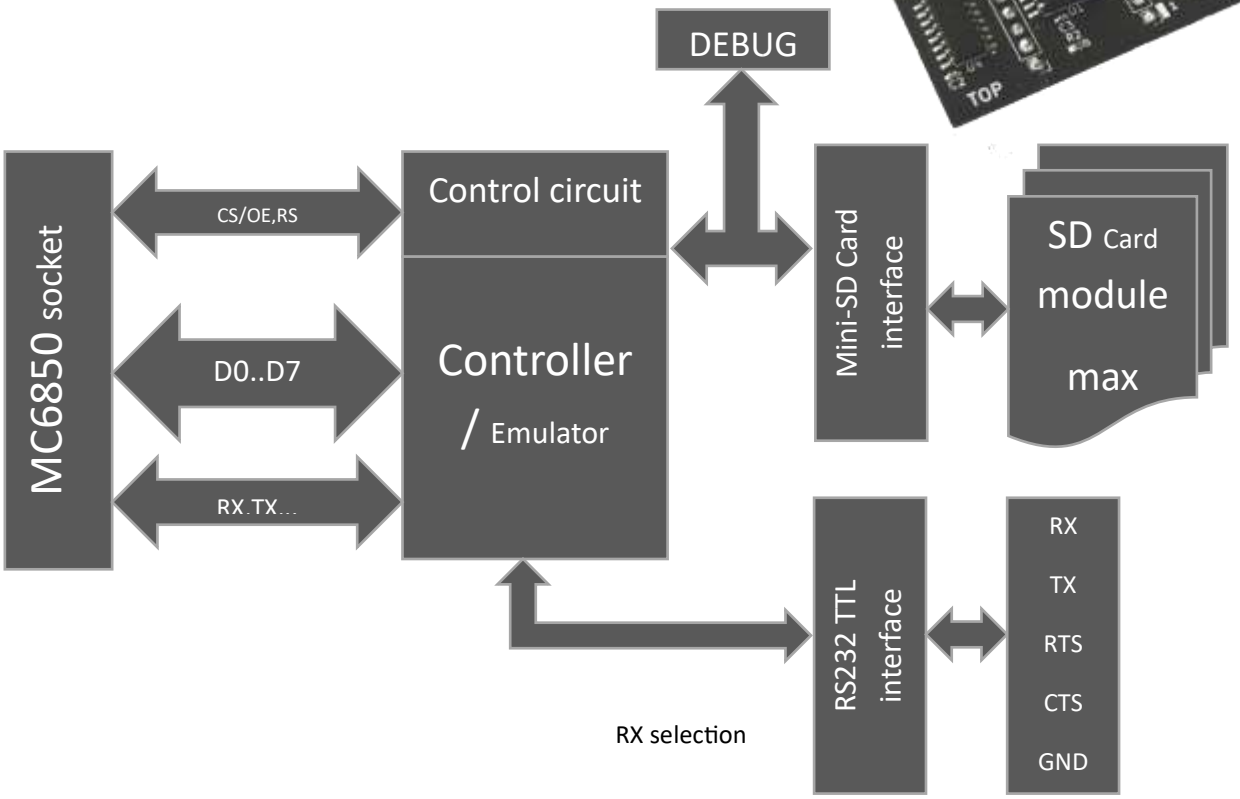
ACIA Emulation -
SD CARD REFERENCE (V1.39 4-2026)

Functional diagram:



How to access extended ACIA Control register:

1- Master reset with $03 to ACIA control register (normally not needed (OPTIONAL))
2- Set ACIA functions (Baudrate, bits,RTS,CTS…) to ACIA control register
   If ACIA special function CR5, CR6 = 11 is set, a byte has to follow to specify the extended function (see tables below)
3- The extended function consists of: Extended command byte (low nibble), Extended function (high nibble)

Example: byte $07 will call extended function 0 and extended command 7 (`= Set FIFO to 8, no return value.

Extended command:

Note: "no return" value means the ACIA data port will not return a status or data byte.
In case a byte/word/string is returned, the program "must" read these or the ACIA will perform a timeout reset, which may lock up the user program.

If a status byte is returned, a "1" means OK, a "0" failed or error.

In addition, the emulated ACIA uses an internal timeout of about 2 seconds. Means if a command call is not processed, waits or is interrupted the ACIA will perform a timeout reset (writing a file for example should not be interrupted longer than 2 seconds).

Most of the extended commands can be interrupted by a reset (0x03) to the ACIA control register (f.eg. Saving a file, etc.). I such a case, no command return value will be presented. The function called will be aborted but processed data will remain.

0 Set/Clear special functions

0: do nothing
1: return version info return, return 1 data byte (00..),
2: reset all special functions, no return value
3: select RS232 internal port, no return (uses system RX/TX lines)
4: select RS232 external port, no return (uses onboard pin header RX/TX lines)
5: Set FIFO to 2 , no return (RTS/CTS handshake by user)
6: Set FIFO to 4 , no return (RTS/CTS handshake by user)
7: Set FIFO to 8 , no return (RTS/CTS handshake by user)
8: Set FIFO to 16 , no return (provides automatic RTS/CTS handshake)
9: read 1 byte from EEPROM, send startL, startH, return 1 data byte (Adr. 0 to 1023)
10: write 1 bytes to EEPROM, send startL, startH, data, return 1 status byte (control)
11: activate(1)/deactivate(0) setup on Boot from EEPROM parameter table, no return
12: write setup to EEPROM, return 1 status byte
13: activate(1)/deactivate(0) RTS for incoming serial data, no return
14: undefined, no return
15: Get last SD Error, returns 1 data byte

1 Execute Loader Code in binary with leading length and start address, no return value

0: Pre-Loader (0240) to load BOOT.SYS from SD, return lod file for OSI monitor loader
1…15: more code may be placed into ATMEGA in future releases

2 Select baud rate, no return value (SW=software defined, HW=by OSI typical TX clocks
Note: HW are the only baud rates supported by emulation and TX clock analysis on Startup.

TX clock cannot be used directly for the emulated UART due to hardware restriction
to only TX clock divider by 1 (Syncron mode of UART)

0: restore from SW to HW serial baud rate from initial reading of TX clock
1: 150 only SW
2: 300 +HW
3: 600 +HW
4: 1200 +HW
5: 2400 +HW
6: 4800 +HW
7: 9600 +HW
8: 14400 only SW
9: 19200 +HW
10: 38400 +HW
11: 57600 only SW -> transmit works, rest not reliable
12: 76800 only SW -> nonstandard, not tested
13: 115200 only SW -> transmit works, rest not reliable
14: 125000 only SW -> not tested
15: 31250 only SW ->> MIDI IN/OUT, +1.58% faster compared to MIDI

3 SD card functions
Note: Only a single file can be opened for R/W. If SD operation holds for more than 2 sec, a soft reset will be initiated to restart the ACIA emulation. So, writing to a file in intervals separated by more than 2 sec will not work!

If an operation was successful, a 0x01 is returned, elsewhere a 0x00
Definition: Path text can be max 31 characters long, ending with 0x00. A Path provided should not have a "/" slash at the end. A Path looks like "config/files". File names can be max 23

characters long, ending with 0x00. Upper- or lower-case charters are handled equally. The file extension can be omitted or of any length desired. The first letter in a file extension is used in some cases to identify the type of file. No file extension is treated as an executable binary file. Extensions recognize are: dot Bxx for Basic text files, dot Pxx for binary executables files with the loading/tart address at the beginning, dot Lxx , dot 6xx and dot Hxx are OSI machine code files. Files larger than 64k can be read but not written (max 64k). In the following functions, a file read from the disk by the user must be interpreted and handled accordingly. The BASIC SDLOAD command extension may autoload the file according its extension or to the target address defined by the user.

Typical random read speed (with a 32GB card, I'm using) is about 12.5 Kbytes/sec compared to the 10 Kbyte/sec for the OSI floppy disk system.

0: start and check SD card (will close all files !), returns 1 if available, 0 if not
1: close and eject SD card, no return (card needs to be re-inserted !)
2: get name of entry number (0..255) as string of selected path, return name zero term.
3: set filename, name ends with byte zero (max 24 characters incl. zero), no return
4: check if file exist, returns 1 status byte
5: set path, ends with byte zero (max 32 characters), no return
6: return directory list, name plus zero. Ends with zero at end of directory.
The directory output may be interrupted by sending a break $03 to the control port.
Returns no value after a break command. Break must be done at the beginning of the
next directory string output. (note: older ACIA firmware may not work correctly)
7: get free disk space in MB, returns 2 data bytes 16bit
8: get file size in bytes, returns 2 da bytes 16bit, returns 65535, if file larger than
65535 bytes (in BASIC -1) or 0 if file does not exist.
9: read file, read 2 bytes file size, returns zero word on error or no file present,
followed by binary data. Aborted by writing $03 to ACIA control port if needed,
no return.
10: create folder in selected path (folder name in filename) returns 1 status byte
11: delete folder in selected path (folder name in filename), returns 1 status byte
12: delete file (defined before with path), returns 1 status byte
13: rename file or folder, send new filename, name ends with byte zero (max 24
characters), returns 1 status byte. Use function 0x33 to specify file to rename.
14: write file, send 2 bytes file size (max 64k), return 1 status byte,
if file was allocated. Next send binary data, may be aborted by writing $03 to ACIA
control port. **returns 1 status byte when done**, nothing on break.
15: copy file, send new path followed by new filename, if path is just zero,
the same path is used as the source. Filename and path must end with byte zero
(max 24 characters for filename and 32 characters for path),
returns 1 status byte. Use function 0x33 and 0x35 before, to specify source file.

Note: before rename, do step 3,5 and 4, to define file to be renamed!

4: IRQ enable/disable timer, does not set IRQ flag in Status register

NOTE: activates even if IRQ enable is not set, IRQ signal resets after 10µsec by itself

Note: CTS can be used as an interrupt input and disables normal CTS behavior

0: disable timer IRQ
1: IRQ on CTS goes low. IRQ resets after 14µsec by itself
2: IRQ 128 µsec
3: IRQ 256 µsec
4: IRQ 512 µsec
5: IRQ 1.024 msec
6: IRQ 2.048 msec
7: IRQ 4.069msec
8: IRQ 10 msec
9: IRQ 16.67 msec (60 Hz)
10: IRQ 20 msec (50 Hz)

11: IRQ 32.78 msec (max)
12: read OSCCAL factory calibration value, returns 1 byte
13: read actual OSCCAL, returns 1 byte
14: write OSCCAL, changing will lead to non-function TX/RX baud rates !!
15: reset OSCCAL to calibration value

Note: to use IRQ on OSI bards, you have to connect pin 7 (Socket of ACIA) with the CPU IRQ line Pin 4. On other systems, this might be already connected.

OSCAL allows variation of system clock in a range of +/- 25%, if you need to adjust to special external baud rates. The calibration value should work for standard baud rates but can become helpful on rates > 38400 to fine tune the interface timing.

5: Virtual disk commands

The virtual disk is an image of a data disk in a single file. The virtual disk comes with a header to identify that it is a valid image and disk format structure.

The **standard sectored disk** access method is done by addressing a 16bit sector address and read or write max 256 sectors from that position onwards.

The **alternative track disk** access method is done by addressing an 8bit track number, addressing a read or write offset to the start of the track. Then read or write data per track of up to 15bit or 32k without looking for sector limits.

0: Close actual virtual disk, no return (use only for virtual disks)

1: Set actual Virtual disk filename by number (0..3) active, no return.
Default names are VDISK_A.DSK...VDISK_D.DSK. Using define filename (function 0x33) afterwards is possible to select any other file name. Use create folder (function 0x3A) before virtual disks are created with function 0x53, if a group of virtual disks should be placed together.

2: define actual virtual disk header (number of sectors per side, sector size, sectors per track per side), 15 bit value, 8 bit value in bits (B7:sides + B6..B5:4 sector size types + B4..B0:32 sectors per track per side),(write protection is always on).
Sector size types are:0=128,1=256,2=512,3=1024 bytes per sector at 1..31 sectors per track per side, no return. No data is written onto the file yet; it will be written only by create virtual disk (function 0x53)

3: Create empty virtual disk on SD, no input required. (header 32 bytes)(VIRTUAL ID “ACIADSK”, Version, in 8 bytes), (number of sectors, sides, sector size, sectors per track) , 19 times 00). Sectors follow in sequence, second side follows first side sectors (double sides disk have twice the sectors), returns error status

4: Open actual virtual disk with name that was given in function 0x51 or 0x33 before, returns error status.
Before selecting any other file, use close virtual disk (function 0x50) and before using any other disk R/W function of the 03x group, because they will close all files after the call.

5: Get actual virtual disk layout (number of sectors per side, sides, sector size, sectors per track per side, number of tracks) returns error status, followed by 15bit value + 8bit value + 8bit value, if no error occurred.

6: Set or Clear actual virtual disk write protection (only during session) 8 bit value, 1=set, 0=clear. elsewhere virtual disks are always write protected by default. No return.

7: Set actual virtual disk position (sector number) 16 bit value (consider single or double side layout), returns error status. see Note Sector logic.

8: Read sectors (number) 8 bit value (max 256 sectors in continuous sectors only form) returns error status. If ready, a one is returned and read transfer of sector data can follow, else termination. No return.

9: Write sectors (number) 8 bit value (max 256 sectors in continuous sectors only form) returns error status. If ready, a one is returned and write transfer of sector data can

follow, else termination. May be aborted by writing $03 to ACIA control port.
returns 1 status byte when done, nothing on break.

10: Scan if SD card is inserted without closing files, like in function 0x30).
returns 1 if SD card is inserted, 0 if not

11: Alternative virtual disk method, Goto track number (8 bit value, 0...255 tracks)
and sets track position to the start of track, returns error status.
Max 256 tracks on double or single side disk, Side select not needed. See Note

12: Alternative virtual disk method, advance read/write track position 15 bit value in bytes (32k range), returns error status. No return

13: Alternative virtual disk reading, read number of bytes starting from actual track position (15 bit value (32k range), returns error status.
If no error, read transfer of byte data can follow, else termination.

14: Alternative virtual disk writing, write data bytes starting from actual track position (15 bit value (32k range), returns error status.
If no error, write transfer of byte data can follow, else termination.
May be aborted by writing $03 to ACIA control port. **returns 1 status byte when done**, nothing on break.

15: NDY

Note to 7 - Direct sectored virtual disk:
max 2 sides
number of sectors 65536 to be addressed om a disk (no track or side info needed)
max 1024 bytes per sector
max capacity 65 Mbytes

Note to 11 (alternative tracked virtual disk):
Range limits are not checked, if they cross the track end limit. In such a case, read/write will occur into the next track.
An error is indicated by a return byte of zero, no error by a one

Sector logic:
- on single or double side disk, all virtual sectors are in sequence, first side track data followed by second side tack data

- on a double-sided disk, you have twice the sectors available. No side select needed.

On each track, a maximum of (31sec*1024bytes) 31k bytes can be set, for all sector formats and single- or double-sided disks.

This results in a max alternative virtual disk capacity

max 2 sides
max 256 tracks
max 31 sectors per track or 2x31 sectors on double sided (no side info needed)
max 1024 bytes per sector or 31k or 62k per track (single or double sided)
max capacity 15.5 Mbytes

Example: Double sided disk, 40 track, 8 sectors/track. This would allow you to read 16 sectors from each track (using both sides) without selecting the side during a read/write sequence.

Single sided disk, 40 track, 8 sectors/track, (@ side 0). This would allow you to read 8 sectors from each track

****

Examples:
Single side 5,25in   (OSI for R/W in alternative format)
number of sectors 360
min 1 side
min 9 sectors per track
typ 1 256 bytes per sector (0x28) calc type (40 tracks)

capacity 90 Kbyte raw (80 Kbytes data)

```
        Single side 8in
        number of sectors 1694
        min 1 side
        typ 16 sectors per track
        typ 0 128 bytes per sector ((0x10) calc type (77 tracks)

        capacity 212 Kbytes
```

Program Example for checking SD card presence and free memory on card:

```
ACIA_S          = $F000         ; SERIAL ACIA Control Port
ACIA_D          = $F001         ; SERIAL ACIA Data Port
SDEXIST         = $xxxx         ; PLACE IN MEMORY

SAMPLE: JSR     SDVAL           ; Do SD Validation on startup

        LDA     SDEXIST
        BEQ     ERROR           ; no SD present

        LDA     #$71            ; Extended mode
        STA     ACIA_S
        LDA     #$37            ; Check free size on SD
        JSR     ACIA_WRITE      ; Get file size

        JSR     ACIA_READ
        TAY                     ; Low to Y
        JSR     ACIA_READ       ; High to A
TEND:   RTS                     ; Returns size in A,Y

SDVAL:
        LDA     ACIA_D          ; Clean up any pending byte
        LDA     #$71            ; enter Extended mode
        STA     ACIA_S

        LDA     #$30
        JSR     ACIA_WRITE      ; function Check if SD present

        JSR     ACIA_READ       ; read return value
        STA     SDEXIST
REND:   RTS

ACIA_READ:
        LDA     ACIA_S
        LSR
        BCC     ACIA_READ
        LDA     ACIA_D
        RTS

ACIA_WRITE:
        PHA
WR1:    LDA     ACIA_S
        AND     #$02
        BEQ     WR1
        PLA
        STA     ACIA_D
        RTS
```

```
; ************************************************************************
;                     BASIC Plus and SD Commands V1.8 for machines above 8k RAM
;                     File to be named BASIC without extension and also BOOT.SYS
;                     SAVE CODE in binary format as BOOT.SYS
;                     New BASIC Syntax, on 8k machines, some BASIC commands are not available
; ************************************************************************

                              ; Size 2k bytes
START_ADR      = $1B00        ; 8k version (reduced syntax)
;START_ADR     = $3800        ; 16k version
;START_ADR     = $5800        ; 24k version
;START_ADR     = $7800        ; 32k version
;START_ADR     = $9800        ; 40k version

BASICPLUS = START_ADR + $0100
FW_NEW         = 1            ; still old firmware


ACIA_S         = $F000        ; SERIAL ACIA Control Port
ACIA_D         = $F001        ; SERIAL ACIA Data Port
BASIC_16_FLOAT        = $AFC1        ; Convert Fixed Point to Floating Point
BASIC_OUT      = $A8E5        ; BASIC Character output
BASIC_CRLF     = $A86C        ; BASIC LINE Return
BASIC_FCERR    = $AE88        ; BASIC FC ERROR
BASIC_LPT      = $00BC        ; Basic Line Processing Vector in Zero-Page
BAS_GET_CR     = $00C2        ; BASIC GET CURREMT CHAR FROM BASIC LINE
BASIC_ALPHA    = $AD81        ; BASIC CHECK CHARACTER FOR ALPHA
BASIC_NEXT     = $A70F        ; BASIC: scan for next basic statement
BASIC_EVAL     = $AAC1        ; BASIC EVALUATE EXPRESSION
BASIC_RELEASE  = $B2B3        ; ROM BASIC, Release string
BASIC_8BIT     = $B3AE        ; BASIC ROM: EVALUATE 8BIT EXPRESSION and convert to byte
BASIC_B2B6     = $B2B6        ; BASIC Free Temp String
BASIC_CLEAR    = $A47A        ; BASIC CLEAR
BASIC_SETUP    = $A5FB        ; BASIC RUN or CLEAR SETUP -1
BASIC_RUN      = $A5FC        ; BASIC RUN
BASIC_A477     = $A477        ; INITIALIZE, KIND OF BASIC CLEAR
BASIC_POKE_PARM = $B3FC              ; BASIC Evaluate like POKE (adress, value)
BASIC_FINDL    = $A432        ; BASIC SEARCH BASIC LINE NUMBER, ADR in AA-AB
BASIC_G16B     = $AAAD        ; BASIC GET 16BIT ARG FROM BASIC LINE
BASIC_RSTOR    = $A621        ; BASIC FINALIZE RESTORE
BASIC_B408     = $B408        ; BASIC Convert FLOAT to INT, Result in 11-12
BASIC_STR      = $B0AE        ; scan and set up string
DISK_BOOT      = $FC00        ; Entry to disk boot in ROM
Screen         = $D000
RESET          = $FFFC
BOOT           = $FF00

HORZ_SIZE      = $FFE1        ; <32 or >32 will indicate horizontal screen resolution (32 or
64)
VERT_SIZE      = $FFE2        ; 0 will indicate 2k, 1 is 4k Screen memory size

BASFLAG        = $0203
LOD            = $FE2A
LODCEG         = $F96B
LODFLAG        = $FB
TMP            = $0B          ; Program start adress. Run with X=USR(X)
PTMP           = $FC          ; Temp counter or flag

ACIA_READ = ACIA_R
ACIA_WRITE= ACIA_W
CLS       = SCREEN_CLS


       ; Loader start address ($02CA)
                              ; 14 bytes header
       ; JMP   LOADER         ; $02D8-14
```

```
        ;.DB    $0              ; 3 TYPE 0=BAS(TXT) 1=LOD, 2=PRG(BIN) >2= Others
        ;.DB    $0              ; 4 ACTION 0=LOAD >0=SAVE. Bit 8 set= ignore Path and Filename
        ;.DB    $0, $0          ; 5,6 START for save
        ;.DB    $0, $0          ; 7,8 END for save
        ;.DB    <Path, >Path    ; 9,10, string must end with "00"
        ;.DB    <File, >File    ; 11,12, string must end with "00"
        ;.DB    $0              ; 13 AUTOSTART 0=not >0=yes for BASIC only

        ; PRG Program or BASIC_LOADER at $02D8 ** NEW version 2025 **
        ; BASHEADER PRG Basic starting at $02D8 (new version V3)
        ;
$A2,$03,$BD,$FB,$02,$95,$79,$CA,$10,$F8,$A9,$A5,$48,$A9,$FB,$48,$AE,$FF,$02,$D0,$0B,$A9,$FD,$8
5,$C3,$A9,$BF,$85,$C4,$4C,$7A,$A4,$4C,$77,$A4
        ; followed by
        ; BASIC START ($02FB)
        ; BASIC END ($02FD)
        ; BASIC Autorun 1/0 ($02FF)


BASIC_LOADER    = $02D8         ; NEW version 2025
BASIC_HEADER    = $02FB         ; BASIC START ($02FB)
                                ; BASIC END ($02FD)
                                ; Autorun 1/0 ($02FF)


        .ORG    START_ADR-2

        .DB     <START_ADR
        .DB     >START_ADR

        .ORG    START_ADR       ; In free Memory

        JMP     BASICPLUS


        .INCLUDE "Internal_Loader_V1.asm"



; ********************+*************************************
; ***************** MENU @ BASICPLUS position **************

        .ORG    BASICPLUS

BASIC:  LDA     #$4C            ; Prepare Jump address
        STA     LOADER
        STA     BASIC_LPT       ; Copy EX_BASIC vector to BASIC_LPT $BC

        LDA     #<EXLOAD
        STA     LOADER+1
        LDA     #>EXLOAD
        STA     LOADER+2
        LDY     #11
        LDA     #0
IN:     STA     LOADER+2,Y      ; Clear header
        DEY
        BNE     IN

        lda     #<EX_BASIC
        sta     BASIC_LPT+1
        lda     #>EX_BASIC
        sta     BASIC_LPT+2

        JSR     CLS
        STA     $0201           ; last CEGMON char is space
```

```
        LDA     #<START_ADR     ; Set Top of Ram
        STA     $81
        STA     $85
        STA     TMP
        LDA     #>START_ADR
        STA     $82
        STA     $86
        STA     TMP+1

        LDA     0               ; Needs to take place here !!!
        CMP     #$4C            ; Check for active BASIC
        BEQ     CHECKCARD       ; No BASIC COLD START
        JMP     BOOT


CHECKCARD:

        LDA     ACIA_D          ; Clean up any pending byte
        JSR     VALC            ; Do SDVAL on startup

        lda     #>BASIC_SETUP
        pha
        lda     #<BASIC_SETUP
        pha

        lda     #$FD    ; Pointer to 3x "00" in BASIC ROMS
        sta     $C3
        lda     #$BF
        sta     $C4
        jmp     BASIC_CLEAR ; Basic CLEAR, goto to BASIC ROM without OR-Error


; ****************************************************************************
;
; All SD command start with SD ($53,$44)
; followed by BASIC Token number
;
; (99)SDLIST                                                List current directory
; (93)SDLOAD FILENAME,[ADDRESS]                                     Loads BASIC or Binary file

; (94)VAL=SDSAVE FILENAME,[START ADDRESS, END ADDRESS]              Saves BASIC or binary file
; (BC)VAL=SDLEN  FILENAME                                   Returns length of file
; (8C)SDGOSUB PATH                                          Select Directory
; (B5)VAL=SDLOG                                                     Returns last error
; (9A)VAL=SDCLEAR FILENAME                                  Removes a file
; (B1)VAL=SDFRE                                                     Returns free space on SD
card
; (95)SDDEF 0/1                                                     AutoStart for BASIC files
; (BE)VAL=SDVAL                                                     Validate SD / Close file
; (89)SDRUN                                                 Runs the "MENU" program
; (91)SDNULL PATH (not on 8k version)                               Remove directory in current
path
; (85)SDDIM PATH (not on 8k version)                        Create directory name in current
path
; (BF)VAL=SDASC FILENAME_OLD, FILENAME_NEW                  Rename File (not on 8k version)
; (C0)SRING=SDCHR$(NUMBER) (only new ACIA firmware)         Get directory entry at position
;
; SDVAL should be used to check SD card presence or restart the SD card
;
; Programs are always saved in binary format, even with .BAS extension (for now in this
revision)
; SDDEF 1 will enable autostart. Do this before save. SDDEF is default 0
;
; Without ADDRESS, the current BASIC program will be taken and saved PRG encoded (binary)
; With optional [ADDRESS] field, a machine code PRG file will be created
```

```
; Using ADDRESS on SDLOAD, the data of the file will be loaded to the ADDRESS instead of its
original location
;
; BASIC Autorun can be set by storing a one to memory location 767 (same as SDDEF 0 or 1)
; SDAL will reset Autorun flag. After SDLOAD, flag is not changed
;
; SDRUN will  run the MENU program located in the root directory.


        .DB $00                 ; End of Table

        .IF FW_NEW              ; ADD VAL syntax ?
        .DW CHR-1               ; CHR$ COMMAND +
        .DB $C0
        .ENDIF

        .IF START_ADR>$2000     ; Check for available memory

        .DW ASC-1
        .DB $BF                 ; SDASC COMNAND +

        .DW DIM-1
        .DB $85                 ; SDDIM COMNAND +

        .DW NULL-1
        .DB $91                 ; SDNULL COMNAND +

        .ENDIF


        .DW RUN-1
        .DB $89                 ; SDRUN COMNAND +

        .DW VAL-1
        .DB $BE                 ; SDVAL COMNAND +

        .DW DEF-1
        .DB $95                 ; SDDEF COMNAND +

        .DW LIST-1
        .db $99                 ; SDLIST COMNAND +

        .dw LOG-1
        .db $B5                 ; SDLOG COMNAND +

        .dw FRE-1
        .db $B1                 ; SDFRE COMNAND +

        .dw CLEAR-1
        .db $9A                 ; SDCLEAR COMNAND +

        .dw SAVE-1
        .db $94                 ; SDSAVE COMNAND +

        .dw GOSUB-1
        .db $8C                 ; SDGOSUB COMNAND +

        .dw LEN-1
        .db $BC                 ; SDLEN COMMAND +

        .dw LOAD-1
        .db $93                 ; SDLOAD COMNAND +


TOKEN:
```

```
EX_BASIC:                   ; BASIC ENTRY POINT
       pla
       cmp    #$30          ; Check if Basic calls from Token analysis
       pha
       bne    Leaved        ; leave
       stx    PTMP          ; Store X in Basic Extension Linkage
       ldy    #$01
       lda    ($C3),y
       jsr    BASIC_ALPHA   ; Check if points to token
       bcc    Start         ; OK, start analysis
Leave:
       ldx    PTMP          ; restore X and leave
Leaved:
       inc    $C3
       bne    Lq
       inc    $C4
Lq:    jmp    BAS_GET_CR    ; Return to Adress $00C2

; ************************************************
;

EX_CMD: PHA                 ; Call extended command in A
       LDA    #$71
       STA    ACIA_S
       PLA
       JMP    ACIA_WRITE

Start:                      ; Start analysis
       lda    $C3
       bne    S1
       dec    $C4           ; one byte back
S1:
       dec    $C3
       dey
       ldx    #$FF

       lda    ($C3),y
       cmp    #$53          ; "S" ?
       bne    TokenE
       iny
       lda    ($C3),y
       cmp    #$44          ; "D" ?
       bne    TokenE
       iny
SR:
       lda    TOKEN-256,x   ; BASIC TOKEN CHECK
       beq    TokenE
       cmp    ($C3),y
       beq    Found

       dex                 ; next Token
       dex
       dex
       bne    SR
Found:
       jsr    BASIC_NEXT    ; BASIC: scan for next basic statement
       pla
       pla
       pla
       pla
       lda    TOKEN-257,x   ; Vector to DOSSUP Basic token code
       pha
       lda    TOKEN-258,x
       pha
       jmp    Leave
```

```
TokenE:                         ; Code not found
        ldy     #$01
        jsr     BASIC_NEXT      ; BASIC: scan for next basic statement
        jmp     Leave           ; and leave


; ****************************************************************************

RUN:
        LDA     ACIA_D          ; Clean up any pending byte

        LDA     #2              ; User LOADER modul to run MENU.prg
        STA     LOADER+3
        STA     LOADER+13       ; Autostart on
        LDA     #0              ; LOAD and with path back to root
        STA     LOADER+4
        LDA     #>PATH
        STA     LOADER+10
        LDA     #<PATH
        STA     LOADER+9
        LDA     #>MENU
        STA     LOADER+12
        LDA     #<MENU
        STA     LOADER+11
        JMP     LOADER          ; RTS in program will return to BASIC

; ****************************************************************************


        .IF START_ADR>$2000     ; Check for available memory

DIM:                            ; Create Subdirectory
        JSR     FILENAME
        LDA     #$3A
        JSR     EX_CMD
        BNE     LEN8R           ; always jump

        .ENDIF

; ****************************************************************************

        .IF START_ADR>$2000     ; Check for available memory

NULL:
        JSR     FILENAME
        LDA     #$3B
        JSR     EX_CMD
        BNE     LEN8R           ; always jump

        .ENDIF


; ****************************************************************************

CLEAR:
        JSR     FILENAME
        LDA     #$3C
        JSR     EX_CMD
        BNE     LEN8R           ; always jump


; ****************************************************************************

VAL:
        JSR     VALC
```

```
        CLC
        BCC     LEN8


VALC:
        LDA     ACIA_D          ; Clean up any pending byte
        LDA     #0
        STA     BASIC_HEADER+4 ; Preset Autorun to OFF for BASIC files
        LDA     #$30            ; Check if SD present
        JSR     EX_CMD
        JSR     ACIA_READ
        STA     SDEXIST
        RTS

; ****************************************************************************

FRE:
        LDA     SDEXIST
        BEQ     LEN8            ; no SD, return zero

        LDA     #$37
        JSR     EX_CMD
        BNE     LEN16           ; always jump

; ****************************************************************************

LEN:
        JSR     FILENAME
        LDA     #$38            ; Get File size
        JSR     EX_CMD

LEN16:
        JSR     ACIA_READ
        TAY                     ; Low to Y
        JSR     ACIA_READ       ; High to A
        JMP     BASIC_16_FLOAT ; Indirect jump to BASIC routine


FCERR:
        JMP     BASIC_FCERR     ; STOP WITH FC ERROR

; ****************************************************************************

LOG:
        LDA     #$0F            ; Get Last SD Error
        JSR     EX_CMD

LEN8R:  JSR     ACIA_READ
LEN8:   TAY                     ; 8 Bit value to Y
        LDA     #$00
        JMP     BASIC_16_FLOAT ; Indirect jump to BASIC routine

; ****************************************************************************

DEF:
        BEQ     FCERR
        JSR     BASIC_8BIT      ; BASIC ROM: EVALUATE 8BIT EXPRESSION and convert to byte
        STX     BASIC_HEADER+4 ; Preset Autorun to OFF for BASIC files
        RTS

; ********************** not yet supported by ACIA firmware ****************

        .IF FW_NEW
CHR:
        BEQ     FCERR
        JSR     BASIC_8BIT      ; BASIC ROM: EVALUATE 8BIT EXPRESSION and convert to byte
```

```
        LDA     #$32            ; Get String of directory entry
        JSR     EX_CMD
        TXA
        JSR     ACIA_WRITE      ; of X

        LDA     #$00
        STA     $9F             ; High
        LDA     #$13
        STA     $9E             ; Low in keyboard buffer

VAL_N:  JSR     ACIA_READ       ; First zero ?
        STA     ($9E),Y
        BEQ     VAL_C
        INY
        BNE     VAL_N

VAL_CL: STA     ($9E),Y
        INY
VAL_C   CPY     #$5A            ; We need to clean up rest of keyboard buffer with 00
        BCS     VAL_E
        BNE     VAL_CL

VAL_E:  LDA     #$13            ; Start
        LDY     #$00
        JMP     BASIC_STR

        .ENDIF

; ********************************************************************************
        .IF START_ADR>$2000     ; Check for available memory

ASC:
        JSR     FILENAME
        JSR     BAS_GET_CR      ; Next BASIC value
        CMP     #','            ; More Parameters ?
        BNE     FCERR           ; Test for "," (New file name, if more parameters)
        JSR     BASIC_LPT       ; BASIC: Advance to next parameter

        jsr     GETSTR          ; Get string pointers or Error
        cmp     #00             ; Filename empty ?
        beq     FCERR           ; return with 0

        LDA     #$3D            ; RENAME
        JSR     EX_CMD

        LDY     #0
NEW1:   LDA     ($9E),y
        CPY     PTMP            ; String length
        BNE     NEW2
        LDA     #0
NEW2:   INY
        JSR     ACIA_WRITE
        BNE     NEW1
        BEQ     LEN8R


        .ENDIF

; ********************************************************************************

LIST:
        JSR     VALC
        BEQ     LEN8            ; Back on no SD

        LDA     #$36            ; Get Directory
        JSR     EX_CMD
```

```
LC:     JSR     BASIC_CRLF      ; BASIC: Do PRINT CR,LF

        JSR     ACIA_READ       ; Any more DIR lines
        BEQ     LE              ; End of directory ?

        PHA
        LDA     $DF00           ; Break listing on Shift Right
        AND     #$02
        BNE     LI1
        PLA
        LDA     #$03            ; Shift key ends DIR Process  (needs to be in active data
transfer !!)
        STA     ACIA_S
        RTS

LI1:    PLA
LI2:    CMP     #$5B
        BCS     LIS
        ORA     #$20            ; only small letters for better reading
LIS:    JSR     BASIC_OUT       ; Send char to screen via Basic
        JSR     ACIA_READ       ; First zero ?
        BEQ     LC
        BNE     LI2
LE:     RTS



; ***************************************************************************


LOAD:
        JSR     GETPAR1         ; Set filename if exist and get first parameter
        BNE     LOADFILE        ; Test for "," (Load file, if no more parameters)

        STA     $9F             ; High
        STY     $9E             ; Low
        JMP     LDF0            ; we have forced address, and no autorun, expect original start
address included in PRG/DATA file


LOADFILE:                       ; use of internal loader
        LDY     #0              ; Check extension
        STY     LOADER+13       ; Default no autorun
EXT0:   LDA     ($9E),Y
        BEQ     EXDEF           ; nothing found, take it as PRG/DATA (default)
        CMP     #'.'
        BEQ     EXT1            ; dot found ?
        INY
        CPY     PTMP            ; String length reached ?
        BNE     EXT0            ; next character
        BEQ     EXDEF           ; no dot found, PRG (2) default

EXT1:   INY
        LDA     ($9E),Y

        LDY     #0              ; filetype in Y
        AND     #$DF            ; capital letter
        CMP     #'B'            ; BAS
        BEQ     EXT2            ; Type 0
        INY
        CMP     #'L'            ; LOD
        BEQ     EXT2
        CMP     #'6'            ; 65V
        BEQ     EXT2
        CMP     #'H'            ; HEX
        BEQ     EXT2            ; Type 1
```

```
        INY
        CMP     #'P'            ; PRG or Other TYPE ?
        BNE     EXT2            ; Something else

EXDEF:  LDY     #$02
        STY     LOADER+13       ; And set to autorun
EXT2:   STY     LOADER+3        ; Setup and execute pogram
        CPY     #$02
        BCS     EXT3            ; Clear screen for Basic text or Mcode
        JSR     CLS
EXT3:   JSR     COPY_FN         ; Copy string with "00" at the end
        LDA     #$80            ; LOAD and ignore path
        STA     LOADER+4
        LDA     #<PATH          ; Path address pointer to zero
        STA     LOADER+9
        LDA     #>PATH
        STA     LOADER+10
        LDA     #<FN
        STA     LOADER+11
        LDA     #>FN
        STA     LOADER+12
        JMP     LOADER          ; Back to internal LOADER


LDF0:   LDA     #$39            ; Get File
        JSR     EX_CMD

LDBIN:  JSR     ACIA_READ       ; Get file lengh
        TAX
        JSR     ACIA_READ
        TAY
        BNE     LDF1
        TXA
        BEQ     LDFE            ; File not found or empty, return

LDF1:   TXA                     ; substract 2 of file length (original start address)
        BNE     LDF21
        DEY
LDF21:  DEX
        BNE     LDF22
        DEY
LDF22:  DEX

        JSR     ACIA_READ       ; file Start adress (ignored)
        JSR     ACIA_READ

LDF42:  LDA     $9E             ; remember Start address
        STA     $9B
        LDA     $9F
        STA     $9C

        STY     PTMP            ; Temp

        LDY     #0
LDF5:   JSR     ACIA_READ
        STA     ($9E),y         ; Store
        INY
        BNE     LDF6
        INC     $9F
        DEC     PTMP
LDF6:   DEX
        BNE     LDF5
        LDA     PTMP
        BNE     LDF5
```

```
LDF9:   LDX     #1              ; Load sub returns 1 if completed
LDFE:   RTS


; ***************************************************************************

GETPAR1:                        ; Set filename if exist and get first parameter
        jsr     FILENAME
        JSR     BAS_GET_CR      ; Next BASIC value
        cmp     #','            ; More Parameters ?
        bne     NOMORE          ; Test for "," (Save Basic, if no more parameters)

GETPAR0:                        ; only get next parameter
        jsr     BASIC_LPT       ; BASIC: Advance to next parameter

        jsr     BASIC_G16B      ; ROM BASIC: EVALUATE 16BIT EXPRESSION, MAKE SURE IT IS NUMERIC
        jsr     BASIC_B408      ; CONVERT TO A 16-BIT VALUE, Result in (A H) and (Y L)
        ldx     #0
NOMORE: rts                     ; return 0 if we have no next parameter



; ***************************************************************************

SAVE:
        jsr     GETPAR1         ; Set filename if exist and get first parameter
        bne     SAVBASIC        ; Test for "," (Save Basic, if no more parameters)
        sta     $9F             ; High
        sty     $9E             ; Low

        jsr     BAS_GET_CR      ; Next BASIC value
        cmp     #','            ; More Parameters ?
        beq     SAV22           ; Test for "," (Save Basic, if no more parameters)
        jmp     $AE88           ; FC-Error

SAV22:  jsr     GETPAR0         ; Get next parameter (end address)
        sta     $9C             ; High
        sty     $9B             ; Low
        jmp     WRITE

SAVBASIC:
        lda     #>BASIC_LOADER
        sta     $9F
        lda     #<BASIC_LOADER ; BASIC PRG pre loader
        sta     $9E

        ldy     #38
        lda     $7C
        sta     $9C
        sta     ($9E),y
        dey
        lda     $7B
        sta     $9B
        sta     ($9E),y
        dey
        lda     $7A
        sta     ($9E),y
        dey
        lda     $79
        sta     ($9E),y
        dey

SAV3:   lda     BASHEADER,y     ; index 34
        sta     ($9E),y
        dey
```

```
        bpl     SAV3


WRITE:  sec
        lda     $9B             ; Low end of BASIC
        sbc     $9E
        tay
        lda     $9C             ; High end of Basic
        sbc     $9F
        tax                     ; Basic start in 9E/9F, Length in X,Y(Low)

        iny                     ; Add 2 to length for start address PRG header
        bne     SAV31
        inx
SAV31:  iny
        bne     SAV32
        inx

SAV32:  LDA     ACIA_D          ; Clear all pending data
        LDA     #$3E            ; Write file
        JSR     EX_CMD

        tya                     ; Send file length and see if it can be allocated
        jsr     ACIA_WRITE
        txa
        jsr     ACIA_WRITE      ; length

        jsr     ACIA_READ       ; get file allocation reponse
        bne     SAVN
        beq     SAVR

SAVN:   lda     $9E             ; write start address (PRG format)
        jsr     ACIA_WRITE
        lda     $9F
        jsr     ACIA_WRITE

        ldy     #0
SAV4:   lda     ($9E),y
        jsr     ACIA_WRITE
        inc     $9E
        bne     SAV5
        inc     $9F
SAV5:   lda     $9B
        cmp     $9E
        bne     SAV4
        lda     $9C
        cmp     $9F
        bne     SAV4
        jsr     ACIA_READ       ; get file final reponse

SAVR:   jmp     LEN8            ; return value 0 or 1



BASHEADER:      ; PRG Basic header starting at $02D8 (new version V3)
        .db
        $A2,$03,$BD,$FB,$02,$95,$79,$CA,$10,$F8,$A9,$A5,$48,$A9,$FB,$48,$AE,$FF,$02,$D0,$0B,$A9
,$FD,$85,$C3,$A9,$BF,$85,$C4,$4C,$7A,$A4,$4C,$77,$A4
                ; followed by
                ; BASIC START ($02FB)
                ; BASIC END ($02FD)
                ; Autorun 1/0 ($02FF)

MENU:   .ASCII "MENU"  ; Specify your MENU file here (now without extension), "00" at the end
is required
PATH    .DB 0
```

```
; ***************************************************************************

GOSUB:
        JSR     GETSTR          ; Get string pointers or Error

GT3:    LDA     #$71
        STA     ACIA_S
        LDA     #$35            ; Define Path
        BNE     PATHNAME


; ***************************************************************************

GETSTR:
        bne     GT1             ; Jump on search name
        lda     #$00
        pha                     ; Pushing A(lenth)
        pha                     ; Pushing Y(High vector)
        pha                     ; Pushing X(Low vector)
        jmp     GT2
GT1:
        jsr     GSP             ; Get String parameters to Stack
GT2:
        pla                     ; ex: $FD / $FF
        sta     $9E
        pla                     ; ex: $7F / $7F / $77
        sta     $9F             ; Pointer to DIR string to $9E/9F
        pla                     ; String length >0 ?
        sta     PTMP            ; Remember length of string
        cmp     #32
        bcc     GTE
FCE:    jmp     BASIC_FCERR     ; FC-Error
GTE:    rts

; ***************************************************************************

GSP:                            ; ***** Get String parameters to Stack ******

        bne     GSP1            ; Command without parameters ?
GSPERR:

        jmp     BASIC_FCERR     ; STOP WITH FC ERROR
;
GSP1:
        pla                     ; Get caller Return address to $9C/9D
        sta     $9C
        pla
        sta     $9D             ; Remember Return address

        jsr     BASIC_EVAL      ; ROM BASIC, Evaluate expression
        bit     $5F             ; Check for String
        bpl     GSPERR          ; ERROR of not a Sting

        jsr     BASIC_RELEASE   ; ROM BASIC, Release string
                                ; Pointer in 71/72, length in A
        sta     PTMP
        pha                     ; Pushing A(lenth) must be <>0
        tya
        pha                     ; Pushing Y(High vector)
        txa
        pha                     ; Pushing X(Low vector)
        lda     $9D
        pha                     ; Restore Return Adr
        lda     $9C
        pha                     ; Restore Return Adr
```

```
        lda     PTMP
        rts

; ***************************************************************************

FILENAME:                       ; ***** Check and sent filename to SD ******
        LDY     SDEXIST
        BEQ     FAIL            ; Back on no SD

        jsr     GETSTR          ; Get string pointers or Error
        cmp     #00             ; Filename empty ?
        beq     FAIL            ; return with 0

        LDA     #$71
        STA     ACIA_S
        LDA     #$33
PATHNAME:
        JSR     ACIA_WRITE      ; Define name

        LDY     #0
LEN1:   LDA     ($9E),y
        CPY     PTMP            ; String length
        BNE     LEN2
        LDA     #0
LEN2:   INY
        JSR     ACIA_WRITE
        BNE     LEN1
        RTS

FAIL:   PLA
        PLA
        JMP     LEN8            ; Stop Executing

COPY_FN:
        LDY     #0
CLEN1:  LDA     ($9E),y
        CPY     PTMP            ; String length
        BNE     CLEN2
        LDA     #0
CLEN2:  INY
        STA     FN-1,Y
        CMP     #0
        BNE     CLEN1
        RTS


FN:     .DCB 23,$00             ; Filename Temp
        .DB $20
```

```
; Internal Loader 1.1
; Save not implemented
; Size 256 bytes (5 bytes left)


LOADER    = BASIC_LOADER-14
CEGMON    = $FF0E             ; On CEGMON this address is 0 and on SYN600 $F7

        ; Loader start address ($02CA)
                              ; 14 bytes header
        ; JMP   LOADER         ; $02D8-14
        ;.DB    $0             ; 3 TYPE 0=BAS(TXT) 1=LOD, 2=PRG(BIN) >2= Others
        ;.DB    $0             ; 4 ACTION 0=LOAD >0=SAVE. Bit 8 set= ignore Path
        ;.DB    $0, $0         ; 5,6 START for save
        ;.DB    $0, $0         ; 7,8 END for save
        ;.DB    <Path, >Path   ; 9,10, string must end with "00"
        ;.DB    <File, >File   ; 11,12, string must end with "00"
        ;.DB    $0             ; 13 AUTOSTART 0=not >0=yes for BASIC only

        ; PRG Program or BASIC_LOADER at $02D8 ** NEW version 2025 **
        ; BASHEADER PRG Basic starting at $02D8 (new version V3)
        ;
$A2,$03,$BD,$FB,$02,$95,$79,$CA,$10,$F8,$A9,$A5,$48,$A9,$FB,$48,$AE,$FF,$02,$D0,$0B,$A9,$FD,$8
5,$C3,$A9,$BF,$85,$C4,$4C,$7A,$A4,$4C,$77,$A4
        ; followed by
        ; BASIC START ($02FB)
        ; BASIC END ($02FD)
        ; BASIC Autorun 1/0 ($02FF)

        ; Loaded program address is in $0B/0C and can be run by X=USR(X)


; ********** Routines for ACIA ***********

SDEXIST:
        .DB     1             ; indicator, if SD card was found (1=true as this data is loaded
from SD)

ACIA_R:
        LDA     ACIA_S        ;(4)
        LSR                   ;(2)
        BCC     ACIA_R        ;(2), (3)
        LDA     ACIA_D        ;(4)
        RTS                   ;(6) sum min (18)

ACIA_W:
        PHA
WR1:    LDA     ACIA_S
        AND     #$02
        BEQ     WR1
        PLA
        STA     ACIA_D
        RTS


SCREEN_CLS:
        LDY     #0
        LDA     #$20
CL1:    STA     Screen,y
        STA     Screen+$100,y
        STA     Screen+$200,y
        STA     Screen+$300,y
        INY
        BNE     CL1
        RTS
```

```
; ********************+*************************************
; ***************** internal LOADER ***********************
;


EXLOAD: LDA     #$71
        STA     ACIA_S
        LDA     #$30
        JSR     ACIA_WRITE      ; Check SD card
        JSR     ACIA_READ
        BNE     LD              ; SD card found ?

STOP:   JMP     BASIC_FCERR     ; Soft Reset OSI, no return to BASIC -> change to FC Error in
BASIC


LD:     LDA     LOADER+4
        BMI     LD_NPF          ; Submit no Path (use last active one)

        LDA     LOADER+9
        STA     PTMP
        LDA     LOADER+10
        STA     PTMP+1

        LDA     #$71
        STA     ACIA_S
        LDA     #$35
        JSR     ACIA_WRITE      ; Define Path
        JSR     OUT_LOOP

LD_NPF:
        LDA     LOADER+11
        STA     PTMP
        LDA     LOADER+12
        STA     PTMP+1

        LDA     #$71
        STA     ACIA_S
        LDA     #$33
        JSR     ACIA_WRITE      ; Define Filename
        JSR     OUT_LOOP

        LDA     LOADER+4
        AND     #1
        BNE     STOP            ; SAVE not yet implemented

SDLOAD:
        LDA     #$20
        STA     $0201           ; last CEGMON char to be space

        LDA     #$71
        STA     ACIA_S
        LDA     #$39
        JSR     ACIA_WRITE      ; Get File

        JSR     ACIA_READ       ; Lengh ignored BASIC text files
        TAX                     ; Low in X
        JSR     ACIA_READ
        TAY                     ; High in Y
        BNE     NX0
        TXA
        BEQ     OUT_RT          ; File not found or empty (must be >0 bytes) - Do RTS to caller

NX0:    LDA     LOADER+3        ; CHECK TYPE 0=BAS 1=LOD, 2=PRG
        BNE     NX1             ; BASIC TEXT file ?
```

```
        LDA     #$FF
        STA     BASFLAG
        JMP     0               ; Go to BASIC and load, Stack will be reset
NX1:    CMP     #1              ; LOD file ?
        BNE     NX2
        BIT     CEGMON          ; Test CEGMON or SYN600 ROM
        BVC     NCEG
        STA     LODFLAG
        JMP     LOD             ; Go to ROM LOD loader
NCEG:   JMP     LODCEG

NX2:    TXA                     ; Handle PRG type files, and all DATA files
        BNE     LD21
        DEY                     ; substract 2 bytes of file length
LD21:   DEX

        BNE     LD22
        DEY
LD22:   DEX


        STY     PTMP
LD4:    JSR     ACIA_READ       ; PRG / DATA file Start address
        STA     TMP
        PHA
        JSR     ACIA_READ
        STA     TMP+1
        PHA

        LDY     #0
LD5:    JSR     ACIA_READ
        STA     (TMP),Y         ; Store
        INY
        BNE     LD6
        INC     TMP+1
        DEC     PTMP
LD6:    DEX
        BNE     LD5
        LDA     PTMP
        BNE     LD5

        PLA
        STA     TMP+1
        PLA
        STA     TMP
        LDA     LOADER+13       ; CHECK AUTOSTART (default for PRG is set, on DATA optional)
        BEQ     OUT_RT          ; Return on no Autostart
        JMP     (TMP)

OUT_LOOP:                       ; Write Path or Filename to ACIA , must end with "00"
        LDY     #0
OUT_1:  LDA     (PTMP),y
        INY
        JSR     ACIA_WRITE
        BNE     OUT_1
OUT_RT: RTS
```

```
;*******************************************************************************
;       MENU(.prg) tool relocated before BASIC extension
;       V1.200 2026 TB (MENU program stars below Basic extension)
;       Will keep the BASIC extension untouched, but uses the internal loader at $xB00
;        BASIC extension must not be present, as the loader part is refreshed again
;*******************************************************************************

START_ADR = $1B00             ; Version for 8k systems
;START_ADR = $3800            ; Version for 16k systems
;START_ADR = $5800            ; Version for 24k systems
;START_ADR = $7800            ; Version for 32k systems
;START_ADR = $9800            ; Version for 40k systems
MENUPLUS  = START_ADR - $0300
BASICPLUS = START_ADR + $0100

ACIA_READ = ACIA_R
ACIA_WRITE= ACIA_W


        ; LOADER start address ($02CA)
                              ; 14 bytes header
        ; JMP   LOADER        ; $02D8-14
        ;.DB    $0            ; 3 TYPE 0=BAS(TXT) 1=LOD, 2=PRG(BIN) >2= Others
        ;.DB    $0            ; 4 ACTION 0=LOAD >0=SAVE. Bit 8 set= ignore Path and Filename
        ;.DB    $0, $0        ; 5,6 START for save
        ;.DB    $0, $0        ; 7,8 END for save
        ;.DB    <Path, >Path  ; 9,10, string must end with "00"
        ;.DB    <File, >File  ; 11,12, string must end with "00"
        ;.DB    $0            ; 13 AUTOSTART 0=not >0=yes for BASIC only

        ; PRG Program or BASIC_LOADER at $02D8 ** NEW version 2025 **
        ; BASHEADER PRG Basic starting at $02D8 (new version V3)
        ;
$A2,$03,$BD,$FB,$02,$95,$79,$CA,$10,$F8,$A9,$A5,$48,$A9,$FB,$48,$AE,$FF,$02,$D0,$0B,$A9,$FD,$8
5,$C3,$A9,$BF,$85,$C4,$4C,$7A,$A4,$4C,$77,$A4
        ; followed by
        ; BASIC START ($02FB)
        ; BASIC END ($02FD)
        ; BASIC Autorun 1/0 ($02FF)

BASIC_LOADER   = $02D8        ; NEW version 2025
BASIC_HEADER   = $02FB        ; BASIC START ($02FB)
                              ; BASIC END ($02FD)
                              ; BASIC Autorun 1/0 ($02FF)

BASIC_LPT = $00BC             ; Basic Line Processing Vector in Zero-Page
BASIC_FCERR = $AE88           ; BASIC FC ERROR
RESET     = $FFFC
BOOT      = $FF00
KEY       = $FD00
Monitor   = $FE00
CEGFLAG   = $FF0E             ; On CEGMON this address is 0 and on SYN600 $F7
TextOut   = $BF2D
Screen    = $D000
PTMP      = $FC               ; Temp counter or flag
POS       = $FC
DATA      = $FD
XOFF      = $FE                     ; X-offset 32x32 vs 64x16
XPOS_S    = 5
XPOS_C    = 14
DIRStart  = $D100
PATH_show = $D340

ACIA_S    = $F000
ACIA_D    = $F001
```

```
BASFLAG  = $0203
LOD      = $FE2A      ; OSI LOAD ADDRESS
LODCEG   = $F96B      ; CEGMON LOAD ADDRESS
LODFLAG   = $FB
TMP      = $0B        ; Program start adress. Run with X=USR(X)


        .ORG    MENUPLUS-2

        .DB     <MENUPLUS
        .DB     >MENUPLUS

        .ORG    MENUPLUS       ; In free Memory


; ********************+*************************************
; ***************** MENU @ BASICPLUS position **************
;


MENU:   JSR     SCREEN_CLS
        LDA     #$4C           ; Prepare Jump address
        STA     LOADER
        LDA     #<EXLOAD
        STA     LOADER+1
        LDA     #>EXLOAD
        STA     LOADER+2
        LDY     #11
        LDA     #0
IN:     STA     LOADER+2,Y     ; Clear header
        DEY
        BNE     IN

        LDA     #XPOS_C
        BIT     CEGMON         ; Test CEGMON or SYN600 ROM
        BVC     CEG
        LDA     #XPOS_S
CEG:    STA     XOFF
        LDX     #>INF1
        LDY     #<INF1
        JSR     TEXTP
        LDX     #>INF0
        LDY     #<INF0
        JSR     TEXTP

        LDA     0              ; Needs to take place here !!!
        CMP     #$4C           ; Check for active BASIC
        BEQ     CARDOK         ; BASIC COLD START done ?

        LDX     #>WA1
        LDY     #<WA1
        JSR     TEXTP
MLEAVE: LDX     #>WA2
        LDY     #<WA2
        JSR     TEXTP
        JSR     KEY
        JMP     BOOT

EX_CMD: PHA                    ; Call extended command in A
        LDA     #$71
        STA     ACIA_S
        PLA
        JMP     ACIA_WRITE
```

```
CARDOK: LDA    #1
        STA    POS            ; Pos from on direcrory is shown

DIRL:
        LDA    ACIA_D         ; Clean up any pending byte
        LDX    #0             ; first line in D0C0+2
        CLC
        LDA    #<DIRStart     ; to be able to clear next line
        ADC    XOFF
        STA    TMP            ; Good for 8 lines until D300
        LDA    #>DIRStart
        STA    TMP+1
        LDA    #'>'           ; Selection cursor
        STA    (TMP,X)
        INC    TMP
        INC    TMP
        JSR    CLRLN          ; set and clear next line

        LDA    #$35
        JSR    EX_CMD         ; Start Path

        LDY    #0
PA1:    LDA    PA,y
        INY
        JSR    ACIA_WRITE
        BNE    PA1            ; until end of Path

        LDA    #$36
        JSR    EX_CMD         ; Get Directory
        BNE    DLR            ; always jump

DLC:    CPX    POS
        BCC    DLR
        CLC
        LDA    TMP
        ADC    #$40           ; Next line in double line mode
        STA    TMP
        BCC    CLRN
        INC    TMP+1
CLRN:   JSR    CLRLN          ; set and clear next line
        BEQ    DLREST         ; end of screen reached ?

DLR:    INX                   ; first line is 1
        LDY    #0
        JSR    ACIA_READ
        BEQ    LEA0           ; End of directory ?
        BNE    DL2
DL1:    JSR    ACIA_READ
        BEQ    DLC            ; End of string ?
DL2:    CPX    POS
        BCC    DL1            ; read empty, if X<POS
        CMP    #$5B
        BCS    DLS
        ORA    #$20
DLS:    STA    (TMP),Y        ; TextOut
        INY
        JMP    DL1

DLREST: JSR    ACIA_READ
        BEQ    LEA0           ; End of directory ?

        LDA    #$03           ; Ends DIR Process  (needs to be in active data transfer !!)
        STA    ACIA_S
        BNE    LEAVE

LEA0:   CLC
```

```
        LDA     TMP
        ADC     #$40            ; Next line in double line mode
        STA     TMP
        BCC     LEAX
        INC     TMP+1
LEAX:   JSR     CLRLN           ; End of directory list, Clear rest of screen
        BNE     LEA0            ; until end of screen


LEAVE:  JSR     KEY             ; ****** KEY Input *********

        CMP     #$1B            ; ESC to go back to Monitor
        BNE     LEA1
LEA00:  LDA     ACIA_D          ; Clean up any pending byte
        JMP     SCREEN_CLS      ; and return to caller

LEA1:   CMP     #'.'            ; > to step one down
        BNE     LEA2
        INC     POS
LEAR:   JMP     DIRL

LEA2:   CMP     #','            ; < to step one up
        BNE     LEA3
        DEC     POS
        BNE     LEAR
        INC     POS             ; Lowest POS is 1
        BNE     LEAR

LEA3:   CMP     #$0D            ; RETURN File/path selected?
        BEQ     LEA40
        CMP     #'D'            ; Delete
        BEQ     DELETE
        CMP     #'/'            ; Back to ROOT
        BNE     LEAVE
        LDA     #0              ; Back to root
        STA     PA
        JSR     Display_PA      ; in this case clear
        JMP     CARDOK

DELETE: LDY     #0              ; Execute program or go into subdirectory
DEL1:   LDA     DIRStart+2,Y
        CMP     #$20            ; End of name ?
        BEQ     DEL2
        CMP     #'['            ; Folder indicator
        BEQ     LEAR
        STA     FN,Y            ; Transfer File name
        INY
        CPY     #23
        BNE     DEL1
DEL2:   LDA     #0              ; terminate with 00
        STA     FN,Y

        LDA     #$33
        JSR     EX_CMD          ; Start File name

        LDY     #0
DEL3:   LDA     FN,y
        INY
        JSR     ACIA_WRITE
        BNE     DEL3            ; until end of File name

        LDA     #$3C
        JSR     EX_CMD          ; Delete File
        JSR     ACIA_READ
DELE:   JMP     DIRL
```

```
; ************************************************

LEA40:  LDY     #0              ; Execute program or go into subdirectory
        LDX     XOFF
LEA4:   LDA     DIRStart+2,X
        CMP     #$20            ; End of name ?
        BEQ     LEA5
        CMP     #'['            ; Folder indicator
        BEQ     LEA6
        STA     FN,Y            ; Transfer File name
        INX
        INY
        CPY     #23
        BNE     LEA4
LEA5:   LDA     #0              ; terminate with 00
        STA     FN,Y

        LDA     #<START_ADR     ; Set X=USR(X) address to Start of MENU
        STA     TMP
        LDA     #>START_ADR
        STA     TMP+1

        LDY     #0              ; Check extension
        STA     LOADER+13       ; Default no autostart
EXT0:   LDA     FN,Y
        BEQ     EXDEF           ; nothing found, take it as PRG (default)
        CMP     #'.'
        BEQ     EXT1            ; dot found ?
        INY
        BNE     EXT0            ; always jump
EXT1:   INY
        LDA     FN,Y
        BEQ     DELE            ; nothing found after dot, break here

        LDY     #0              ; filetype in Y
        AND     #$DF            ; capital letter
        CMP     #'B'            ; BAS
        BEQ     EXT2            ; Type 0
        INY
        CMP     #'L'            ; LOD
        BEQ     EXT2
        CMP     #'6'            ; 65V
        BEQ     EXT2
        CMP     #'H'            ; HEX
        BEQ     EXT2            ; Type 1
        INY
        CMP     #'P'            ; PRG
        BNE     DELE            ; unknown, do nothing

EXDEF:  LDY     #$83            ; Default PRG with Autorun
EXT2:   STY     LOADER+3        ; Setup and execute pogram

        JSR     SCREEN_CLS

        LDA     #0              ; LOAD with Path output before
        STA     LOADER+4
        LDA     #<PA
        STA     LOADER+9
        LDA     #>PA
        STA     LOADER+10
        LDA     #<FN
        STA     LOADER+11
        LDA     #>FN
        STA     LOADER+12
        JMP     LOADER          ; Back to internal LOADER
```

```
LEA6:   INX                     ; **** Folder selected ****
        INX

        LDA     #<PA            ; Check end of existing path
        STA     TMP
        LDA     #>PA
        STA     TMP+1

        LDY     #$FF
LEA61:  INY
        LDA     (TMP),Y
        BNE     LEA61
        CPY     #0              ; Are we already down the path ?
        BEQ     LEA7
        LDA     #'/'
        STA     (TMP),Y
        INY

LEA7:   LDA     DIRStart+2,X
        CMP     #$20
        BEQ     LEA8
        STA     (TMP),Y         ; Transfer PATH name
        INX
        INY
        CPY     #31             ; Max path length ?
        BCC     LEA7

LEA8:   LDA     #0
        STA     (TMP),Y
        JSR     Display_PA
        JMP     CARDOK          ; Restart in new folder


; ********** Routines for MENU ***********


TEXTP:  STY     TMP             ; Print text info from position X,Y
        STX     TMP+1
        LDY     #0
        LDA     (TMP),Y         ; Start address on screen
        CLC
        ADC     XOFF
        STA     TX+1
        INY
        LDA     (TMP),Y
        STA     TX+2
TXL:    INY
        LDA     (TMP),Y         ; text to display
        BEQ     CLRE            ; ends with zero
TX      STA     $8888
        INC     TX+1
        BNE     TXL
        INC     TX+2
        BNE     TXL             ; always

CLRLN:
        LDA     TMP+1
        CMP     #$D3
        BEQ     CLRE
        LDA     #$20            ; Clean line
        LDY     #38
CLR1:   STA     (TMP),Y
        DEY
        BPL     CLR1
        LDA     #1
CLRE:   RTS                     ; zero flag is set at end of screen
```

```
Display_PA:
       LDX     XOFF
       LDY     #0
DI2:   LDA     PA,Y
       STA     PATH_show,X
       BEQ     DI3
       INX
       INY
       BNE     DI2
DI3:   LDA     #$20
       STA     PATH_show,X
       INX
       CPX     #38
       BNE     DI3
       RTS


WA1:   .DB $80,$D0
       .ASCII "MISSING BASIC COLDSTART",0

WA2:   .DB $C0,$D0
       .ASCII "PRESS ANY KEY",0

INF0:  .DB $80,$D0
       .ASCII "*** OSI SD-MENU V2 ***",0

INF1:  .db $80,$D3
       .ASCII ">FWD <BWD /HOME DEL ESC"

PATH:  .db 0

FN:    .DCB 23,$00
       .DB $20
PA:    .DCB 31,$00
       .DB $20

       .ORG   START_ADR      ; In free Memory
       JMP    BASICPLUS      ; In reference to extended BASIC
       .INCLUDE "Internal_Loader_V1.asm"
```