B         )
E7D7 42                 MOV   B,D       )
E7D8 4B                 MOV   C,E       ) Re-arrange registers
E7D9 EB                 XCHG            )
E7DA E1                 POP   H         )
E7DB FEC0               CPI   :C0       Operator is '+'?
E7DD CAEBE7             JZ    :E7EB     Then append 2 strings

                 * If string compare:

E7E0 CD21D1             CALL  :D121     Compare 2 strings
E7E3 CDF8E7             CALL  :E7F8     Returns operands if temp
E7E6 C1                 POP   B         restore
E7E7 5F                 MOV   E,A       Opcode in E
E7E8 C333E9             JMP   :E933     Evaluate the compare

                 * If operator is '+':

E7EB E5          ROS10  PUSH  H
E7EC CD06D1             CALL  :D106     Make 1 string out of 2
E7EF E3                 XTHL            Save pntr to result/store
                                        pntr to operand
E7F0 CDF8E7             CALL  :E7F8     Clean up heap
E7F3 E1                 POP   H         Pntr to result in HL
E7F4 C1                 POP   B         Program pntr in BC
E7F5 1E02               MVI   E,:02     Status: temporary
E7F7 C9                 RET
                 *
                 * CLEAR UP HEAP AFTER STRING OPERATION:
                 *
                 * Entry: B,C: Code for 1st resp. 2nd operand.
                 *             (0=const, 1=var, 2=temp).
                 *         DE: Points to 1st operand.
                 *         HL: Points to 2nd operand.
                 * Exit:   DEHL corrupted, AFBC preserved.
                 *
E7F8 F5          DROPS  PUSH  PSW
E7F9 79                 MOV   A,C       Get code 2nd operand
E7FA FE02               CPI   :02       Temporary?
E7FC CC87D1             CZ    :D187     Then clear string in heap
E7FF EB                 XCHG
E800 78                 MOV   A,B       Get code 1st operand
E801 FE02               CPI   :02       Temporary?
E803 CC87D1             CZ    :D187     Then clear string in heap
E806 F1                 POP   PSW
E807 C9                 RET
                 *
                 **********************************
                 * EVALUATE A NUMERIC EXPRESSION *
                 **********************************
                 *
```

```
                   * Entry: BC: Points to a numeric function argument
                   *            or a numeric expression in program.
                   * Exit:  BC updated, AFDEHL preserved.
                   *        Result in MACC.
                   *
E808 F5            REXNA   PUSH  PSW
E809 D5                    PUSH  D
E80A E5                    PUSH  H
E80B CD19E8                CALL  :E819     Eval arguments in num expr
E80E 7C                    MOV   A,H
E80F B5                    ORA   L
E810 CA15E8                JZ    :E815     Abort if result in MACC
E813 E7                    RST   4         Else: copy operand to MACC
E814 0C                    DATA  :0C
E815 E1            LOE146  POP   H
E816 D1                    POP   D
E817 F1                    POP   PSW
E818 C9                    RET
                   *
                   ******************************************
                   * EVALUATE ARGUMENTS IN NUMERIC EXPRESSION *
                   ******************************************
                   *
                   * Checks for constants, functions, variables and
                   * operators. The right-hand side of the expression
                   * is therefore evaluated. The value of the variable
                   * is stored at its varptr location.
                   *
                   * Entry: BC:   Points to expression in program.
                   *       (BC): 1.... Expr begins with operator.
                   *             01... Variable reference.
                   *             001.. Function calll.
                   *             Else  Constant.
                   *        D<>0: MACC must be preserved.
                   *
E819 1600          REXPN   MVI   D,:00     Set MACC free

                   * Called by lower levels:

E81B 0A            RXN10   LDAX  B         Get 1st byte
E81C 07                    RLC
E81D DA50E8                JC    :E850     Jump if expr starts with
                                           operator
E820 07                    RLC
E821 DA6BE9                JC    :E96B     Jump if var reference
E824 07                    RLC
E825 DA30E8                JC    :E830     Jump if function call

                   * If numeric constant:

E828 03                    INX   B         Past flag byte
E829 60                    MOV   H,B       ) HL pnts to constant
E82A 69                    MOV   L,C       )
E82B 03                    INX   B
E82C 03                    INX   B
E82D 03                    INX   B
E82E 03                    INX   B         Program pntr pnts beyond
E82F C9                    RET

                   * If numeric function:

E830 D5            RFUNN   PUSH  D
```

```
E831 7A                 MOV   A,D
E832 B7                 ORA   A
E833 CA46E8             JZ    :E846      Jump if MACC free
E836 CD18C0             CALL  :C018      Save MACC on stack
E839 CDD9E9             CALL  :E9D9      Evaluate function call
                                         result in MACC
E83C 212901             LXI   H,:0129    Addr WORKE
E83F E7                 RST   4          Copy result to WORKE
E840 0F                 DATA  :0F
E841 CD1BC0             CALL  :C01B      Restore MACC from stack
E844 D1                 POP   D
E845 C9                 RET
E846 CDD9E9     RFN10   CALL  :E9D9      Evaluate function call,
                                         result in MACC
E849 D1                 POP   D
E84A 16FF               MVI   D,:FF      Set MACC to be preserved
E84C 210000             LXI   H,:00      Flag 'result in MACC'
E84F C9                 RET

                * If expr begins with numeric operator:

E850 0A         RONUM   LDAX  B          Get operator
E851 E67F               ANI   :7F        Clip operator bit
E853 03                 INX   B
E854 FE1A               CPI   :1A        Bracket?
E856 CA1BE8             JZ    :E81B      Then ignore it
E859 15                 DCR   D
E85A 14                 INR   D          Check if D<>0
E85B C418C0             CNZ   :C018      Then save MACC on stack
E85E D5                 PUSH  D
E85F 5F                 MOV   E,A        Opcode in E
E860 21DCE8             LXI   H,:E8DC
E863 E5                 PUSH  H          Returnaddr on stack
E864 CD19E8             CALL  :E819      Get 1st operand
E867 7B                 MOV   A,E        Opcode in A
E868 E61F               ANI   :1F
E86A FE1C               CPI   :1C
E86C D29FE8             JNC   :E89F      Jump if unitary operation

                * If boolean operator:

E86F E5                 PUSH  H          Save pntr to 1st operand
E870 CD1BE8             CALL  :E81B      Get 2nd operand
E873 7C                 MOV   A,H
E874 B5                 ORA   L
E875 C27DE8             JNZ   :E87D      Jump if HL pnts to WORKE
E878 212901             LXI   H,:0129    Addr WORKE
E87B E7                 RST   4          Copy 2nd operand to WORKE
E87C 0F                 DATA  :0F
E87D E3         RON10   XTHL
E87E 7C                 MOV   A,H
E87F B5                 ORA   L
E880 CA85E8             JZ    :E885      Jump if 1st operand in MACC
E883 E7                 RST   4          Else copy it from WORKE
E884 0C                 DATA  :0C        to MACC
E885 7B         LOE153  MOV   A,E        Get opcode
E886 E61F               ANI   :1F
E888 FE10               CPI   :10
E88A D2C9E8             JNC   :E8C9      If relational operation

                * If arithmetic operation:
```

```
E88D BB                  CMP   E
E88E 21FDE8              LXI   H,:E8FD   Addr table INT routines
E891 C297E8              JNZ   :E897     Jump if INT
E894 21EEE8              LXI   H,:E8EE   Addr table FPT routines
E897 1600       RON20    MVI   D,:00     Set MACC free
E899 5F                  MOV   E,A       Opcode in E
E89A 19                  DAD   D
E89B 19                  DAD   D
E89C 19                  DAD   D         Find routine in table
E89D E3                  XTHL            Addr routine on stack; pntr
                                         to 2nd operand in HL
E89E C9                  RET             Perform routine
                *
                * If an unitary operation:
                *
                * Entry: HL: Points to operand (0 if in MACC).
                *        E:  Full opcode.
                *        A:  Lower 5 bits opcode.
                *        Returnaddr on stack (E8DC).
                * Exit:  Result in MACC.
                *        ABCDEHL preserved
                *
E89F F5         RON40    PUSH  PSW
E8A0 7C                  MOV   A,H
E8A1 B5                  ORA   L
E8A2 CAA7E8              JZ    :E8A7     If operand in MACC
E8A5 E7                  RST   4         Else: operand in MACC
E8A6 0C                  DATA  :0C
E8A7 F1         LOE156   POP   PSW
E8A8 C8                  RZ              Ready if unitary '+'
E8A9 BB                  CMP   E         Bits 6,7 opcode 0?
E8AA CABEE8              JZ    :E8BE     Then change MACC to INT
E8AD FE1E                CPI   :1E       INOT?
E8AF DABBE8              JC    :E8BB     Then change sign MACC (INT)
E8B2 C2B8E8              JNZ   :E8B8     Then convert MACC to FPT
E8B5 E7                  RST   4         Perform INOT
E8B6 6C                  DATA  :6C
E8B7 C9                  RET
                *
E8B8 E7         LOE157   RST   4         Convert MACC to FPT
E8B9 4B                  DATA  :4B
E8BA C9                  RET
                *
E8BB E7         RON44    RST   4         Change sign MACC (INT)
E8BC 60                  DATA  :60
E8BD C9                  RET
                *
E8BE FE1D       RON45    CPI   :1D
E8C0 CAC6E8              JZ    :E8C6     Then change sign MACC (FPT)
E8C3 E7                  RST   4         Convert MACC to INT
E8C4 48                  DATA  :48
E8C5 C9                  RET
                *
E8C6 E7         RON49    RST   4         Change sign MACC (FPT)
E8C7 1B                  DATA  :1B
E8C8 C9                  RET
                *
                * If relational numeric operation:
                *
                * Entry: 1st operand in MACC, 2nd operand on stack.
                *        E: Full opcode.
                *        A: Lowest 5 bits opcode.
```

```
                    * Exit:  BC preserved, DEHL corrupted.
                    *
E8C9 E1             RON50   POP   H          Get pntr 2nd operand
E8CA BB                     CMP   E
E8CB CAD6E8                 JZ    :E8D6      Jump if FPT
E8CE CD15C0                 CALL  :C015      Compare 2 INT numbers
E8D1 E1             RON55   POP   H          Kill returnaddr
E8D2 E1                     POP   H          Kill saved DE
E8D3 C333E9                 JMP   :E933      Return logical result
E8D6 CD0CC0         RON60   CALL  :C00C      Compare 2 FPT numbers
E8D9 C3D1E8                 JMP   :E8D1
                    *
                    * MOVE OPERAND:
                    *
                    * REX.. routines return here after operation.
                    * Moves operand to proper location after computing.
                    *
                    * Entry: DE and returnaddress on stack.
                    * Exit:  ABC preserved.
                    *
E8DC D1             RON30   POP   D
E8DD 15                     DCR   D
E8DE 14                     INR   D          Check D=0 (MACC free)
E8DF 16FF                   MVI   D,:FF
E8E1 210000                 LXI   H,:0000    Flag 'result in MACC'
E8E4 C8                     RZ               Abort if operand in MACC
E8E5 212901                 LXI   H,:0129    Addr WORKE
E8E8 E7                     RST   4          Copy MACC to WORKE
E8E9 0F                     DATA  :0F
E8EA CD1BC0                 CALL  :C01B      Restore old MACC from stack
E8ED C9                     RET
                    *
                    * TABLE OF JUMPS TO INT/FPT OPERATOR ROUTINES:
                    *
E8EE E7             ROFTAB  RST   4
E8EF 00                     DATA  :00        MFADD; +
E8F0 C9                     RET
                    *
E8F1 E7                     RST   4
E8F2 03                     DATA  :03        MFSUB; -
E8F3 C9                     RET
                    *
E8F4 E7                     RST   4
E8F5 09                     DATA  :09        MFDIV; /
E8F6 C9                     RET
                    *
E8F7 E7                     RST   4
E8F8 06                     DATA  :06        MFMUL; *
E8F9 C9                     RET
                    *
E8FA E7                     RST   4
E8FB 24                     DATA  :24        MPWR ; ^
E8FC C9                     RET
                    *
E8FD E7             ROITAB  RST   4
E8FE 4E                     DATA  :4E        MIADD; +
E8FF C9                     RET
                    *
E900 E7                     RST   4
E901 51                     DATA  :51        MISUB; -
E902 C9                     RET
                    *
```

```
E903 E7                       RST    4
E904 57                       DATA   :57        MIDIV; /
E905 C9                       RET
                   *
E906 E7                       RST    4
E907 54                       DATA   :54        MIMUL; *
E908 C9                       RET
                   *
E909 00                       DATA   :00
E90A 00                       DATA   :00
E90B 00                       DATA   :00
E90C 00                       DATA   :00
E90D 00                       DATA   :00
E90E 00                       DATA   :00
E90F 00                       DATA   :00
E910 00                       DATA   :00
E911 00                       DATA   :00
E912 00                       DATA   :00
E913 00                       DATA   :00
E914 00                       DATA   :00
E915 00                       DATA   :00
E916 00                       DATA   :00
E917 00                       DATA   :00
                   *
E918 E7                       RST    4          MIAND
E919 63                       DATA   :63
E91A C9                       RET
                   *
E91B E7                       RST    4          MIOR
E91C 66                       DATA   :66
E91D C9                       RET
                   *
E91E 00                       DATA   :00
E91F 00                       DATA   :00
E920 00                       DATA   :00
                   *
E921 E7                       RST    4          MIXOR
E922 69                       DATA   :69
E923 C9                       RET
                   *
E924 E7                       RST    4          MSHL
E925 6F                       DATA   :6F
E926 C9                       RET
                   *
E927 E7                       RST    4          MSHR
E928 72                       DATA   :72
E929 C9                       RET
                   *
E92A E7                       RST    4          MIREM
E92B 5A                       DATA   :5A
E92C C9                       RET
                   *
                   *
                   *
E92D                          END

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

DROPS   E7F8   LOE146 E815   LOE153 E885   LOE156 E8A7
LOE157 E8B8   REXNA   E808   REXPN   E819   REXPS   E79D
```

RFN10	E846	RFUNN	E830	ROFTAB	E8EE	ROITAB	E8FD
RON10	E87D	RON20	E897	RON30	E8DC	RON40	E89F
RON44	E8BB	RON45	E8BE	RON49	E8C6	RON50	E8C9
RON55	E8D1	RON60	E8D6	RONUM	E850	ROS10	E7EB
ROSTR	E7BD	RXN10	E81B	RXS10	E7B3		

```
                        ORG   :E92D
                 *
                 *
                 *
                 *************************
                 * LENGTH OF BLOCK IN BC *
                 *************************
                 *
                 * Part of Run 'CLEAR' (D214).
                 *
E92D CD1ADE      MPT45  CALL  :DE1A      Calc. length of block
E930 44                 MOV   B,H        )
E931 4D                 MOV   C,L        ) Length in BC
E932 C9                 RET
                 *
                 **********************
                 * EVALUATE A COMPARE *
                 **********************
                 *
                 * Decodes flags and opcode to a truthtable.
                 * Following a XFCOMP or a XICOMP by a jump to
                 * E933 leaves FF (true) or 00 (false) in A as
                 * result.
                 *
                 * Entry: E: Opcode.
                 *        F: Flags.
                 * Exit:  BCDEF preserved, HL corrupted.
                 *        A: Truth value.
                 *
E933 F5          ROREL  PUSH  PSW        Save flags
E934 7B                 MOV   A,E        )
E935 E60F               ANI   :0F        ) Calc offset
E937 87                 ADD   A          )
E938 87                 ADD   A          )
E939 2143E9             LXI   H,:E943    Base addr
E93C CD30DE             CALL  :DE30      Add offset to base
E93F F1                 POP   PSW        Restore flags
E940 3EFF               MVI   A,:FF      Init truth value
E942 E9                 PCHL             Goto routine
                 *
E943 F0          ROGEQ  RP               FF if MACC >= M
E944 C358E9             JMP   :E958      (S=0)
                 *
E947 FA58E9      ROGT   JM    :E958      FF if MACC > M
E94A 00                 NOP              (S=0 and Z=0)
                 *
E94B C0          RONEQ  RNZ              FF if MACC <> M
E94C C358E9             JMP   :E958      (Z=0)
                 *
E94F C8          ROLEQ  RZ               FF if MACC <= M
E950 00                 NOP              (S=1 or Z=1)
E951 00                 NOP
E952 00                 NOP
                 *
E953 F8          ROLT   RM               FF if MACC < M
E954 C358E9             JMP   :E958      (S=1)
                 *
E957 C8          ROEQ   RZ               FF if MACC = M
                                         (Z=1)
                 *
E958 2F          RFALSE CMA              00 if condition false
E959 C9                 RET
```

```
                    *
                    ***************************
                    * RUN A VARIABLE REFERENCE *
                    ***************************
                    *
                    * Produces a pointer to the value of a variable.
                    * The variable may be simple or subscripted and
                    * of any type. If subscripted, subscripts are
                    * evaluated and checked for range.
                    *
                    * Entry: BC: Points to encoded variable.
                    *        D:  0 if MACC free.
                    * Exit:  BC updated, DE preserved, F corrupted.
                    *        HL: Pqints to variable storage.
                    *        A:  Type of variable from symbol table.
                    *
E95A D5             RARRN   PUSH  D
E95B AF                     XRA   A          Array name only
                    *
E95C 1600           RVREN   MVI   D,:00
E95E C33DD7                 JMP   :D73D      Run varptr
                    *
E961 FF                     DATA  :FF
E962 FF                     DATA  :FF
                    *
                    ************************************
                    * RUN VARPTR (ARRAY WITH ARGUMENTS) *
                    ************************************
                    *
                    * Runs a varptr with A=FF, D=0.
                    *
                    * Exit: BC updated, DE preserved, MACC corrupted.
                    *       HL: Varptr.
                    *       A:  T/L byte.
                    *
E963 D5             RVAR    PUSH  D
E964 3EFF                   MVI   A,:FF      Set mask for arrays (not
                                             name only)
E966 C35CE9                 JMP   :E95C      Run varptr
                    *
E969 FF                     DATA  :FF
E96A FF                     DATA  :FF
                    *
                    **************************
                    * RUN A VARIABLE POINTER *
                    **************************
                    *
                    * RVARE: Entry for arrays.
                    * RVR05: 'Normal' entry.
                    *
                    * Entry: BC: Points to var.reference in program.
                    *        A:  Mask.
                    *        D:  0 if MACC free.
                    * Exit:  HL: Varptr (for strings: stringpntr).
                    *        A:  T/L of symtab.
                    *        BC updated, DE preserved.
                    *
E96B 3EFF           RVARE   MVI   A,:FF      Not array name only
E96D F5             RVR05   PUSH  PSW
E96E D5                     PUSH  D
E96F 0A                     LDAX  B          )
E970 03                     INX   B          )
```

```
E971 E63F                ANI   :3F        ) Get offset in symtab
E973 57                  MOV   D,A        ) in DE
E974 0A                  LDAX  B          )
E975 03                  INX   B          )
E976 5F                  MOV   E,A        )
E977 2AA102              LHLD  :02A1      Get start symtab
E97A 19                  DAD   D          HL pnts to actual addr
                                          in symtab
E97B D1                  POP   D
E97C F1                  POP   PSW        Restore mask
E97D A6                  ANA   M          And with T/L byte
E97E E640                ANI   :40        Bit 6 only
E980 7E                  MOV   A,M
E981 23                  INX   H
E982 C8                  RZ               Abort if simple variable or
                                          array name only

                * Compute actual position in array:

E983 15                  DCR   D
E984 14                  INR   D          Check D=0: MACC free
E985 C418C0              CNZ   :C018      Save MACC on stack if not
E988 D5                  PUSH  D
E989 F5                  PUSH  PSW
E98A 5E                  MOV   E,M        ) Get pntr from symtab
E98B 23                  INX   H          ) in DE
E98C 56                  MOV   D,M        )
E98D 23                  INX   H
E98E 7A                  MOV   A,D        ) Check if undimensioned
E98F B3                  ORA   E          )
E990 3E0F       ERRUA    MVI   A,:0F      Then run error
E992 CAF5D9              JZ    :D9F5      'UNDEFINED ARRAY'
E995 D5                  PUSH  D
E996 210000              LXI   H,:0000    Init index
E999 E3                  XTHL             Pntr to array in HL
E99A 0A                  LDAX  B          Get nr of subscripts
E99B 03                  INX   B
E99C 57                  MOV   D,A        in D
E99D BE                  CMP   M          Comp. with nr of dimensions
E99E 23                  INX   H
E99F C229DA              JNZ   :DA29      Run 'SUBSCRIPT ERROR' if
                                          not identical
E9A2 CD4FE7     RVR10    CALL  :E74F      Get variable type in A
E9A5 BE         RVR15    CMP   M
E9A6 DAB1E9              JC    :E9B1      If value in A is offset
E9A9 CAB1E9              JZ    :E9B1
E9AC 3E05                MVI   A,:05      If subscript <0 or >FF:
E9AE C3F5D9              JMP   :D9F5      Run 'SUBSCRIPT ERROR'

                * Calc reference to Nth array element
                * via (a1*(d2+1)+a2*(d3+1)+..+aN). aN
                * is argument, dN is dimension:

E9B1 E3         RVR20    XTHL             Restore HL=00
E9B2 CD30DE              CALL  :DE30      Add offset to index
E9B5 15                  DCR   D          decr nr of arguments
E9B6 E3                  XTHL
E9B7 23                  INX   H
E9B8 CAC5E9              JZ    :E9C5      If no more subscipts
E9BB 7E                  MOV   A,M        Next dimension
E9BC 3C                  INR   A
E9BD E3                  XTHL
```

```
E9BE CD8FDE              CALL  :DE8F      Multiply index by it
E9C1 E3                  XTHL
E9C2 C3A2E9              JMP   :E9A2      Process next subscript
                 *
E9C5 EB          RVR30   XCHG
E9C6 E1                  POP   H          Get index to array from
                                          symtab
E9C7 F1                  POP   PSW        Get type byte
E9C8 F5                  PUSH  PSW
E9C9 E630                ANI   :30        Bits 5,6 only
E9CB FE20                CPI   :20
E9CD 29                  DAD   H          Add offset to element
E9CE CAD2E9              JZ    :E9D2
E9D1 29                  DAD   H
E9D2 19          RVR40   DAD   D          Abs addr of element in HL
                                          (STR: pntr to string)
E9D3 F1                  POP   PSW
E9D4 D1                  POP   D
E9D5 C41BC0              CNZ   :C01B      Evt retrieve MACC from stack
E9D8 C9                  RET
                 *
                 ****************************
                 * EVALUATE A FUNCTION CALL *
                 ****************************
                 *
                 * Finds address of function routine in
                 * indirection table (#E9F0) and performs the
                 * function routine.
                 *
                 * Entry: BC: Points to function label (#20)
                 *            in program line.
                 * Exit:  MACC: Numeric result.
                 *        BC updated, AFDEHL corrupted.
                 *
E9D9 03          RFUN    INX   B          Pnts past label
E9DA 0A                  LDAX  B          Get function code
E9DB 03                  INX   B
E9DC 6F                  MOV   L,A        Code in HL
E9DD 2600                MVI   H,:00
E9DF 11F0E9              LXI   D,:E9F0    Get startaddr. table
E9E2 29                  DAD   H          Code *2
E9E3 19                  DAD   D          Add offset to startaddr
E9E4 5E                  MOV   E,M        lobyte addr in E
E9E5 23                  INX   H
E9E6 7E                  MOV   A,M        hibyte addr in A
E9E7 F680                ORI   :80        Set msb = 1
E9E9 57                  MOV   D,A        hibyte in D
E9EA BE                  CMP   M          Was it already E...?
E9EB D5                  PUSH  D          Addr function on stack
E9EC CC08E8              CZ    :E808      Then evaluate 1st num
                                          expr, result in MACC
E9EF C9                  RET              Go to function routine
                 *
                 *******************************
                 * FUNCTION INDIRECTION TABLE *
                 *******************************
                 *
                 * Startaddress table is E9F0. The function code
                 * is given between brackets.
                 * If the msb of the address is set, the first
                 * argument is a numeric one.
```

```
                          * In the textbuffer, the Basic fuctions are
                          * indicated by 20 xx, (xx is function code).
                          *
E9F0 50EA                 FUNIT    DBL     :EA50       (00)   ABS
E9F2 53EA                          DBL     :EA53       (01)   ALOG
E9F4 CB6A                          DBL     :6ACB       (02)   ASC
E9F6 D26A                          DBL     :6AD2       (03)   CHR$
E9F8 B86A                          DBL     :6AB8       (04)   CURY
E9FA BE6A                          DBL     :6ABE       (05)   CURX
E9FC 56EA                          DBL     :EA56       (06)   EXP
E9FE 59EA                          DBL     :EA59       (07)   FRAC
EA00 436B                          DBL     :6B43       (08)   FRE
EA02 5CEB                          DBL     :EB5C       (09)   FREQ
EA04 796B                          DBL     :6B79       (0A)   GETC
EA06 836A                          DBL     :6A83       (0B)   HEX$
EA08 82EB                          DBL     :EB82       (0C)   INP
EA0A 8DEB                          DBL     :EB8D       (0D)   INT
EA0C E26A                          DBL     :6AE2       (0E)   LEFT$
EA0E C46A                          DBL     :6AC4       (0F)   LEN
EA10 A16B                          DBL     :6BA1       (10)   VARPTR
EA12 5CEA                          DBL     :EA5C       (11)   LOG
EA14 5FEA                          DBL     :EA5F       (12)   LOGT
EA16 A76B                          DBL     :6BA7       (13)   XMAX
EA18 AE6B                          DBL     :6BAE       (14)   YMAX
EA1A 0E6B                          DBL     :6B0E       (15)   MID$
EA1C C16B                          DBL     :6BC1       (16)   PDL
EA1E 166C                          DBL     :6C16       (17)   PEEK
EA20 1D6C                          DBL     :6C1D       (18)   PI
EA22 FF6A                          DBL     :6AFF       (19)   RIGHT$
EA24 27EC                          DBL     :EC27       (1A)   RND
EA26 9D6C                          DBL     :6C9D       (1B)   SCRN
EA28 7BEC                          DBL     :EC7B       (1C)   SGN
EA2A 8C6A                          DBL     :6A8C       (1D)   SPC
EA2C 62EA                          DBL     :EA62       (1E)   SQR
EA2E 77EA                          DBL     :EA77       (1F)   STR$
EA30 A26A                          DBL     :6AA2       (20)   TAB
EA32 256B                          DBL     :6B25       (21)   VAL
EA34 65EA                          DBL     :EA65       (22)   SIN
EA36 68EA                          DBL     :EA68       (23)   COS
EA38 6BEA                          DBL     :EA6B       (24)   TAN
EA3A 6EEA                          DBL     :EA6E       (25)   ASIN
EA3C 71EA                          DBL     :EA71       (26)   ACOS
EA3E 74EA                          DBL     :EA74       (27)   ATN
                          *
                          *
                          *
EA40                               END
```

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

ERRUA   E990   FUNIT   E9F0   MPT45   E92D   RARRN   E95A
RFALSE  E958   RFUN    E9D9   ROEQ    E957   ROGEQ   E943
ROGT    E947   ROLEQ   E94F   ROLT    E953   RONEQ   E94B
ROREL   E933   RVAR    E963   RVARE   E96B   RVR05   E96D
RVR10   E9A2   RVR15   E9A5   RVR20   E9B1   RVR30   E9C5
RVR40   E9D2   RVREN   E95C
```

```
                             ORG    :EA40
                    *
                    *
                    *
                    ****************************
                    * part of RUN TALK (0E67B) *
                    ****************************
                    *
                    * Sets oscillator volumes.
                    *
                    * Entry: A:  New volume.
                    *        HL: Address POROM/POR1M.
                    *
EA40 77             MPT48    MOV    M,A       Update POROM/POR1M
EA41 1170FA                  LXI    D,:FA70
EA44 19                      DAD    D         HL= PORO/POR1
EA45 77                      MOV    M,A       Update osc.volume
EA46 E1                      POP    H         Get parameter pntr
EA47 23             RTK55    INX    H         Pnts to next code
EA48 C367CD                  JMP    :CD67     Handle next code
                    *
EA4B FF                      DATA   :FF
EA4C FF                      DATA   :FF
EA4D FF                      DATA   :FF
EA4E FF                      DATA   :FF
EA4F FF                      DATA   :FF
                    *
                    *************************
                    * RUN basicfunction ABS *
                    *************************
                    *
EA50 E7             RABS     RST    4         MFABS
EA51 18                      DATA   :18
EA52 C9                      RET
                    *
                    **************************
                    * RUN basicfunction ALOG *
                    **************************
                    *
EA53 E7             RALOG    RST    4         MALOG
EA54 30                      DATA   :30
EA55 C9                      RET
                    *
                    *************************
                    * RUN basicfunction EXP *
                    *************************
                    *
EA56 E7             REXP     RST    4         MEXP
EA57 2A                      DATA   :2A
EA58 C9                      RET
                    *
                    **************************
                    * RUN basicfunction FRAC *
                    **************************
                    *
EA59 E7             RFRAC    RST    4         MFRAC
EA5A 21                      DATA   :21
EA5B C9                      RET
                    *
                    *************************
                    * RUN basicfunction LOG *
                    *************************
```

```
                      *
 EA5C E7              RLOG    RST    4          MLOG
 EA5D 27                      DATA   :27
 EA5E C9                      RET
                      *
                      **************************
                      * RUN basicfunction LOGT *
                      **************************
                      *
 EA5F E7              RLOGT   RST    4          MLOGT
 EA60 2D                      DATA   :2D
 EA61 C9                      RET
                      *
                      *************************
                      * RUN basicfunction SQR *
                      *************************
                      *
 EA62 E7              RSQR    RST    4          MSQR
 EA63 33                      DATA   :33
 EA64 C9                      RET
                      *
                      *************************
                      * RUN basicfunction SIN *
                      *************************
                      *
 EA65 E7              RSIN    RST    4          MSIN
 EA66 36                      DATA   :36
 EA67 C9                      RET
                      *
                      *************************
                      * RUN basicfunction COS *
                      *************************
                      *
 EA68 E7              RCOS    RST    4          MCOS
 EA69 39                      DATA   :39
 EA6A C9                      RET
                      *
                      *************************
                      * RUN basicfunction TAN *
                      *************************
                      *
 EA6B E7              RTAN    RST    4          MTAN
 EA6C 3C                      DATA   :3C
 EA6D C9                      RET
                      *
                      **************************
                      * RUN basicfunction ASIN *
                      **************************
                      *
 EA6E E7              RASIN   RST    4          MASIN
 EA6F 3F                      DATA   :3F
 EA70 C9                      RET
                      *
                      **************************
                      * RUN basicfunction ACOS *
                      **************************
                      *
 EA71 E7              RACOS   RST    4          MACOS
 EA72 42                      DATA   :42
 EA73 C9                      RET
                      *
                      *
```

```
                    **************************
                    * RUN basicfunction ATAN *
                    **************************
                    *
EA74 E7             RATN    RST     4          MATAN
EA75 45                     DATA    :45
EA76 C9                     RET
                    *
                    **************************
                    * RUN basicfunction STR$ *
                    **************************
                    *
                    * Converts a FPT number into a string.
                    *
EA77 CD9BCE         RSTR    CALL    :CE9B      Convert MACC for FPT output
                                               string in DECBUF
EA7A 00                     NOP
EA7B 00                     NOP
EA7C 00                     NOP
EA7D 2A33C0         RST20   LHLD    :C033      Get addr DECBUF
EA80 1E01                   MVI     E,:01      Pretend it is a variable
EA82 C9                     RET
                    *
                    **************************
                    * RUN basicfunction HEX$ *
                    **************************
                    *
                    * Converts a INT number into an equivalent string.
                    *
EA83 CD08E8         RHEX    CALL    :E808      Eval expr, result in MACC
EA86 CD2DC0                 CALL    :C02D      Conv. MACC to HEX for output
EA89 C37DEA                 JMP     :EA7D      Get addr DECBUF, pretend it
                                               is a variable
                    *
                    *************************
                    * RUN basicfunction SPC *
                    *************************
                    *
                    * Returns a string of a number of spaces.
                    * From SPC10 used by TAB if DOUTC<>0.
                    *
EA8C CD1DE7         RSPC    CALL    :E71D      Get nr of spaces in A

                    * Entry from RTAB:

EA8F CD8BD1         SPC10   CALL    :D18B      Get place in heap for string
                                               of spaces
EA92 E5                     PUSH    H          Save pntr to heap
EA93 B7             SPC20   ORA     A
EA94 CA9EEA                 JZ      :EA9E      Jump if ready
EA97 23                     INX     H
EA98 3620                   MVI     M,:20      Space into heap
EA9A 3D                     DCR     A
EA9B C393EA                 JMP     :EA93      Next space
EA9E E1             SPC30   POP     H          HL pnts to start string
EA9F C3DFEA                 JMP     :EADF      Pretend it is a temp string
                    *
                    *************************
                    * RUN basicfunction TAB *
                    *************************
                    *
                      Returns a string of spaces to move cursor to a
```

```
                   * given character position (only to the right).
                   * Works only if output switch DOUTC=0, else it
                   * returns one space only.
                   *
EAA2 CD60CE        RTAB    CALL   :CE60      Get nr of tabs in L,
                                             DOUTC in A
EAA5 B7                    ORA    A          Check output direction
EAA6 3E01                  MVI    A,:01      Init 1 space
EAA8 C28FEA                JNZ    :EA8F      Jump if DOUTC<>0
EAAB 7D                    MOV    A,L        Get nr of tabs
EAAC 00                    NOP
EAAD 00                    NOP
EAAE EF                    RST    5          Ask cursor pos and size
EAAF 0C                    DATA   :0C        char screen
EAB0 95                    SUB    L          Calc nr of spaces reqd
EAB1 D28FEA                JNC    :EA8F      Run SPC if not past tab pos
EAB4 AF                    XRA    A          If past TAB:
EAB5 C38FEA                JMP    :EA8F      Run SPC for no spaces
                   *
                   **************************
                   * RUN basicfunction CURX *
                   **************************
                   *
EAB8 EF            RCURX   RST    5          Ask cursor pos and size
EAB9 0C                    DATA   :0C        char screen
EABA 7D                    MOV    A,L        X-coord in A
EABB C37CEB                JMP    :EB7C      and into MACC
                   *
                   **************************
                   * RUN basicfunction CURY *
                   **************************
                   *
EABE EF            RCURY   RST    5          Ask cursor pos and size
EABF 0C                    DATA   :0C        char screen
EAC0 7C                    MOV    A,H        Y-coord in A
EAC1 C37CEB                JMP    :EB7C      and in MACC
                   *
                   *************************
                   * RUN basicfunction LEN *
                   *************************
                   *
                   * Given a string, returns length of the string.
                   *
EAC4 CD91E7        RLEN    CALL   :E791      Eval string expr
EAC7 7E            RLE10   MOV    A,M        Get length in A
EAC8 C37CEB                JMP    :EB7C      and into MACC
                   *
                   *************************
                   * RUN basicfunction ASC *
                   *************************
                   *
                   * Given a string, returns ASCII value of 1st char.
                   *
EACB CD91E7        RASC    CALL   :E791      Eval string expr
EACE 7E                    MOV    A,M        Get length in A
EACF C37ECF                JMP    :CF7E      Check if length is 0; get
                                             1st char in MACC if not
                   *
                   **************************
                   * RUN basicfunction CHR$ *
                   **************************
                   *
```

```
EAD2 CD1DE7     RCHR    CALL   :E71D      Get argument value in A
EAD5 F5                 PUSH   PSW        Save it
EAD6 3E01               MVI    A,:01
EAD8 CD8BD1             CALL   :D18B      Find place in heap for
                                          a 1-byte string
EADB F1                 POP    PSW        Get argument
EADC 23                 INX    H
EADD 77                 MOV    M,A        Store it in Heap
EADE 2B                 DCX    H          Pnts to length byte

                * Entry from 'SPC':

EADF 1E02       RCR10   MVI    E,:02      Status: temporary
EAE1 C9                 RET
                *
                ***************************
                * RUN basicfunction LEFT$ *
                ***************************
                *
                * Given a string, returns a number of characters
                * from the left end.
                *
EAE2 CD9DE7     RLEFT   CALL   :E79D      Eval string expr
EAE5 E5                 PUSH   H          Save string pntr
EAE6 D5                 PUSH   D
EAE7 CD1DEB             CALL   :EB1D      Reqd length in A
EAEA 1600               MVI    D,:00
EAEC 5F         RLF10   MOV    E,A        Length in DE
EAED CD4FD1     RLF20   CALL   :D14F      Extract substring
EAF0 D215DA             JNC    :DA15      Evt. run error 'NUMBER OUT
                                          OF RANGE'
EAF3 D1                 POP    D
EAF4 E3                 XTHL
EAF5 7B                 MOV    A,E        Get status
EAF6 FE02               CPI    :02        Temporary?
EAF8 CC87D1             CZ     :D187      Then clear heap entry
EAFB E1                 POP    H
EAFC 1E02               MVI    E,:02      Status temporary
EAFE C9                 RET
                *
                ****************************
                * RUN basicfunction RIGHT$ *
                ****************************
                *
                * Extracts a number of characters from the
                * right end of a given string.
                *
EAFF CD9DE7     RRIGHT  CALL   :E79D      Eval string expr
EB02 E5                 PUSH   H          Save string pntr
EB03 D5                 PUSH   D
EB04 CD1DEB             CALL   :EB1D      Get length substring
EB07 5F                 MOV    E,A        in E
EB08 7E                 MOV    A,M        Get total string length
EB09 93                 SUB    E
EB0A 57                 MOV    D,A        Startposition in D
EB0B C3EDEA             JMP    :EAED      Extract substring
                *
                **************************
                * RUN basicfunction MID$ *
                **************************
                *
EB0E CD9DE7     RMID    CALL   :E79D      Eval string expr
```

```
EB11 E5                          PUSH  H          Save string pntr
EB12 D5                          PUSH  D
EB13 CD1DEB                      CALL  :EB1D      Get startposition
EB16 57                          MOV   D,A        in D
EB17 CD1DE7                      CALL  :E71D      Get length in A
EB1A C3ECEA                      JMP   :EAEC      Extract substring
                        *
                        ******************************
                        * GET VALUE OF ARGUMENT IN A *
                        ******************************
                        *
                        * Exit: DEHL preserved.
                        *
EB1D E5                 REXIK    PUSH  H
EB1E D5                          PUSH  D
EB1F CD1DE7                      CALL  :E71D      Get value of argument in A
EB22 D1                          POP   D
EB23 E1                          POP   H
EB24 C9                          RET
                        *
                        *************************
                        * RUN basicfunction VAL *
                        *************************
                        *
                        * Takes a string and converts it to a FPT number.
                        *
EB25 CD91E7             RVAL     CALL  :E791      Eval string expr
EB28 C5                          PUSH  B
EB29 7E                 SUEPT    MOV   A,M        Length of string in A
EB2A 323401                      STA   :0134      and in EFECT
EB2D 23                          INX   H          HL pnts to 1st string byte
EB2E 223201                      SHLD  :0132      Addr into EFEPT
EB31 0E00                        MVI   C,:00      Char count
EB33 213501                      LXI   H,:0135
EB36 3601                        MVI   M,:01      Input from string
EB38 CD1EC0                      CALL  :C01E      encode FPT number into MACC
EB3B 35                          DCR   M          Input from keyboard
EB3C 3E0A                        MVI   A,:0A      If over/underflow: run
EB3E D2F5D9                      JNC   :D9F5      error 'INVALID NUMBER'
EB41 C1                          POP   B
EB42 C9                          RET
                        *
                        *************************
                        * RUN basicfunction FRE *
                        *************************
                        *
                        * Returns a INT given size of free RAM space.
                        * Result in MACC.
                        * FR2BY: Also used to copy HL into MACC.
                        *
                        * Exit: BC preserved, AFDEHL corrupted.
                        *
EB43 CD51EB             RFRE     CALL  :EB51      Calc free RAM space in HL
EB46 AF                 FR2BY    XRA   A
EB47 C5                          PUSH  B
EB48 D5                          PUSH  D
EB49 4C                          MOV   C,H        ) Free space in CD
EB4A 55                          MOV   D,L        )
EB4B 47                          MOV   B,A        A,B=0
EB4C E7                          RST   4          Copy reg A,B,C,D into MACC
EB4D 12                          DATA  :12
EB4E D1                          POP   D
```

```
EB4F C1                POP   B
EB50 C9                RET
               *
               ****************************
               * CALCULATE FREE RAM SPACE *
               ****************************
               *
               * Exit: HL: Free RAM space.
               *       DE: STBUSE.
               *       ABC preserved, F corrupted.
               *
EB51 2AA302    SIZE    LHLD  :02A3     Get end symtab
EB54 EB                XCHG            in DE
EB55 2AA502            LHLD  :02A5     Get bottom screen RAM
EB58 CD1ADE            CALL  :DE1A     Calc. free space in HL
EB5B C9                RET
               *
               **************************
               * RUN basicfunction FREQ *
               **************************
               *
               * Given a frequency in Hz, returns a period in
               * 'oscillator cycles' (INT).
               *
               * Entry: MACC: Value for freq.
               * Exit:  BC preserved, AFDEHL corrupted.
               *
EB5C 212901    RFREQ   LXI   H,:0129   Startaddr scratch area for
                                       expression evaluation
EB5F E7                RST   4         Copy reqd freq to scratch
EB60 0F                DATA  :0F       area
EB61 E5                PUSH  H         Save startaddr scratch area
EB62 21EDD0            LXI   H,:D0ED   Addr sound constant
EB65 E7                RST   4         Sound constant into MACC
EB66 0C                DATA  :0C
EB67 E1                POP   H         Get start scratch area
EB68 E7                RST   4         Calc sound const/reqd freq
EB69 09                DATA  :09
EB6A E7                RST   4         Change it to INT
EB6B 48                DATA  :48
EB6C C5                PUSH  B
EB6D E7                RST   4         Copy result to reg A,B,C,D
EB6E 15                DATA  :15
EB6F B0                ORA   B         > 64K ? Then run error
EB70 C215DA            JNZ   :DA15     'NUMBER OUT OF RANGE'
EB73 C1                POP   B
EB74 C9                RET
               *
               *********************
               * DATA - (not used) *
               *********************
               *
EB75 15        L0E235  DATA  :15       Sound constant
EB76 F4                DATA  :F4
EB77 24                DATA  :24
EB78 00                DATA  :00
               *
               **************************
               * RUN basicfunction GETC *
               **************************
               *
               * Gets one character from keyboard. Returns its
```

```
                 * ASCII value in MACC; 0 if no inputs.
                 * FR1BY: Also used to copy 1 byte into MACC.
                 *
EB79 CDBBD6      RGETC   CALL  :D6BB     Scan keyboard, result in A
EB7C 6F          FR1BY   MOV   L,A       Result in L
EB7D 2600                MVI   H,:00
EB7F C346EB              JMP   :EB46     Result into MACC
                 *
                 ************************
                 * RUN basicfunction INP *
                 ************************
                 *
                 * Reads a byte from a Real World address (DCE-bus).
                 *
EB82 CD1DE7      RINP    CALL  :E71D     Get RW addr in A
EB85 57                  MOV   D,A       and in D
EB86 CDE0D8              CALL  :D8E0     Get input from DCE-bus
EB89 7B                  MOV   A,E       Result in A
EB8A C37CEB              JMP   :EB7C     Result into MACC
                 *
                 ************************
                 * RUN basicfunction INT *
                 ************************
                 *
                 * Returns a integer FPT value, just less than the
                 * FPT argument given.
                 * REMARK: Routine is wrong if -1 < nr < 0. Then
                 *         the result is -1 !
                 *
EB8D C5          RINT    PUSH  B
EB8E E7                  RST   4         Copy MACC to reg A,B,C,D
EB8F 15                  DATA  :15
EB90 C1                  POP   B
EB91 E7                  RST   4         Change MACC to INT, and then
EB92 1E                  DATA  :1E       to FPT
EB93 21F1D0              LXI   H,:D0F1   Addr FPT(-1)
EB96 B7                  ORA   A
EB97 F29CEB              JP    :EB9C     Abort if positive
EB9A E7                  RST   4         Add -1 if MACC negative
EB9B 00                  DATA  :00
EB9C C9          L0E239  RET
                 *
                 *********************
                 * DATA - (not used) *
                 *********************
                 *
EB9D 81          L0E240  DATA  :81       FPT (-1)
EB9E 80                  DATA  :80
EB9F 00                  DATA  :00
EBA0 00                  DATA  :00
                 *
                 ***************************
                 * RUN basicfunction VARPTR *
                 ***************************
                 *
EBA1 CD63E9      RVPT    CALL  :E963     Get varptr in HL, T/L in A
EBA4 C346EB              JMP   :EB46     Varptr into MACC
                 *
                 *************************
                 * RUN basicfunction XMAX *
                 *************************
                 *
```

```
EBA7 CDB4EB        RXMAX    CALL   :EBB4      Get max Y,X-coord graph area
EBAA EB                     XCHG              Max X-coord in HL
EBAB C346EB                 JMP    :EB46      and into MACC
                   *
                   *************************
                   * RUN basicfunction YMAX *
                   *************************
                   *
EBAE CDB4EB        RYMAX    CALL   :EBB4      Get max Y,X-coord graph area
EBB1 C37CEB                 JMP    :EB7C      Max Y-coord into MACC
                   *
                   *****************************************
                   * GET MAX. Y,X-COORDINATES GRAPHICS AREA *
                   *****************************************
                   *
                   * Exit: DE: Max. X-coordinate.
                   *       A:  Max. Y-coordinate.
                   *       BC preserved.
                   *
EBB4 210000        L0E245   LXI    H,:0000    ) Coord dot 0,0
EBB7 C5                     PUSH   B          )
EBB8 4C                     MOV    C,H        )
EBB9 EF                     RST    5          Ask colour of point and
EBBA 27                     DATA   :27        size graphics screen
EBBB DA02E6                 JC     :E602      Evt run screen error
EBBE 78                     MOV    A,B        Max Y-coord in A
EBBF C1                     POP    B
EBC0 C9                     RET
                   *
                   ************************
                   * RUN basicfunction PDL *
                   ************************
                   *
                   * A given paddle channel is enabled. Counter 0 is
                   * set to FFFF. The counter is read over and over
                   * until it is counted out.
                   *
                   * Exit: BC updated, AFDEHL corrupted.
                   *
EBC1 3E05          RPDL     MVI    A,:05
EBC3 CD43E7                 CALL   :E743      Get paddle select (0-5)
EBC6 57                     MOV    D,A        into D
EBC7 3A4000                 LDA    :0040      Get POROM
EBCA E6F8                   ANI    :FB        ROM/cass.select only
EBCC B2                     ORA    D          OR with paddle select
EBCD F608                   ORI    :08        Paddle enable
EBCF CD08D8                 CALL   :D808      Load PORO/POROM
EBD2 C5                     PUSH   B
EBD3 3E30                   MVI    A,:30
EBD5 0106FC                 LXI    B,:FC06    Addr 8253 cmd word
EBD8 02                     STAX   B          Select ch.0, mode 0, 2 byte
EBD9 21FFFF                 LXI    H,:FFFF
EBDC 2200FC                 SHLD   :FC00      Load counter ch.0
EBDF 3A01FD                 LDA    :FD01      Get pdl timer trig impulse
                                              (Useless: A is cleared in
                                              0EBE3)
EBE2 EB            PDL10    XCHG              DE = FFFF
EBE3 3E00                   MVI    A,:00
EBE5 02                     STAX   B          (FC06)=00: counter 0, latch
                                              operation
EBE6 2A00FC                 LHLD   :FC00      Get contents counter 0
EBE9 CD14DE                 CALL   :DE14      Compare HL-DE
```

```
EBEC DAE2EB                 JC      :EBE2      Again if DE > HL
EBEF CD26DE                 CALL    :DE26      HL = 2-compl. of HL
EBF2 11CEFF                 LXI     D,:FFCE    ) Substract 49
EBF5 19                     DAD     D          )
EBF6 DAFCEB                 JC      :EBFC      If result negative
EBF9 210000                 LXI     H,:0000
EBFC 7C            PDL20    MOV     A,H
EBFD B7                     ORA     A
EBFE CA06EC                 JZ      :EC06
EC01 2EFF                   MVI     L,:FF
EC03 00                     NOP
EC04 00                     NOP
EC05 00                     NOP
EC06 3E36          PDL30    MVI     A,:36
EC08 02                     STAX    B          (FC06)=#36: Chan 0, mode 3
EC09 3A4000                 LDA     :0040      Get PORO/POROM
EC0C E6F0                   ANI     :F0        Disable paddle operation
EC0E CD06D8                 CALL    :D806      Load PORO/POROM
EC11 C1                     POP     B
EC12 7D                     MOV     A,L        A=0 if result negative,
                                               else FF
EC13 C37CEB                 JMP     :EB7C      Move A into MACC
                   *
                   *
                   *
EC16                        END
```

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

FR1BY   EB7C   FR2BY   EB46   LOE235 EB75   LOE239 EB9C
LOE240  EB9D   LOE245  EBB4   MPT48  EA40   PDL10  EBE2
PDL20   EBFC   PDL30   EC06   RABS   EA50   RACOS  EA71
RALOG   EA53   RASC    EACB   RASIN  EA6E   RATN   EA74
RCHR    EAD2   RCOS    EA68   RCR10  EADF   RCURX  EAB8
RCURY   EABE   REXIK   EB1D   REXP   EA56   RFRAC  EA59
RFRE    EB43   RFREQ   EB5C   RGETC  EB79   RHEX   EA83
RINP    EB82   RINT    EB8D   RLE10  EAC7   RLEFT  EAE2
RLEN    EAC4   RLF10   EAEC   RLF20  EAED   RLOG   EA5C
RLOGT   EA5F   RMID    EB0E   RPDL   EBC1   RRIGHT EAFF
RSIN    EA65   RSPC    EA8C   RSQR   EA62   RST20  EA7D
RSTR    EA77   RTAB    EAA2   RTAN   EA6B   RTK55  EA47
RVAL    EB25   RVPT    EBA1   RXMAX  EBA7   RYMAX  EBAE
SIZE    EB51   SPC10   EA8F   SPC20  EA93   SPC30  EA9E
SUEPT   EB29
```

```
                                ORG   :EC16
                        *
                        *
                        *
                        *************************
                        * RUN basicfunction PEEK *
                        *************************
                        *
                        * Returns the contents of a memory location
                        * with an address given as INT argument.
                        *
EC16 CDF8E6             RPEEK   CALL  :E6F8     Get addr in HL
EC19 7E                         MOV   A,M       Get its contents
EC1A C37CEB                     JMP   :EB7C     Move it into MACC
                        *
                        ***********************
                        * RUN basicfunction PI *
                        ***********************
                        *
                        * Returns a value of 3.14159.
                        *
EC1D 21F5D0             RPI     LXI   H,:D0F5   Addr FPT (PI)
EC20 E7                         RST   4         FPT (PI) into MACC
EC21 0C                         DATA  :0C
EC22 C9                         RET
                        *
                        ********************
                        * DATA - (not used) *
                        ********************
                        *
EC23 02                 LOE252  DATA  :02       FPT (PI)
EC24 C9                         DATA  :C9
EC25 0F                         DATA  :0F
EC26 DB                         DATA  :DB
                        *
                        ************************
                        * RUN basicfunction RND *
                        ************************
                        *
                        * Returns a random or pseudo-random number.
                        * If argument > 0: Returns a pseudo-random number
                        *                  in the range 0 <= R < argument.
                        * If argument = 0: Returns a hardware random number
                        *                  0 < R < 1.
                        * If argument < 0: Number replaces the kernel for
                        *                  calculating further pseudo-
                        *                  random numbers.
                        *
EC27 CD8AEC             RRND    CALL  :EC8A     Test if arg is 0
EC2A CA0CE4                     JZ    :E40C     Then hardware random
EC2D 212D01                     LXI   H,:012D   Addr randdom number kernel
EC30 F235EC                     JP    :EC35     If pseudo random number
EC33 E7                         RST   4         Copy MACC to kernel
EC34 0F                         DATA  :0F
EC35 EB                         XCHG
EC36 212901                     LXI   H,:0129   Addr scratch area WORKE
EC39 E7                         RST   4         Copy MACC to WORKE
EC3A 0F                         DATA  :0F
EC3B E5                         PUSH  H         Saveaddr WORKE
EC3C EB                         XCHG
EC3D E5                         PUSH  H         Save addr RNUM
                                MVI   M,:01     Limit range 1-2
```

```
EC40 E7                 RST   4          Copy last nr from RNUM
EC41 0C                 DATA  :0C        into MACC
EC42 1605               MVI   D,:05
EC44 21A8C6     RRD10   LXI   H,:C6A8    Addr RNDA
EC47 E7                 RST   4          Multiply R0*RNDA
EC48 54                 DATA  :54
EC49 21ACC6             LXI   H,:C6AC    Addr RNDB
EC4C E7                 RST   4          Add RNDB to R0*RNDA
EC4D 4E                 DATA  :4E
EC4E 00                 NOP
EC4F 00                 NOP
EC50 00                 NOP
EC51 00                 NOP
EC52 00                 NOP
EC53 21FDD0             LXI   H,:D0FD    Addr AND mask
EC56 E7                 RST   4          IAND: pick out mantissa
EC57 63                 DATA  :63
EC58 21B0C6             LXI   H,:C6B0    Addr OR mask
EC5B E7                 RST   4          IOR: set mantissa top
EC5C 66                 DATA  :66        bit, + range 1-2
EC5D 15                 DCR   D
EC5E C244EC             JNZ   :EC44      Again if D<>0
EC61 E1                 POP   H          Get addr RNUM
EC62 E7                 RST   4          Copy MACC to RNUM
EC63 0F                 DATA  :0F
EC64 21F1D0             LXI   H,:D0F1    Addr FPT (-1)
EC67 E7                 RST   4          Add -1 to MACC (range
EC68 00                 DATA  :00        0-1)
EC69 E1                 POP   H          Get addr WORKE
EC6A E7                 RST   4          Frig range: multiply
EC6B 06                 DATA  :06        MACC*(WORKE)
EC6C C9                 RET
                *
                ***************************
                * part of RUN TALK (CD64) *
                ***************************
                *
                * Entry: Z=1: Code = 0C (delay).
                *        Z=0: Code = 0D (ML call)
                *
EC6D 5E         MPT46   MOV   E,M        )
EC6E 23                 INX   H          ) Wait-time/ML address
EC6F 56                 MOV   D,M        ) in HL
EC70 23                 INX   H          )
EC71 EB                 XCHG             )
EC72 CCCCDA             CZ    :DACC      If to be waited
EC75 C4A9C8             CNZ   :C8A9      Else: Run ML routine
EC78 C367CD             JMP   :CD67      Return: Handle next code
                *
                **************************
                * RUN basicfunction SGN *
                **************************
                *
                * Takes a FPT value and returns:
                *                +1  if value is positive.
                *                 0  if value is zero.
                *                -1  if value is negative.
                *
EC7B CD8AEC     RSGN    CALL  :EC8A      Test if variable is zero
EC7E C8                 RZ               Then ready
EC7F 21F1D0             LXI   H,:D0F1    Addr FPT(-1)
EC82 E7                 RST   4          Copy -1 into MACC
```

```
EC83 0C                   DATA   :0C
EC84 FA89EC               JM     :EC89       Ready if already negative
EC87 E7                   RST    4           Else change sign MACC
EC88 1B                   DATA   :1B         (make MACC  +1)
EC89 C9          LOE257   RET
                 *
                 **********************
                 * TEST A FPT VARIABLE *
                 **********************
                 *
                 * Entry: Variable in MACC.
                 * Exit:  Z=1: Variable is zero.
                 *         Z=0: Other flags set on exponent byte
                 *              of variable.
                 *        ABCDEHL preserved.
                 *
EC8A C5          FTEST    PUSH   B
EC8B D5                   PUSH   D
EC8C F5                   PUSH   PSW
EC8D E7                   RST    4           Copy MACC to reg A,B,C,D
EC8E 15                   DATA   :15
EC8F 5F                   MOV    E,A         Exp byte in E
EC90 B0                   ORA    B           )
EC91 B1                   ORA    C           ) Check if nr is zero
EC92 B2                   ORA    D           )
EC93 CA98EC               JZ     :EC98       Then quit
EC96 7B                   MOV    A,E         Get exp byte
EC97 B7                   ORA    A           Set flags on it
EC98 D1          FTS10    POP    D
EC99 7A                   MOV    A,D
EC9A D1                   POP    D
EC9B C1                   POP    B
EC9C C9                   RET
                 *
                 **************************
                 * RUN basicfunction SCRN *
                 **************************
                 *
EC9D CDF3E5      RSCRN    CALL   :E5F3       Eval given coord
ECA0 C5                   PUSH   B
ECA1 4F                   MOV    C,A         Y-coord in C
ECA2 EF                   RST    5           Ask colour of dot on screen
ECA3 27                   DATA   :27         + size graphics screen
ECA4 C1                   POP    B
ECA5 DA02E6               JC     :E602       Evt run screen error
ECA8 C37CEB               JMP    :EB7C       Contents screen loc in MACC
                 *
                 *
                 *  ==============
                 *** LIST HANDLER ***
                 *  ==============
                 *
                 * This module lists a program from the textbuffer
                 * onto the screen (or into other required direction)
                 *
                 **********************
                 * LIST A PROGRAM LINE *
                 **********************
                 *
                 * Entry: BC: Points to start of textline.
                 * Exit:  BC: Points to start of next line.
                 *            DEHL preserved, AF corrupted.
```