

Symbiface 3 v2.x

15 April 2020

SF3 specs 2.x:

Microcontroller:	Cortex -M7 216 MHZ
RAM:	2 MB (SRAM)
"ROM"	2 MB (SRAM)
USB Host	Hid Mouse
USB Host	Fat32 mass storage device
AUDIO	MP3 Player
AUDIO	Recorder / prepare for VOIP
WIFI	IOT WIFI module/HTTP(S)/TLS/SSL/MQTT
RTC	+Battery
VU	Stereo level indicator
CPC 464	Extern memory hack
JTAG	Onboard
MEASUREMENTS	Power 5v, ARM temp, RTC batt
SD Card	for internal system storage
OLED	
BUZZER	



Multi computer manual:

The color of the selected text indicates for which computer it is intended:

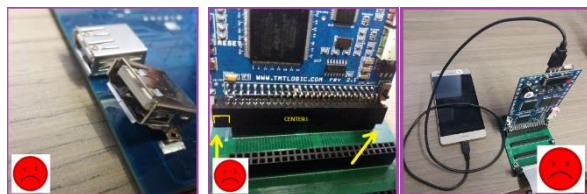
```
-- CPC ---
-- EF Enterprise --
-- PCW --
-- MSX --
```

Thanks to:

Prodatron	Symbos
SOS	cubeMDOS
Gflorez	Enterprise beta testing
Frits H	ARM support
TFM/GUNHED	futureOS
Stéphane	beta testing
Toto	CPC464 memory hack

Do not:

- Charge not your Mobile Phone or another USB power devices, with the USB port. USB is only for a HID mouse or USB mass store device (max 150mA)
- Don't use another External RAM or ROM (by default SF3.INI settings) see ENA_RAM / ENA_ROM
-



Important points:

Are you sure that the power to the SF3 is not below the 4.6 Volt ! The current of the SF3 is max 600mA

Regarding ROM[7]: **Not every CPC can install ROM 7 externally.**
Be aware that if your CPC cannot do this, you should NOT USE ROM[7] =
a hardware conflict may occur on the z80's data bus.

The first start:

Remove Sd card, if this is present.

Switch power to the SF3 and CPC



When the SD card is not present or there is a error found, the SF3 load automatic the SF3-RSX.BIN to CPC ROM[2]
SF3-RSX.BIN is made by SOS, there are a lot of RSX commands available.

Note, when the SD card is present, SF3-RSX.BIN is not automatic loaded. You must self this define in the SF3.INI !

In basic:

Press
mode 2

|LOG

The CPC shows follow information (look alike this)

```
----- Start system init -----
_sys      USB stick found
_usb      card is mounted 1: is oke
_sys      5v power is 5172 mV
_sys      CPC is actief !
_aux      Start init VS1053
_aux      VS1053 register 5 is: 1F40
..
..
END.
```

When there are errors or problems with the INI file, you can found here information

For CPC users, make an SF3_CPC.INI file in the root of your SD card, example see below.

For Enterprise users, make an SF3_EP.INI file in the root of your SD card, example see below.

Press a Fat32 mini SD card in the SD card holder. (you can remove the SD card by pressing once, then pulling it)
Is there an Error on the OLED screen? possible the SD card is not useable. Format the SD card as standard FAT or FAT32
Using a PC, place the favorite ROMS on this SD card. (SF3 use not long filenames at the moment).

Files from SOS can you found here: <https://github.com/BeCubel/Symbiface-3>

Example CPC 6128 SF3.INI

See google drive :Default USB/SD files

Example CPC 464 SF3.INI

```
*****
*                               CPC464
*                               Filename: SF3.INI
*****
*Wifi settings

*WIFI_PASS   = ..
*WIFI_SSID   = ..

*WIFI_CONNECT
*****
* CPC 464 select external RAM hack (CPC464=OFF is default)
* Note: CPC464=TOTO Forced the MREQ line of the Z80 !

CPC464=TOTO
*****
* SOS cubeMDOS Fat32 CPC
* If you are sure that you can replace CPC ROM 7 externally (Default OFF)
* ROM7_ENA=ON

ROM[6]   =SD:CUBEMDOS.ROM
ROM[9]   =SD:SF3-RSX.ROM

*Example  roms in subdirectory:

*ROM[6]   =SD:/CPC/ROMS/CUBEMDOS.ROM
*ROM[9]   =SD:/CPC/ROMS/SF3-RSX.ROM

*****
* Audio settings SYMAMP for symbos audio >> SEONE = MP3A

AUDIO=    SYMAMP
*****
```

Example CPC Enterprise SF3_EP.INI

```
*****
*                               Enterprise
*                               Filename: SF3_EP.INI
*****
Wifi settings
WIFI_PASS   = ...
WIFI_SSID   = ..
*WIFI_CONNECT
*****
*Enable Ram blocks

EPRAM[#08-#1F]
EPRAM[#30-#5F]
EPRAM[#73-#BB]

*Enable Rom blocks

ROM[#60]   =SD:/EP/SF3BOOT.ROM
ROM[#64]   =SD:/EP/SPEMU128.ROM
ROM[#67]   =SD:/EP/epd19uk.rom
ROM[#69]   =SD:/EP/forth.rom
ROM[#70]   =SD:/EP/BASIC.ROM
ROM[#71]   =SD:/EP/TWP8UKC.ROM
ROM[#72]   =HTTP:www.tmtlogic.com/cyrus.rom

*****
* Audio settings SYMAMP for symbos audio >> SEONE = MP3A
```

```
AUDIO= SYMAMP
*****
```

Reset SF3:

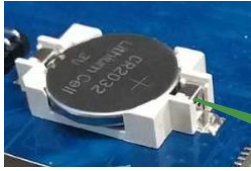
todo

Reset CPC:

todo

Replace RTC battery:

If the battery is empty, the battery can be change with type CR2032
me by careful !!
don't remove the battery with metal tool, a short circuit may occur



Warning lip on top

Don't use metal tools !

Problems:

SF3 does nothing and the red led right above burn:
the SF3 is in DFU mode, the switch right under must be pull down!

Test of the SF3 responds:

```
10 out &FD4F,170: REM 10101010
20 print(&FD4f)
30 out &FD4F,85: REM 01010101
40 print(&FD4f)
```

```
170
85
```

CPC prints 170 and 85 that's oke, if not there is another problem.

CPC464 extern memory does not work

In the INI file must be set: CPC464=TOTO (set no another CPC464= commands)
the MERQ signal is pull-up some Z80 processors are too strong for this function.

The power supply to the SF3 is too low. Minimum power limit is 4.7 volt
Check power with F125

No sound

for testing audio , use an headphone. Connect normal the green to an extern audio amplifier

No sound Symbols

In the INI file must be set: AUDIO=SYMAMP (set no another AUDIO= commands)
the green connector is the audio output

When symamp is started the OLED shows "SYMAMP", if not, you use the MSX version not the CPC

Wifi password of SSID error

Maybe you have use a space in the password or SSID
Default removed the SF3 all spaces and tabs in the INI file
Change the INI row with this:

```
Old
WIFI_SSID=ab cd
```

```
New
@WIFI_SSID=ab cd
```

Wifi no connection with AP

Reset the router
Maybe the channels are too close to each other.
Try your hotspot from the smartphone

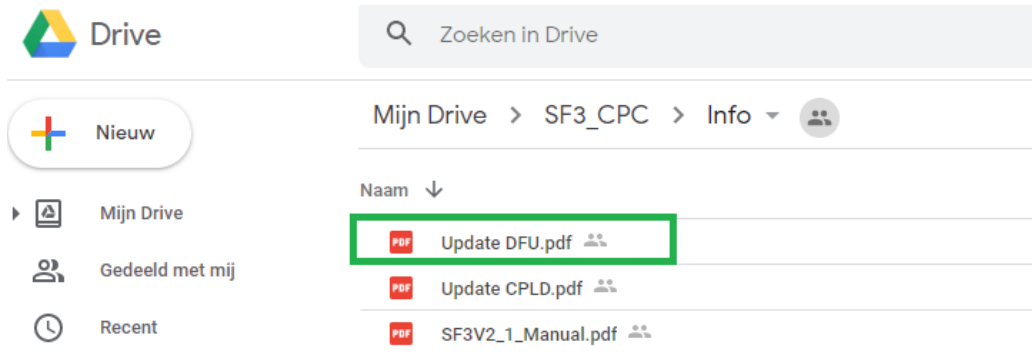
Questions:

Q1: Can I use the microphone input to mix the CPC sound?

A1: No. the pink microphone input is only intended for individual microphones and a headset.
The ground pin of this input is not the same as the ground from the SF3

Upgrade Firmware SF3 though DFU:

Go to the website www.tmtlogic.com > support > sf3_cpc > google link



SF3 Address range inp/out(&FD40 - &FD4F):

&FD40	reserve for AT command set	
&FD41	Command functions / response	
&FD42	port 0, Data read/write buffers	(Serveral)
&FD43	port 1, Data read/write buffers	(Fatfs read)
&FD44	port 2, Data read/write buffers	(Fatfs write)
&FD45	port 3, Data read buffers	(Wifi socket channel 0)
&FD46	port 4, Data read buffers	(Wifi socket channel 1)
&FD47	port 5, Data read buffers	(Wifi socket channel 2)
&FD48	port 6, Data read buffers	(WIFI socket channel 3)
&FD49	port 7, Data read/write buffers	(WIFI AT command's, non-socket)
&FD4A	port 8, Data read/write buffers	(2th Fatfs read)
&FD4B	port 9, Data read/write buffers	(2th Fatfs write)
&FD4C	reserve for TMTNET	
&FD4D	reserve for TMTNET	
&FD4E	WIFI process status byte	
&FD4F	system databus echo	

SF3 keywords:

SF3_CPC.INI	this file initializes the SF3, and must be place in the root of the SD card. - for CPC users use SF3.INI - for Enterprise users use SF3_EP.INI
ARM	Arm is the stand-alone microcontroller on the SF3. the most functions on the SF3 are controlled by the ARM
DFU	Device firmware upgrade, to upgrade the firmware of the ARM. there is a manual for updating see above (Upgrade Firmware SF3 though DFU)
WIFI module	WIFI is controlled by an external module on the backside of the SF3, this module handles all WIFI protocols.

Wifi procces status number inp(&FD4E):

Read:

- 0 Idle
- 1 Busy
- 2 Error
- 3 Offline
- 4 Socket not available

System databus echo inp/out(&FD4F)

Echo test:

Out &FD4F,234

Print(inp(&hFD4f))

234

Error table:

"Succeeded	"//..0
"Disk IO error	"//..1
"Assertion failed	"//..2
"The physical drive cannot work	"//..3
"Could not find the file	"//..4
"File not found	"//..5
"Invalid path name	"//..6
"Access denied directory full	"//..7
"Access denied	"//..8
"Invalid file/directory	"//..9
"Write protected	"//..10
"Invalid logical drive	"//..11
"The volume has no work area	"//..12
"No valid FAT volume	"//..13
"The f_mkfs() aborted	"//..14
"Time out	"//..15
"Rejected according sharing policy"	"//..16
"LFN could not be allocated	"//..17
"Number of open files	"//..18
"Invalid parameter	"//..19
"Unknow at commando	"//..20
"Syntax error	"//..21

```

"Usb busy                "///.22
"Invalid at command       "///.23
"Usb already installed    "///.24
"Parameter error          "///.25
"AT not in uppercase      "///.26
"ARM flash error          "///.27
"Invalid input            "///.28
"Wrong SE mode            "///.29
"Out of range             "///.30
"Wifi offline             "///.31
"Wifi_resonse_error       "///.32
"Wifi_stream error        "///.33
"Wifi socket error        "///.34
"Wifi socket is already open "///.35
"Wifi_RxBuffer_overflow   "///.36
""

```

LEDS:

```

Green Led RX      Burn when one of the 4 receive sockets is not empty
Blue led WIFI     Burn when Wifi is ONLINE
Blue led FAT32    Burn when USB stick is installed and ready to use
Green led MOUSE   Burn when mouse in ready to use

```

Yellow LED left under, process led z80 bus

Red LED right above, burn alone when ARM stay in DFU mode

Switches:

DFU... when turn up, the ARM stay in DFU mode (Device upgrade mode)

Extra functions:

Menu functions comes later

Total SF3 memory array map:

SF3 2v1:

```

Rom 0 - 127      &0000.0000 - &001F.FFFF      = 2 MB      128 x 16KB
Ram              &0020.0000 - &003F.FFFF      = 2 MB

```

ROM ARRAY:

Rom Array is a memory what's stored in the ARM memory, with a content of 16kb(cpc) and other 64kb.

See Function F62,x

This array can you easy change or modify and hacking.

There is also a command that upload the Array to CPC ROM location (F66)

Call ARM check ready:

Check of the ARM is busy, to make sure that the SF3_ARM ready to use.

REM call ARM ready

```

1000 I = INP(&FD41)
1010 IF I = 1 THEN RETURN
1020 IF I = 2 THEN PRINT"ARM Error"          Get Error, see F72,4 / F72,5
1030 REM your program
..

```

Call WIFI check ready:

Check of the Wifi module is busy, to make sure that the wifi module ready to use.

REM call WIFI ready

```

1000 I = INP(&FD4E)
1010 IF I = 1 THEN RETURN                    arm is not ready
1020 IF I > 1 THEN PRINT"WIFI Error"        Get Error, see F72,4 / F72,5
1030 REM your program
..

```

Call ARM response:

REM call ARM response

```

1000 I = INP(&FD41)
1010 IF I = 1 THEN GOTO 1000                wait processing 0 = oke 1 = busy 2 = error
1020 IF I = 2 THEN PRINT"ARM Error"        Get Error, see F72,4 / F72,5
1030 RETURN

```

Call WIFI response:

REM call WIFI resonse

```

1000 I = INP(&FD4E)                        wait processing 0 = oke 1 = busy 2 = error 3 = offline
1010 IF I = 1 THEN GOTO 1000
1020 IF I = 0 THEN RETURN
1030 IF I = 2 THEN PRINT"WIFI Error"        Get Error, see F72,4 / F72,5 or F91 for extra
1040 IF I = 3 THEN PRINT"WIFI Offline"
1050 IF I = 4 THEN PRINT"WIFI Socket not available"
1060 RETURN

```

Global functions: out &hFD41,xx

```

00      Reset buffer port 0      reset intern write buffer pointers0 (RTC, mouse, oled, and others)
01      Reset buffer port 1      reset intern write buffer pointers1 (fatfs read)

```

```

02      Reset buffer port 2          reset intern write buffer pointers2 (fatfs write)
03-06   no function, internal wifi streaming
07      Reset buffer port 7          reset intern write buffer pointers7 (wifi command's)

```

MOUSE:

```

F20      USB mouse data read
F21      USB mouse host info

F20      USB mouse data read

      110 out &FD41,20                      (no ARM response needed)

      120 print inp(&FD42);                  X as (relative) -100 + 100
      130 print inp(&FD42);                  Y as (relative) -100 + 100
      140 print inp(&FD42);                  byte (LSB is button 0)
      150 print inp(&FD42);                  Wheel steps   -100 + 100

F21      USB mouse host info

      110 out &FD41,21
      120 call ARM response

      Device information

      Max 1000 characters

      text      <13>
      text      <13>
      text      <13>

      Hid report information

      05,09      <13>
      75,01      <13>
      ..
      CC CC      <13>                      cc,cc is the end
      <10>

      1000 C = INP(&FD42)
      1010 IF C = 10 THEN PRINT:END
      1020 IF C = 13 THEN PRINT:GOTO 1000
      1020 PRINT CHR$(C);
      1030 GOTO 1000

```

AUDIO:

```

F40      Select Audio application mode

      App:

      0 = Disable SE-ONE by hardware &FF20-&FF27 (mp3 player)
      1 = Seone_MP3A
      2 = Seone_MP3B
      3 = Seone_FM
      4 = Seone_REG
      5 = AudioStream

      110 out &FD41,0                      reset write buffer pointer 0
      120 out &FD42, App                    Application
      130 out &FD41, 40                     function set date
      140 call ARM response

      Error response:
      Nr. 28   Invalid input;

F41      Seone audio settings

      F41,0:   Set volume L+R the same time

      110 out &FD41,0                      reset write buffer pointer 0
      120 out &FD42, Volume                 (0-100)
      130 out &FD41, 41                     function set date
      140 call ARM response

      Error response:
      none

      F41,1:   Set volume L R

      10 out &FD41,0                      reset write buffer pointer 0
      120 out &FD42, Volume left            (0-100)
      125 out &FD42, Volume right          (0-100)
      130 out &FD41, 41                     function set date
      140 call ARM response

      Error response:
      none

      TODO      set bass,treable

```

MSD device file level:

```

F47 Mass storage device file level functions

      Device is:

      0: SD
      1: USB

```

```

F47,0:      Status
            120 out &FD41,0                      reset pointers 0
            130 out &FD42,0
            140 call ARM response
            150 print inp(&FD42);                  status Read      0 = Idle
                                                    1 = Active
            160 print inp(&FD42);                  status Write     0 = Idle
                                                    1 = Active

F47,1:      Set Parameters

            SD      ( 0):
            USB      ( 1):
            HTTP      (10):
            HTTPS     (11):

F47,2:      Get Parameters
            Set Path Read
            Get Path Write
            Get file info (string format)
            Files size
            Files attrib

F47,3       End Save

F47,10:     Load file

            Modify: 11-02-2020                      24 to 32 bits

            //todo path

            120 out &FD41,0                      reset pointers 0
            130 out &FD42,10
            140 out &FD42,device number
            150 out &FD42,chr$("T")                (path) file name
            160 out &FD42,chr$("E")
            170 out &FD42,chr$("S")
            180 out &FD42,chr$("T")
            190 out &FD42,chr$(".".")
            200 out &FD42,chr$("T")
            210 out &FD42,chr$("X")
            220 out &FD42,chr$("T")
            230 out &FD41,47
            240 call ARM response

            250 print inp(&FD42);                  Highest byte file size 32 bit
            270 print inp(&FD42);                  file size 32 bit
            271 print inp(&FD42);                  file size 32 bit
            280 print inp(&FD42);                  Lowest byte file size 32 bit
            290 print inp(&FD42);                  Attrib
            300 print inp(&FD42);                  Attrib
            310 print inp(&FD42);                  Attrib
            320 print inp(&FD42);                  Attrib
            330 print inp(&FD42);                  Attrib

            Error response:
            Nr. 4      _Could not find the file

F47,11:     Save file

            Modify: 11-02-2020                      24 to 32 bits

            120 out &FD41,0                      reset pointers 0
            130 out &FD42,11
            140 out &FD42,device number
            150 out &FD42,filesizes                MSB 32 bit
            160 out &FD42,filesizes                32 bit
            161 out &FD42,filesizes                32 bit
            180 out &FD42,filesizes                LSB 32 bit

            190 out &FD42,0                      reserve attrib
            200 out &FD42,0
            210 out &FD42,0
            220 out &FD42,0

            230 out &FD42,chr$("T")                (path) file name
            240 out &FD42,chr$("E")
            250 out &FD42,chr$("S")
            260 out &FD42,chr$("T")
            270 out &FD42,chr$(".".")
            280 out &FD42,chr$("T")
            290 out &FD42,chr$("X")
            300 out &FD42,chr$("T")
            310 out &FD41,47
            320 call ARM response

F47,12:     End Save      (close file)

```

```

100 out &FD41,0
110 out &FD42,12
120 out &FD41,47
130 call ARM response

F47,12: Kill/Del file
F47,13: Files/Cat
F47,14: Copy

F48      Mass storage device file level read data

          Maximum read is 2048 bytes

          100 out &FD41,48
          110 call ARM response

          120 print inp(&FD4A);          Bytes read High (max 2048 bytes)
          30 print inp(&FD4A);          Bytes read Low

          160 print inp(&FD4A);          Data
          170 print inp(&FD4A);          Data
          180 print inp(&FD4A);          Data
          ..
          ..

F49      Mass storage device file level write data

          Maximum read is 2048 bytes

          10
          100 out &FD41,9                reset pointers 9 (B)
          110 out &FD4B,data
          120 out &FD4B,data
          130 out &FD4B,data
          140 out &FD4B,data
          ..
          ..
          ..                maximal 2048 bytes
          200 out &FD41,49
          210 call ARM response

          220 print inp(&FD4B);          Bytes written High (max 2048 bytes)
          230 print inp(&FD4B);          Bytes written Low

```

FAT32 (USB MSD) sector level:

```

F51      Read sector from USB MSD
F52      Write protection FAT32 (software setting)
F53      Write sector to USB Stick
F54      USB fatfs host info
F55      Check if USB stick available

```

info:

When USB stick is installed, the blue led with the text FATFS will burn.

while exchanging the USB stick you must make sure that the FAT32 software on the CPC supports it.

Changing the USB stick takes about 10s, change not to fast

F50 (removed)

```

F51      Read sector from USB MSD

          110 call ARM ready
          120 out &FD41,1                reset pointers 1
          130 out &FD43, byte3          MSB      32bit
          140 out &FD43, byte2
          150 out &FD43, byte1
          160 out &FD43, byte0          LSB      32bit

          170 out &FD43, Sectors        Sector write (max 8)
          180 out &FD41,51              active function
          190 call ARM response

          1000 S = sectors * 512
          1010 FOR T = 1 TO S
          1020 PRINT INP(&FD43);
          1030 NEXT T
          1040 I = INP(&FD43)
          1050 IF I <> &1A THEN PRINT "End of file check error"

```

F52 Write protection FAT32 (software setting)

write protection lock off:

```

110 out &FD41,2                reset buffer pointer 2 fatfs write
120 out &FD44, &AA              lock of code
130 out &FD41,52                active function
140 call ARM response          16kbyte 16ms / 512bytes 900us

```

write protection lock on:

```

110 out &FD41,2                reset buffer pointer 2 fatfs write

```



```

120 out &FD44, <> &AA
130 out &FD41,52
140 call ARM response

lock of code (all but &AA)
active function
16kbyte 16ms / 512bytes 900us

F53      Write sector to USB Stick

110 out &FD41,2
120 out &FD44, byte3
130 out &FD44, byte2
140 out &FD44, byte1
150 out &FD44, byte0

reset buffer pointer 2 fatfs write
MSB      32bit

160 out &FD44, Sectors
Sectors write (max 8)
LSB      32bit

1000 S = sectors * 512
1010 A [0..S] = DATA, DATA.
1020 FOR T = 1 TO S
1030 OUT &FD44, A[T]
1040 NEXT T
1050 OUT &FD44, &1A

2000 out &FD41,53
2010 call ARM response

active function

F54      USB fatfs host info
// TODO

110 out &FD41,54
120 call ARM response

Device information

Max 1000 characters

text      <13>
text      <13>
text      <13>
text      <10>

1000 C = INP(&FD42)
1010 IF C = 10 THEN PRINT: END
1020 IF C = 13 THEN PRINT: GOTO 1000
1020 PRINT CHR$(C);
1030 GOTO 1000

F55      Check if USB stick available

110 out &FD41,55
120 call ARM response

130 print inp(&FD42);
140 print inp(&FD42);

1 = USB stick is available and installed
0 = USB stick is NOT available
hot-plug counter
every hot-plug the counter are increment +1 (8bit)

```

USB CPC DSK file support:

```

F56,0:   open *,DSK file
F56,1:   DSK read sector
F56,2:   (todo) DSK Write sector
F56,3:   DSK Close
F56,10:  (todo) DSK info

```

F56 DSK image functions:

Please make a DUMMY.DSK in the root of the device

F56,0: Start/open DSK

Device is:

0: SD
1: USB

```

100 out &FD41, 0
110 out &FD42, 0
120 out &FD42, device number
130 out &FD43, byte3
140 out &FD43, byte2
150 out &FD43, byte1
160 out &FD43, byte0
170 out &FD41, 56
180 call ARM response

sub function 0
device number
MSB      LBA 32bit
LSB      LBA
Function 56

```

Error Response:
Nr.-- FAT error table

F56,1: DSK read sector

```

120 out &FD41,0
130 out &FD42,1
140 out &FD42, track
150 out &FD42, Sector
160 out &FD42, Side
170 out &FD41, 56
180 call ARM response

```

```

1000 FOR T = 1 TO ??
1020 PRINT INP(&FD42);
1030 NEXT T

```

Error Response:
?? Nr.19 Invalid parameter
Nr.1 Disk error

```

F56,2      (todo) DSK Write sector

          110 out &FD41, 0
          130 out &FD42, 2
          140 out &FD42, track
          150 out &FD42, Sector
          160 out &FD42, Side

          1010 A [0..S] = DATA, DATA.
          1020 FOR T = 1 TO S
          1030 OUT &FD42, A[T]
          1040 NEXT T
          2000 out &FD41,56
          2010 call ARM response

F56,3      DSK Close

          110 out &FD41, 0
          120 out &FD42, 3
          130 out &FD41,56
          140 call ARM response

F56,10     DSK info
          (TODO)

F56,100    FAT32 (2th) Select Volume (default 0=SD)

          Volume:
          0: SD msd card
          1: USB msd stick

          110 out &FD41,0
          120 out &FD42,100
          130 out &FD42,Volume 0..1
          140 out &FD41,56
          150 call ARM response
                                     active function

F56,101    FAT32 (2th) Write protection (z80 software setting)

          write protection lock off:

          110 out &FD41,0
          120 out &FD42,101
          130 out &FD42,&AA
          140 out &FD41,56
          150 call ARM response
                                     lock of code
                                     active function

          write protection lock on:

          110 out &FD41,0
          120 out &FD42,101
          130 out &FD42, <> &AA
          140 out &FD41,56
          150 call ARM response
                                     lock of code (all but &AA)
                                     active function

F56,102    FAT32 (2th) host info
          (todo)

```

FAT32 (SD) sector level:

```

F57      FAT32 (2th) Check if SD card is available
F58      FAT32 (2th) Read sector from MSD
F59      FAT32 (2th) Write sector to MSD

F57      FAT32 (SD) Check if volume is available

          140 out &FD41,57
          150 call ARM response

          160 print inp(&FD42);
          170 print inp(&FD42);

                                     1 = Volume is available and installed
                                     0 = Volume is NOT available
                                     3 (TODO)
                                     hot-plug counter
                                     every hot-plug the counter are increment +1 (8bit)

F58      FAT32 (2th) Read sector from MSD

          110 call ARM ready
          120 out &FD41,8
          130 out &FD4A, byte3
          140 out &FD4A, byte2
          150 out &FD4A, byte1
          160 out &FD4A, byte0

                                     reset pointers 8(A)
                                     MSB      32bit

          170 out &FD4A, Sectors
          180 out &FD41,58
          190 call ARM response

                                     LSB      32bit

          1000 S = sectors * 512
          1010 FOR T = 1 TO S
          1020 PRINT INP(&FD4A);
          1030 NEXT T
          1040 I = INP(&FD4A)
          1050 IF I <> &1A THEN PRINT "End of file check error"

F59      FAT32 (2th) Write sector to MSD

          110 out &FD41,9
          120 out &FD4B, byte3
          130 out &FD4B, byte2
          140 out &FD4B, byte1

                                     reset buffer pointer 9 (B) fatfs write
                                     MSB      32bit

```

```

150 out &FD4B, byte0          LSB      32bit
160 out &FD4B, Sectors        Sectors write (max 4)

1000 S = sectors * 512
1010 A [0..S] = DATA, DATA.
1020 FOR T = 1 TO S
1030 OUT &FD4B, A[T]
1040 NEXT T
1050 OUT &FD4B, &1A

2000 out &FD41,59
2010 call ARM response

```

ROM:

```

F62,11: Set Enable/Disable All ROMs 0-126
F62,3:  Set Disable/Enable ROM[n] by Hardware
F62,0:  Get Hardware ROM[n] enable / disable

```

```

F62,1:  Get list of Hardware Enables ROMs (127 is lower ROM)
F62,2:  Get all first byte of the ROM's (127 is lower ROM)

```

```

F62,4:  Write first byte of the ROM with 255
F62,6:  Read data form ROM address nn
F62,7:  Write data to ROM address nn

```

```

F62,8:  Clear Array
F62,5:  Fill Array and ROM[] with number n

```

```

F62,10: CPC/Z80 >>> Array
F65     WEB    >>> Array
F62,14  MSD     >>> Array

```

```

F62,9:  Array >>> Z80
F66     Array >>> ROM[]
F62,13  Array >>> External RAM
F62,12:  Array >>> MSD

```

ROM TABLE sd card:

```

F62,20  copy Array >>> ROM[nn]    sd:..ROMTABLE
F62,21  Delete Rom[nn]            sd:..ROMTABLE
F62,22  Get File size from ROM[nn] sd:..ROMTABLE
F62,25  Save Free text for ROM[nn] sd:..ROMTABLE
F62,26  Load Free text from ROM[nn] sd:..ROMTABLE

```

```

F63     CPC/Z80 >>> CPC ROM    (direct)
F64     MSD     >>> CPC ROM    (direct)

```

Rom Sub functions:

```

F62,0:  Get Hardware Rom enable / disable

        110 out &FD41, 0          reset write buffer pointer 0
        120 out &FD42, 0          active sub function
        130 out &FD42, Rom number
        140 out &FD41, 62         active function
        150 call ARM response
        160 print inp(&FD42)

        Error Response:
        Nr.19   Invalid parameter

F62,1:  Get list of Hardware Enables ROMs (0-15 127 is lower ROM)

        110 out &FD41, 0          reset write buffer pointer 0
        120 out &FD42, 1          active sub function
        130 out &FD41, 62         active function
        140 call ARM response

        150 T = 0
        160 I = INP(&hfd42)        Example Rom Numbers Enables:  2 3 4 13 255
        170 IF I = 255 THEN PRINT "TOTAL ROM ENABLE: ";T: END
        180 T = T + 1
        190 PRINT I;
        200 GOTO 160
        ...
        Last byte is 255          Rom 255 don't use

        Error Response:
        Nr.30   Out of range

F62,2:  Get all first byte of the ROMs (127 is lower ROM)

        110 out &FD41, 0          reset write buffer pointer 0
        120 out &FD42, 2          active sub function
        130 out &FD41, 62         active function
        140 call ARM response

        150 FOR T = 0 TO 127
        160 PRINT "ROM: ";T; " FIRST BYTE: ";INP(&FD42)
        170 NEXT T

F62,3:  Set Disable/Enable ROM by Hardware

        110 out &FD41, 0          reset write buffer pointer 0
        120 out &FD42, 3          active sub function
        130 out &FD42, Rom number 0-15 127
        140 out &FD42, 0 or 1     1 = Enable

```

```

150 out &FD41, 62          active function
160 call ARM response

Error Response:
Nr.19      Invalid parameter

F62,4:   Clear first byte of ROM with 255

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 4          active sub function
130 out &FD42, Rom number
140 out &FD41, 62          active function
150 call ARM response

Error Response:
Nr.19      Invalid parameter

F62,5:   Fill Rom with number n

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 5          active sub function
130 out &FD42, Rom number
140 out &FD42, fill number
150 out &FD41, 062        active function
160 call ARM response

Error Response:
Nr. 15     Timeout        Background upload to Rom more than 5 sec.
                                please contact me !!          tmtlogic

F62,6:   Read one data form Rom address nn

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 6          active sub function
130 out &FD42, Rom number
140 out &FD42, High byte address      address max &4000
150 out &FD42, Low byte address
160 out &FD41, 062        active function
170 call ARM response

180 print inp(&hfd42)      data

Error Response:
Nr.30      Out of range
Nr. 19     Invalid parameter        Rom select > 127

F62,7:   Write one data to Rom address nn

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 7          active sub function
130 out &FD42, Rom number
140 out &FD42, High byte address      address max &4000
150 out &FD42, Low byte address
160 OUT &FD42, data
170 out &FD41, 062        active function
180 call ARM response

Error Response:
Nr.30      Out of range
Nr. 19     Invalid parameter        Rom select > 31

F62,8:   Clear Array

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 8          active sub function
130 out &FD41, 062        active function
140 call ARM response

F62,9:   Array >>> CPC/Z80 data, begin adr nn
Note: Address is autoincrement

Maximal address is &4000

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 9          active sub function
130 out &FD42, High byte address      address
140 out &FD42, Low byte address
150 out &FD41, 062        active function
160 call ARM response

170 print inp(&hfd42)      data
180 print inp(&hfd42)      data
190 print inp(&hfd42)      data
..

Error response:
Nr.30      Out of range

F62,10:  CPC/Z80 >>> Array data begin adr nn

Note: Address is auto increment

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 10         active sub function
130 out &FD42, High byte address      Start address max &4000
140 out &FD42, Low byte address

150 out &FD42, data
160 out &FD42, data

```

```

170 out &FD42, data
..
200 out &FD41, 062          active function
210 call ARM response

Error response:
Nr.30      Out of range

F62,11: Set Enable/Disable All ROMs 0-126 (by hardware)

110 out &FD41, 0            reset write buffer pointer 0
120 out &FD42, 11          active sub functions
130 out &FD42, 0-1         0 = Disable >=1 Enable(default)
140 out &FD41, 062          active function
150 call ARM response

F62,12: Array >>> MSD (sd / usb)      16kb

110 out &FD41, 0            reset write buffer pointer 0
120 out &FD42, 12
130 out &FD42, 0            0= SD 1=USB
140 out &FD42, ASC("T")     maximal 8+4 characters 12345678.123
150 out &FD42, ASC("E")
160 out &FD42, ASC("S")
170 out &FD42, ASC("T")
180 out &FD42, ASC("F")
190 out &FD42, ASC("I")
200 out &FD42, ASC("L")
210 out &FD42, ASC("E")
220 out &FD42, ASC(".")
230 out &FD42, ASC("R")
240 out &FD42, ASC("O")
250 out &FD42, ASC("M")
260 out &FD41, 062          active function
270 call ARM response

F62,13 Array >>> External RAM[]

Array size is &4000

110 out &FD41, 0            reset write buffer pointers0
120 out &FD42, 13
120 out &FD42, RAM number   RAM number 0-127
120 out &FD42, MEM block    MEM block 0-255
140 out &FD41, 62          reset write buffer pointers0
150 call ARM response

Error response:
Nr.13      Invalid parameter
Nr.15      Timeout

F62,14 MSD >>> Array

File size is maximal &4000
File size is maximal &FFFF

Example path: map1/map2/map3/test.rom

110 out &FD41, 0
120 out &FD42, 14          reset write buffer pointer 0
120 out &FD42, device number 0=SD 1=USB
130 out &FD42, path        example: test/hello.rom
140 out &FD42, path
150 out &FD42, path
..
300 out &FD41, 62          active function
310 call ARM response

Error response:
Nr.6.      Invalid path name
Nr.19.     Invalid parameter
Nr.30.     Out of memory

F62,15 res
F62,16 res
F62,17 res
F62,18 res
F62,19 res

ROMTABLE is a directory on then SD card
For every computer system his own PATH, example CPC: SD:/CPC/ROMTABLE
When function F62,20 or F62,25 is used this directory is automatically created.
If the SF3 can find this directory, all ROMs will be loaded automatically
This is done before executing the INI file

ROM001.BIN means , the content of this file loaded in ROM place 1, or segment 1

F62,20 copy Array >>> ROM[nn] sd:..ROMTABLE

This function create this file: ROMnnn.BIN nnn = rom number

110 out &FD41, 0            reset write buffer pointer 0
120 out &FD42, 20          active sub function
130 out &FD42, Rom number
170 out &FD41, 062          active function
180 call ARM response

Error Response:
Nr.30      Out of range
Nr. 19     Invalid parameter      Rom select > 31

F62,21 Delete Rom[nn]          sd:..ROMTABLE

```

This function delete the files ROMnnn.BIN and ROMnnn.TXT

```

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 21         active sub function
130 out &FD42, Rom number
140 out &FD41, 62         active function
180 call ARM response

F62,22  Get list of available ROM          sd:..ROMTABLE

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 22         active sub function
130 out &FD42, Rom number
140 out &FD41, 62         active function
180 call ARM response
190 Print"Roms available:"
200 i = inp(&hfd42)
210 if i <> 0 then PRINT i;",";:GOTO 200

```

```

F62,23  Get File size from ROM[nn]      sd:..ROMTABLE

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 23         active sub function
130 out &FD42, Rom number
140 out &FD41, 62         active function
180 call ARM response
200 print chr$(inp(&FD42))    1
210 print chr$(inp(&FD42))    6
220 print chr$(inp(&FD42))    3
230 print chr$(inp(&FD42))    8
240 print chr$(inp(&FD42))    4

When inp(&FD42) = 0 , last character

```

```

F62,25  Save Free text for ROM[nn]      sd:..ROMTABLE

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 25
130 out &FD42, Rom number
140 out &FD42, asc("T")
150 out &FD42, asc("e")
160 out &FD42, asc("s")
170 out &FD42, asc("t")
180 out &FD41, 62
190 call ARM response

```

```

F62,26  Load Free text from ROM[nn]    sd:..ROMTABLE

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 26
130 out &FD42, Rom number
180 out &FD41, 62
190 call ARM response
200 print chr$(inp(&FD42))    t
210 print chr$(inp(&FD42))    e
220 print chr$(inp(&FD42))    s
230 print chr$(inp(&FD42))    t

When inp(&FD42) = 0 , last character

```

F63 Transfer CPC To ROM

Note:
CRC header is automatic detected

```

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, Rom number  ROM number 0-127 127 is Lower ROM
120 out &FD42, Rom number  Mem block number 0-255
130 out &FD42, 0          0 = normal. (todo) 1 = under CPC reset)

140 out &FD42, data        Max data size is &4000+80
150 out &FD42, data
160 out &FD42, data
..

300 out &FD41, 063        active function
310 call ARM response

```

```

Error response:
Nr. 15  Timeout          Background upload to Rom more than 5 sec.
                             please contact me !! TMTLOGIC
Nr. 19  Invalid parameter Rom select > 127
Nr. 30  Out if range      Rom > 0x4080
Nr. 30  Out if range      Mem > 0xFFFF

```

F64 Transfer MSD To ROM

Write ROM data from SD USB to ROM

Example path: map1/map2/map3/test.rom

..TODO LOWER ROM upload under CPC RESET

```

110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, device number 0=SD 1=USB
130 out &FD42, Rom number  Rom number 0-126 127 is Lower ROM
130 out &FD42, Mem number  Mem number 0-255

140 out &FD42, path        example: test/hello.rom

```

```

150 out &FD42, path
160 out &FD42, path
..
300 out &FD41, 064          active function
310 call ARM response

```

```

Error response:
Nr.6.      Invalid path name
Nr.19.     Invalid parameter

```

F65 Transfer Internet To Array

Note:
After download, the CPC CRC Header is automatic removed

Target:

```

0 = HTTP
1 = HTTPS

```

```

(todo)
2 = MQTT
3 = TMTNET
4 = FTP
5 = SMB

```

..TODO LOWER ROM upload under CPC RESET

```

.
120 call WIFI ready

130 out &FD41, 0          reset write buffer pointer 0
140 out &FD42, target     0=HTTP 1=HTTPS
150 out &FD42, ASC("w")   example: www.tmtlogic.com/hello.rom
160 out &FD42, ASC("w")
170 out &FD42, ASC("w")
180 out &FD42, ASC(".")
190 out &FD42, ASC("t")
...
300 out &FD41, 065        active function
310 call ARM response

Error response:
Nr.6.      Invalid path name
Nr.31      offline
Nr.33      steam error    if http response not good

320 call WIFI response

```

F66 Transfer Array to ROM[]

```

.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, Rom number Rom number 0-127 127 is Lower ROM
130 out &FD42, 0          reserve default =0, later function load under Reset
131 out &FD42, High       size High byte
132 out &FD42, LOW        size low byte

140 out &FD41, 66         reset write buffer pointers0
150 call ARM response

Error response:
Nr.15      Time out

```

RAM:

F67,0: Set Enable/Disable external RAM (by hardware)

F72,14: Read data from Total Ram array SF3 (4MB)
F72,15: Write data to Total Ram array SF3 (4MB)

F67 Ram Sub functions:

F67,0: Set Enable/Disable All external RAM

```

.
110 out &FD41, 0          reset write buffer pointer 0
120 out &FD42, 0          active sub functions
130 out &FD42, 0-1        0 = Disable >=1 Enable(default)
140 out &FD41, 67        active function
150 call ARM response

```

F67,1: reserved for cubeMDos write memory 256 bytes
(if you modify this 256 bytes, contact SOS)

```

120 out &FD41,0          reset pointers 0
130 out &FD42,1
140 out &FD42,start address    0..255
150 out &FD42,data
150 out &FD42,..

150 out &FD41,67

```

```

320 call ARM response    < 500ns
Error response:
Nr.30      out of range

```

F67,2: reserved for cubeMDos read memory 256 bytes
(if you modify this 256 bytes, contact SOS)

```

120 out &FD41,0          reset pointers 0
130 out &FD42,2
140 out &FD42,start address 0..255
150 out &FD42,..

150 out &FD41,67
320 call ARM response    < 500ns
399 print inp(&hfd42)     data
399 print inp(&hfd42)     data
399 print inp(&hfd42)     data
...
```

SYSTEM:

F70 Get DFU filename

F72,0: Reset CPC
F79: Get 3 buttons

F72,11: Get logbook
F71 Get System info
F72 System sub Functions:

F72,1: Memory test
F72,2: SD card test
F72,3: Hello.rom test
F72,10: Echo test Microphone and Ear
F72,6: Led test

F72,4: Get Error number
F72,5: Get Error Text
F72,16: Get (n) Error Text
F250,7: Make Error

F72,7: Set VU LED
F72,12: Set bit for activate this ini File command #...
F72,13: Get bit for activate this ini File command #...
F72,14: Read data from Total Ram array SF3
F72,15: Write data to Total Ram array SF3

F72,16: Reinit ROM RAM settings user.CFG file

F70 Get DFU filename

```

.
110 out &FD41,70          Function Get DFU filename
120 call ARM response
130 C = inp(&FD42)
140 IF C = 10 THEN PRINT: END
150 IF C = 13 THEN PRINT: GOTO 130
160 PRINT CHR$(C);
170 GOTO 130
```

F71 Get System info

Note:
Check if the Symbiface is connected, you can read the first 3 characters "Sys", this is always the same text

```

.
110 out &FD41,71          Function Get System info
120 call ARM response
```

```

System: SYMBIFACE 3v2.1    <13>
                          <13>
SF3V2D1                   <13>
                          <13>
MPU: CORTEX M7 216Mhz     <13>
RAM: 2MB                  <13>
ROM: 2MB                  <13>
WIFI: RAK473              <13>
AUDIO: VS1053            <13>
OLED: SS1306 128x32      <13>
CPLD: ALTERA 172560dd    <13>
                          <10>
```

```

130 C = inp(&FD42)
140 IF C = 10 THEN PRINT: END
150 IF C = 13 THEN PRINT: GOTO 130
160 PRINT CHR$(C);
170 GOTO 130
```

F72 System sub functions:

F72,0: Reset CPC

Note:
Activate bus Reset pin of the CPC

One byte for safety access !
if the access number <> 123 then Oled prints "No access"

```

.
110 out &FD41,0          reset write buffer pointers0
120 out &FD42, 0         active sub function
130 out &FD42, 123       access number (123)

140 out &FD41, 72       active function
150 call ARM response
```

F72,1: Memory test

Note:
time around 10 sec,

don't use another ROMs
Information is reading on the Oled display

```
.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 1          active sub function
130 out &FD41, 72         active function
140 call ARM response
```

F72,2: SD card test

Note:
Symbiface3 create a file on the SD card (SDTEST.TXT)

```
.
110 out &FD41, 0          write buffer pointers0
120 out &FD42, 2          active sub function
130 out &FD41, 72         active function
140 call ARM response
```

F72,3: Hello.rom test (made by SOS)

Note:
Running Hello.rom (is internal stored in arm) ROM2
After CPC reset is this rom activate
Print to CPC screen: Hello
Or activate with |H

```
.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 3          active sub function
130 out &FD41, 72         active function
140 call ARM response
```

F72,4: Get Error number

Note:
remove the error text on the oled
reset WIFI error

```
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 4          active sub function
130 out &FD41, 72         active function
140 call ARM response
```

```
150 print(inp(&FD42))      Error number
```

(option)

```
160 print chr$((inp(&FD42))) E
170 print chr$((inp(&FD42))) R
180 print chr$((inp(&FD42))) R
190 print inp(&FD42)        32x..
..
300 print inp(&FD42)        <10>
310 print inp(&FD42)        <13>
```

F72,5: Get Error Text

Note: remove the error text on the oled
Note the Error string is 34 bytes long and 2 bytes (LF CR)

```
.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 5          active sub function
130 out &FD41, 72         active function
140 call ARM response

160 print chr$((inp(&FD42))) E
170 print chr$((inp(&FD42))) R
180 print chr$((inp(&FD42))) R
190 print inp(&FD42)        32x..
..
300 print inp(&FD42)        <10>
310 print inp(&FD42)        <13>
```

F72,6: Led test

Note: test all leds on board

```
if (&FD41 = 1) return      check ARM busy?
out &FD41, 0               reset write buffer pointers0
out &FD42, 6               active subfunction
out &FD41, 72              active function
call ARM response
```

F72,7: Set VU LED

Note: first byte is left, second is right
LSB is green,MSB is red
This function set the MP3 app off (F40 = 0)

```
.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 7          active sub function
130 out &FD42, byte        left VU
140 out &FD42, byte        right VU
150 out &FD41, 72         active function
160 call ARM response
```

F72,10: Echo test Microphone and Ear

```

Note:      Connect the Headset on the SF3
           Green connector is Ear
           Pink connector is microphone

.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 10         active sub function
130 out &FD41, 72         active function
140 call ARM response

F72,11:  Get logbook

The logging can be deactivated by the SF3.INI file DEBUG=OFF

.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 11         active sub function
130 out &FD41, 72         active function
140 call ARM response

150 print chr$((inp(&FD42))) L      max &h2000 bytes
160 print chr$((inp(&FD42))) O
170 print chr$((inp(&FD42))) G
..
..
500 print chr$((inp(&FD42))) '0'    last byte is &h00

F72,12:  Set bit for activate this INI File command #...

Note,
When this bit is disable, the INI file skipped the line with #<>

.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 12         active subfunction
130 out &FD42, 0 or 1      =0 then skip the command line
140 out &FD41, 72         active function
150 call ARM response

F72,13:  Get bit for activate this INI File command #...

.
110 out &FD41, 0          reset write buffer pointers0

120 out &FD42, 13         active sub function
130 out &FD41, 72         active function
140 call ARM response
150 Print inp(&FD42)       = 0 then skip command

F72,14:  Read data from Total Ram array SF3 (4MB)
         &0000.0000 - &003F.FFFF

.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 14         active sub function
130 out &FD42, High       32 bits address High byte
140 out &FD42,
150 out &FD42,
160 out &FD42, LOW        32 bits address
170 out &FD41, 72         active function
180 call ARM response
190 Print inp(&FD42)       data

Error response
Nr.30      Out of range

F72,15:  Write data to Total Ram array SF3
         &0000.0000 - &003F.FFFF

.
110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 15         active sub function
130 out &FD42, High       32 bits address High byte
140 out &FD42,
150 out &FD42,
160 out &FD42, LOW        32 bits address
170 out &FD42, data
180 out &FD41, 72         active function
190 call ARM response

Error response
Nr.30      Out of range

F72,16:  Get (n) Error Text

Note: remove the error text on the oled

Note the Error string is 34 bytes long and 2 bytes (LF CR)

110 out &FD41, 0          reset write buffer pointers0
120 out &FD42, 16         active sub function
130 out &FD42, N          error number 0-37
140 out &FD41, 72         active function
150 call ARM response

160 print chr$((inp(&FD42)))      E
170 print chr$((inp(&FD42)))      R
180 print chr$((inp(&FD42)))      R
190 print inp(&FD42)              32x..
..
500 print inp(&FD42)              <10>
510 print inp(&FD42)              <13>

F72,18:  Activate the NMI from the z80

```

Make an puls from 100ms

```
110 out &FD41, 0
120 out &FD42, 18
140 out &FD41, 72      active function
150 call ARM response
```

F72,19: Activate WIFI WEB update

```
110 out &FD41, 0
120 out &FD42, 19
130 out &FD42, 53      access code
140 out &FD42, 88      access code
150 out &FD41, 72      active function
160 call ARM response
```

Error response:

```
19.      Invalid parameter
32.      Wifi response error
```

See WIFI update manual

F79 Get 3 buttons

Note: bits: x x x x x right mid left (sorry)

```
110 out &FD41, 79      active function
120 call ARM response
130 print inp(&FD42)    0-7 (3 bits)
```

BUZZER:

F80 Set buzzer

Note: Tone 00 buzzer off
 01 low
 02 middles
 03 high

rhythm 00 continue
 01 slow
 02 middles
 03 fast

```
110 out &FD41,0         reset buffer pointers 0
120 out &FD42, tone      2 bits
130 out &FD42, rhythm    2 bits
140 REM out &FD42, res
150 REM Out &FD42, res
160 out &FD41,80d        active function
170 call ARM response
```

WIFI: --- in development ---

F65 Transfer Internet To Array

F90,0: Wifi modele ResetPin on/off

F90,1: Set wifi password

F90,2: Set wifi ssid network name

F90,3: wifi connect with access point

F90,4: wifi At-Command Get DNS

F90,11: wifi At-Command IP config

F90,10: wifi At-Command Close socket

F90,5: wifi At-Command Open TCP socket client

F90,6: WIFI Static ip

F90,13: wifi Get free socket number

F90,8: wifi Send socket data

F90,18: WIFI Check and Read socket data

F90,21: WIFI Check and Read and send to audiostream

F91: WIFI Get WIFI module state (pointer 0 !)

inp(&FD4E)status number

F90 WIFI Sub functions

F90,0: WIFI module Reset Pin on/off

```
110 out &FD41, 7      resets write buffer pointers 7
120 out &FD49, 0      active sub function
130 out &FD49, reset 0 or 1  0 = RESET WIFI module
                           1 = WIFI active
140 out &FD41, 90      active function
150 call ARM response
```

F90,1: Set WIFI password

```
110 out &FD41, 7      reset write buffer pointer 7
120 out &FD49, 1      active sub function
130 out &FD49, 'h'     data
140 out &FD49, 'e'     data
150 out &FD49, 'y'     data
160 out &FD41, 90      active function
```

```

170 call ARM response

F90,2: Set WIFI SSID network name

110 out &FD41, 7          reset write buffer pointer 7
120 out &FD49, 2          active sub function
130 out &FD49, 'N'        data
140 out &FD49, 'e'        data
150 out &FD49, 't'        data
160 out &FD41, 90        active function
170 call ARM response

F90,3: WIFI connect with access point

Note: this function can take a few seconds, The SF3 ARM is busy with the WIFI module

110 out &FD41, 7          reset write buffer pointer 7
120 out &FD49, 3          active sub function
125 out &FD49, DHCP       0 = client 1 = server
130 out &FD41, 90        active function
140 call ARM response

Error response:
Nr. 31  WIFI offline

F90,4: WIFI At-Command Get DNS

110 out &FD41, 7          reset write buffer pointer 7
120 out &FD49, 4          active sub function
130 out &FD49, 'w'        data  www.google.com
140 out &FD49, 'w'        data
150 out &FD49, 'w'        data
160 out &FD49, '.'        data
170 out &FD49, 'g'        data
..
300 out &FD41, 90        active function
310 call ARM response
320 call WIFI response

500 print inp(&hfd49)      172
510 print inp(&hfd49)      217
520 print inp(&hfd49)      17
530 print inp(&hfd49)      35

Error response:

offline
Nr. 32  Wifi_resonse_error
Nr. 33  Wifi stream error

F90,5: WIFI At-Command Open TCP socket client

120 out &FD41, 7          reset write buffer pointer 7
130 out &FD42, 5          active sub functions
140 out &FD49,192         IP address 1th
150 out &FD49,168         IP address 2th
160 out &FD49,2          IP address 3th
170 out &FD49,151         IP address 4th
180 out &FD49,0D         Port address High (example 3333)
190 out &FD49,05         Port address Low
200 out &FD49,01         Socket Channel 0-3

210 out &FD41, 90        active function
220 call ARM response
230 call WIFI response

Error response:
Nr. 35  "Wifi socket is already open"
Nr. 31  "Wifi offline"
Nr. 25  "Parameter error"

F90,6: WIFI Static ip

120 out &FD41, 7          reset write buffer pointer 7
130 out &FD42, 6          active sub functions

140 out &FD49, 192         IP address 1th
150 out &FD49, 168         address 2th
160 out &FD49, 2          address 3th
170 out &FD49, 151         address 4th

180 out &FD49, 255         NETMASK address 1th
190 out &FD49, 255         address 2th
200 out &FD49, 255         address 3th
210 out &FD49, 0          address 4th

220 out &FD49,192         GATEWAY address 1th
230 out &FD49,168         address 2th
240 out &FD49,2          address 3th
250 out &FD49,254         address 4th

260 out &FD49,192         DNS1 address 1th
270 out &FD49,168         address 2th
280 out &FD49,2          address 3th
290 out &FD49,254         address 4th

300 out &FD49,0          DNS2 address 1th
310 out &FD49,0          address 2th
320 out &FD49,0          address 3th
330 out &FD49,0          address 4th

340 out &FD41, 90        active function
350 call ARM response
360 call WIFI response

```

```

Error response:

Nr. 25    "Parameter error"

*F90,7:   Deleted !!

F90,8:    WIFI Send socket data

note: send buffer is maximal 1004 bytes

.

120 out &FD41, 7
130 out &FD49, 8
140 out &FD49, socket channel

150 out &FD49, data
160 out &FD49, data
170 out &FD49, data
180 out &FD49, data
190 out &FD49, data
..
400 out &FD41, 90
410 call ARM response
Error response:
Nr. 19    "Invalid parameter"
420 call WIFI response

*F90,9    Deleted !!

F90,10:   WIFI At-Command Close socket

.
120 out &FD41, 7                reset write buffer pointers0
130 out &FD49, 10              active sub function
140 out &FD49, n                socket channel number 0-3
150 out &FD41, 90              active function
160 call ARM response
Error Response:
Nr.19    Invalid parameter
Nr.34    Wifi socket error

170 call WIFI response
Error response:
Nr.34    Wifi socket error

F90,11:   WIFI At-Command IP config

120 out &FD41, 7                reset write buffer pointers7
130 out &FD49, 11              active sub function
140 out &FD41, 90              active function
150 call ARM response
160 call WIFI response

6x print inp(&hfd49);           mac:      &hx &hx &hx &hx &hx &hx
4x print inp(&hfd49);           IP:       192 168 2 19
4x print inp(&hfd49);           Mask:     255 255 255 0
4x print inp(&hfd49);           Gateway:  192 168 2 192
4x print inp(&hfd49);           dns1:    168 2 254 0
4x print inp(&hfd49);           dns2:    0 0 0 0

*F90,12:   Deleted !! wifi Get socket state

F90,13:    WIFI Get free socket number

Note: Gives a free socket number

.
110 out &FD41, 7                reset write buffer pointers 7
120 out &FD49, 13              active sub function
130 out &FD41, 90              active function
140 call ARM response
150 print inp(&hfd49);          free socket number 0-3   255 is no free socket or offline

F90,16     check type connection

0 none
1 TCP client
2 TCP server
3 UDP client
4 UDP server

userSocket(0-3) > intern Flag number 0-7 8-12
Out:
Socket type,
number of connections (server), max 7
2 byte port, 6 byte ip
2 byte port, 6 byte ip
2 byte port, 6 byte ip
2 byte port, 6 byte ip
2 byte port, 6 byte ip
2 byte port, 6 byte ip
2 byte port, 6 byte ip

F90,17     check socket connection

120 out &FD41, 7                reset write buffer pointers 7

```

```

130 out &FD49, 17          active sub function
140 out &FD49, Socket      (0-3)
150 out &FD41, 90          active function
160 call ARM response
170 print inp(&hfd49);      0 = socket channel open
                           1 = socket channel close
                           2 = WIFI offline

180 print inp(&hfd49);      0 none
                           1 TCP client
                           2 TCP server
                           3 UDP client
                           4 UDP server

```

F90,18 WIFI Check and Read socket data to rx buffer

```

120 out &FD41, 7
130 out &FD49, 18
140 out &FD49, Socket      (0-3)
150 out &FD41, 90
160 call ARM response
170 call WIFI response
180 print inp (&FD49)      content of received bytes    High byte
190 print inp (&FD49)      content of received bytes    Low byte
200 print inp (&FD49)      content of received bytes    Low byte

```

Read the data (the content of received bytes decrement automatic -1) every read

```

1000 print inp (&fd45 + Socket)    data
1010 print inp (&fd45 + Socket)    data
..
..

```

F90,21 WIFI Check if there is data and send it to the audiostream (webradio)

```

120 out &FD41, 7
130 out &FD49, 21
140 out &FD49, Socket      (0-3)
150 out &FD41, 90
160 call ARM response
170 call WIFI response

```

F91 WIFI Get WIFI module stat

States are :

This is changed ...

0 = WifiRak_idle

```

.
110 out &FD41, 91          active function
120 call ARM response
130 print inp(&hfd42)      WIFI module state    number
140 print inp(&hfd42)      High byte 16bit Len RX At-Command data buffer
150 print inp(&hfd42)      Low    byte 16bit

```

RTC:

F105 RTC sub functions
F105,0: RTC Set Backup register
F105,1: RTC Get Backup register

```

F100    Set time    hour minute sec
F101    Get time    hour minute sec
F102    Reset RTC
F103    Get time BCD hour minute sec
F104    Get date    BCD day month year
F110    Set date    day month year
F111    Get date    day month year

```

F100 Set time hour minute sec

```

.
110 out &FD41,0            reset buffer pointers 0
120 out &FD42, hours       hours 0 - 23
130 out &FD42, minutes     minutes 0 - 59
140 out &FD42, seconds     seconds 0 - 59
150 out &FD41,100          function set time
160 call ARM response

```

F101 Get time hour minute sec

```

110 out &FD41,101          active function
120 call ARM response

130 print inp(&FD42)        hours    0-23
140 print inp(&FD42)        minutes 0-59
150 print inp(&FD42)        seconds 0-59

```

F102 Reset RTC
Note: Only needed when are problem with the time and date, all value are 0 !

```

110 out &FD41,102          active function

```

```

120 call ARM response

F103  Get time BCD hour minute sec

110 out &FD41,103          active function
120 call ARM response

130 print hex$(inp(&FD42))    hours    0-23
140 print hex$(inp(&FD42))    minutes   0-59
150 print hex$(inp(&FD42))    seconds   0-59

F104  Get date BCD day month year

110 out &FD41,104          active function
120 call ARM response

130 print hex$(inp(&FD42))    day       1-31
140 print hex$(inp(&FD42))    month     1-12
150 print hex$(inp(&FD42))    year      0-99 (real is + 2000)
160 print hex$(inp(&FD42))    Weekday   0-31

F105  RTC sub functions

F105,0:  RTC Set Backup register

110 out &FD41, 0
120 out &FD42, 0
120 out &FD42, Backup Register (0-119)
130 out &FD42, Data
140 out &FD41, 105
150 call ARM response

Error response:
(25) parameter error

F105,1:  RTC Get Backup register

110 out &FD41, 0
120 out &FD42, 1
120 out &FD42, Backup Register (0-119)
140 out &FD41, 105
150 call ARM response
170 print inp(&FD42)

Error response:
(25) parameter error
Todo .. batt low

F105,2:  RTC Set date in BCD

110 out &FD41,0            reset buffer pointers 0
120 out &FD42,2            day       1 -31
130 out &FD42, day         day       1 -31
140 out &FD42, month       month     1 -12
150 out &FD42, year        year      0-99 (real is + 2000)
160 out &FD41,105          active function
170 call ARM response

Error response:
(25) parameter error

F105,3:  RTC Set time in BCD

110 out &FD41,0
120 out &FD42,3
130 out &FD42, hours       hours 0 - 23
140 out &FD42, minutes     minutes 0 - 59
150 out &FD42, seconds     seconds 0 - 59
160 out &FD41,105          function set time
170 call ARM response

Error response:
(25) parameter error

F110  Set date day month year

110 out &FD41,0            reset buffer pointers 0
120 out &FD42, day         day       1 -31
130 out &FD42, month       month     1 -12
140 out &FD42, year        year      0-99 (real is + 2000)
150 out &FD41,110          active function
160 call ARM response

F111  Get date day month year

110 out &FD41,111          active function
120 call ARM response
130 print inp(&FD42)        day       1-31
140 print inp(&FD42)        month     1-12
150 print inp(&FD42)        year      0-99 (real is + 2000)
160 print inp(&FD42)        Weekday   0-31

```

MEASUREMENTS:

```

F125  Get Power Supply 5v , RTC batt voltage , ARM temperature

110 out &FD41,125          active function
120 call ARM response

130 print chr$(inp(&FD42))    4        power Supply 4565mV
140 print chr$(inp(&FD42))    5
150 print chr$(inp(&FD42))    6
160 print chr$(inp(&FD42))    5
170 print inp(&FD42)          <13>

180 print chr$(inp(&FD42))    3        RTC batt power 3101mV (todo)

```

```

190 print chr$(inp(&FD42))          1
200 print chr$(inp(&FD42))          0
210 print chr$(inp(&FD42))          1
220 print inp(&FD42)                <13>

230 print chr$(inp(&FD42))          1      ARM internal Temperature 18.6 Celsius (todo)
240 print chr$(inp(&FD42))          8
250 print chr$(inp(&FD42))          6
260 print chr$(inp(&FD42))          0
270 print inp(&FD42)                <13>
280 print inp(&FD42)                <10>

```

TMTNET: (BETA)

F130 tmtnet subfunctions

F130,0 connection login with tmtnet broker
091219

```

10 out &hfd41,0
20 out &hfd42,0
30 out &hfd42,High 0x0b      userID 3000 + your userID decimal (1-60)
40 out &hfd42,Low 0xbc
50 out &hfd42,'h'
60 out &hfd42,'a'
70 out &hfd42,'s'
80 out &hfd41,130
90 call ARM response

```

F130,1 get status
091219

```

10 out &hfd41,0
30 out &hfd42,1
40 out &hfd41,130
50 call ARM response
70 print inp(&hfd42)

0 = Wifi Offline
1 = tcp connection with the server, but user is not connected
2 = waiting for server feedback
3 = user is connected

```

F130,2 query online status to all users

```

10 out &hfd41,0
30 out &hfd42,2
40 out &hfd41,130
50 call ARM response

```

F130,3 query online status to all users

```

note:
wait 2 seconds after F130,2
UserID 0 = server

10 out &hfd41,0
30 out &hfd42,3
40 out &hfd41,130
50 call ARM response

70 for w = 1 to 2000:next w

70 for t = 0 to 20
80 print "UserID: ";t;" online: ";inp(&hfd42)
90 next t

```

F130,4 start measure response time

```

10 out &hfd41,0
30 out &hfd42,4
40 out &hfd41,130

```

F130,5 get response time

```

Note:
wait 1 seconds after F130,4

10 out &hfd41,0
30 out &hfd42,5
40 out &hfd41,130
50 call ARM response

70 c = inp(&hfd42)
80 if c = 10 then print :end
80 print chr$(c);
90 goto 70

...ms
0065ms

```

F130,6 UserID logoff Disconnect from tmtnet broker

```

10 out &hfd41,0
30 out &hfd42,6
40 out &hfd41,130

```


OLED DISPLAY:

```

F200    Oled clear display (black)
F201    Oled Write data to SSD1306
F202    Oled print text in Upper row
F203    Oled print text in Lower row
F210    Oled print text on display  x, y, font, text

F200    Oled clear display (black)

        110 out &FD41,200                active function
        120 call ARM response

F201    Oled Write data to SSD1306

        Manual : https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

        110 out &FD41,0                    reset intern write buffer pointers

        example scroll to right

        120 out &FD42, 0x2E                De-Activated scroll
        130 out &FD42, 0xA3                Direction
        140 out &FD42, 0x00                Dummy
        150 out &FD42, 0x00                Start page
        160 out &FD42, 0x02                Interval
        170 out &FD42, 0x02                End page
        180 out &FD42, 0x0B                vertical scrolling offset
        190 out &FD42, 0x2F                Activated scroll
        ..
        ..
        ..
        300 out &FD41,201                active function
        310 call ARM response

F202    Oled print text in Upper row

        Note: maximum is 12 characters

        110 out &FD41,0                    reset intern writes buffer pointers
        120 out &FD42, char
        130 out &FD42, char
        140 out &FD42, char
        ..
        ..
        300 out &FD41,202                active function
        310 call ARM response

F203    Oled print text in lower row

        See F202

F210    Oled print text on display  x, y, font, text

        Note: this function is change, now with auto erase

        110 out &FD41,0                    reset write pointer 0
        120 out &FD42, x                    x as 0-127          (left to right)
        130 out &FD42, y                    y as 0-31           (up to down)
        140 out &FD42, font                 Font types :10,18,26
        150 out &FD42, char                 to print character
        160 out &FD42, char                 to print character
        170 out &FD42, char                 to print character
        ..
        ..
        300 out &FD41,210                active function
        310 call ARM response

```

TEST: Development test functions:

```

F250    test sub Functions:

        F250,7:  Make Error

                Note:
                For testing error number en text routine

                110 out &FD41, 0            reset write buffer pointers0
                120 out &FD42, 7            active sub function
                130 out &FD42, 34           Error number
                140 out &FD41, 250         active function
                150 call ARM response

```

SF3.INI file:

Note:

- This file must be place in the root of the SD card.
- Tabs and spaces has no effect.
- Uppercase is important.

Keywords:

```

ENA_ROM=ON
ENA_ROM=OFF
ENA_RAM=
EPRAM[]
ROM[]=

```

```
WIFI_SSID=
WIFI_PASS=
WIFI_CONNECT
```

```
WIFI_IP=
WIFI_NETMASK=
WIFI_GATEWAY=
WIFI_DNS1=
WIFI_DNS2=
```

```
WIFI_DHCP=STATIC
WIFI_DHCP=DYNAMIC
```

```
RESET
RESET=ON
RESET=OFF
RESET=PLS
AUDIO=SYMAMP
AUDIO=PLAYMSD
```

```
CPC464=OFF (Default CPC6128)
CPC464=TOTO
```

```
*          text line
?..... 12 characters user text
<        block begin
>        block end
\t       tab
```

Note use no spaces into command line!!

```
*          REM free text

<        begin block
>        End block

@        the SF3 don't remove the spaces in this line
```

EPRAM[] Activate a memory segment to the extern ram

```
EPRAM[96]
EPRAM[#60]
EPRAM[#11-#FF] an segment range
```

--- RAM enable bits ---*

```
bit 0      Memory block 4h .. 7h
bit 1      Memory block 8h .. Fh
bit 2      Memory block 10h .. 17h
bit 3      Memory block 18h .. 1Fh
bit 4      Memory block 20h .. 27h
bit 5      Memory block 28h .. 2Fh
bit 6      Memory block 30h .. 37h
bit 7      Memory block 38h .. 3Fh
bit 8      Memory block 40h .. 47h
bit 9      Memory block 48h .. 4Fh
bit 10     Memory block 50h .. 57h
bit 11     Memory block 58h .. 5Fh
bit 12     Memory block 60h .. 67h      enabled at power on
bit 13     Memory block 68h .. 6Fh      enabled at power on
bit 14     Memory block 70h .. 77h
bit 15     Memory block 78h .. 7Fh
bit 16     Memory block 80h .. 87h
bit 17     Memory block 88h .. 8Fh
bit 18     Memory block 90h .. 97h
bit 19     Memory block 98h .. 9Fh
bit 20     Memory block A0h .. A7h
bit 21     Memory block A8h .. AFh
bit 22     Memory block B0h .. B7h
bit 23     Memory block B8h .. BFh
bit 24     Memory block C0h .. C7h
bit 25     Memory block C8h .. CFh
bit 26     Memory block D0h .. D7h
bit 27     Memory block D8h .. DFh
bit 28     Memory block E0h .. E7h
bit 29     Memory block E8h .. EFh
bit 30     Memory block F0h .. F7h
bit 31     Memory block F8h .. FBh
```

--- Rom enable bit ---**

```
bit 0      Memory block 4h .. 7h
bit 1      Memory block 8h .. Fh
bit 2      Memory block 10h .. 17h
bit 3      Memory block 18h .. 1Fh
bit 4      Memory block 20h .. 27h
bit 5      Memory block 28h .. 2Fh
bit 6      Memory block 30h .. 37h
bit 7      Memory block 38h .. 3Fh
bit 8      Memory block 40h .. 47h
bit 9      Memory block 48h .. 4Fh
bit 10     Memory block 50h .. 57h
bit 11     Memory block 58h .. 5Fh
bit 12     Memory block 60h .. 67h      enabled at power on
bit 13     Memory block 68h .. 6Fh
bit 14     Memory block 70h .. 77h
bit 15     Memory block 78h .. 7Fh
```

```
*          The bit are use by the SF3
**         RAM and Rom enable bit are set automatically by upload an ROM[]=
```

```
DEBUG=OFF Disable debug information to COM port and Logbook (default on)
          When debug is ON , the SF3 is slower
```

CPC464=

```
Select the CPC464 external memory hack
thanks tot TotO and RICH

CPC464=OFF          (Default CPC6128)
CPC464=TOTO
(todo)CPC464=RICH   (Revaldhino)
```

```
ROM[]=
    Upload ROM:          upload ROM to SF3 ROM memory
    Note: when file comes from internet. Than First activate WIFI_INIT

    ROM[x]    = MSD:FILENAME

    X         = 0 to 126      decimal(127 is lower ROM)
    X         = #00 to #7F    hexadecimal

    MSD       = SD            from SD card
               = USB          from USB stick
               = HTTP         from Internet
               = HTTPS        from Internet

    Example, upload hello.rom from SD to CPC ROM place 2

    ROM[2] = SD:HELLO.ROM

WIFI settings:

    WIFI_SSID =abc           your WIFI SSID
    WIFI_PASS =abc           your WIFI Password

    Dhcp IP static:

    WIFI_DHCP  =STATIC
    WIFI_IP    =192.168.2.100
    WIFI_NETMASK =192.168.2.100
    WIFI_DNS1  =192.168.2.100
    WIFI_DNS2  =192.168.2.100

    Dhcp IP dynamic:

    WIFI_DHCP  =DYNAMIC

    WIFI_CONNECT          WIFI module goes connected (time is +/-10 sec)

RESET    Reset the CPC after ini.file
RESET_ON Set the CPC in reset
RESET_OFF Reset the CPC in reset
RESET_PLS reset CPC immediately
```

Examples:

Example: SF3.INI

```
*****
*                SF3 init File                *
*****
*Wi-Fi initialization

WIFI_SSID = yourSSID
WIFI_PASS = yourpassword

*****

*if SF3 wifi as DHCP server :
WIFI_DHCP  = STATIC

WIFI_IP    = 192.168.002.111
WIFI_NETMASK = 255.255.255.001
WIFI_GATEWAY = 192.168.002.254
WIFI_DNS1    = 192.168.002.254
WIFI_DNS2    = 000.000.000.000
*****

*if SF3 wifi as DHCP client:
WIFI_DHCP    = DYNAMIC
*****

WIFI_CONNECT

*****
*Enable Disable ROM (0-126) (ON is default)
*ROM_ENA     =ON
*ROM_ENA     =OFF
*****
*Enable Disable RAM (ON is default)
*RAM_ENA     =ON
*RAM_ENA     =OFF
*****
*Upload ROM'S

*127 is lower rom

ROM[127] =SD:FW.ROM

ROM[2]   =SD:MAP/HELLO.ROM

*****
*Game from github
```

```

ROM[2]      =HTTPS:raw.githubusercontent.com/HansTillema/SF3/master/Arkanoid.rom

*****
*Start symbOS from sd card

*ROM[1]=SD:sym-romA.rom
*ROM[2]=SD:sym-romB.rom
*ROM[3]=SD:sym-romC.rom
*ROM[4]=SD:sym-romD.rom

*from tmtlogic site

*ROM[1]=HTTP:www.tmtlogic.com/symbos/sym-romA.rom
*ROM[2]=HTTP:www.tmtlogic.com/symbos/sym-romB.rom
*ROM[3]=HTTP:www.tmtlogic.com/symbos/sym-romC.rom
*ROM[4]=HTTP:www.tmtlogic.com/symbos/sym-romD.rom
*****

```

Wifi:

Open TCP client:

WIFI must be connected, WIFI led must burn

Wi-Fi connect initialization:

SF3.INI file:

WIFI_SSID=your SSID
WIFI_PASS=your Password

WIFI_DHCP = DYNAMIC

WIFI_CONNECT

Or

F90,2 Set WIFI password
F90,3 Set WIFI ssid network name
F90,4 WIFI connect

WIFI led must burn

Get free socket number:

F90,13: WIFI Get free socket number (0-3)

Open TCP socket:

F90,5: WIFI At-Command Open TCP socket client (0-3)

Send socket data:

F90,8: WIFI Send socket data

Receive socket data

F90,18

Close TCP socket

F90,10: WIFI At-Command Close socket

SE-ONE

Description:

The SE-ONE is an music player for the MSX and CPC home computers.

The SE-ONE has a lot of modes. This mode and another option can be set and read with a user-friendly AT command set or Register commands. Mode MP3A is the Sunrise compatible mode (Symbos uses this mode with SymAmp), there is IO compatibility with the Sunrise mp3 cartridge.

Features:

- Fully user-friendly AT command set
- MP3 chip VS1053 decodes multiple formats:
- Mostly compatible with the Sunrise MP3 player
- Stereo VU LED bar
- User IO ports for CPC: &FF20 - &FF27

Thanks to:

EdoZ. (SymbOS),RJB (MSX / TEST),Totaly (webshop),MVM(MSX computerclub),Frits H.(C controllers),Kebu synthesizer music,Prodatron(SymbOS /CPC) ,GFlorez(Enterprise)

Also a word of thank to the Sunrise Foundation for the clear manual / documentation of the MP3 cartridge. This has contributed to the creation of the SE ONE, as it is now.

MODES:

The SE-ONE has multiple modes:

MP3A: this mode is I/O identical to the SUNRISE MP3 player,
 there are differences in the software because the SE-ONE has another chipset and all of the functions convert to this chipset.
MP3B: mode: this mode you can play music files from USB stick
FM: The SE-ONE is an FM Radio
REG: Register functions , faster than AT commands
TEST: Test mode

AT commandos:

The protocol of the AT-command set has been used for a long time. AT-commands exist of a short string of text with added settings or parameters. You can also request data from the hardware with the AT-commands. You need a small program to use AT-commands that sends the string of text to the SE-ONE and processes the response of the SE-ONE. They are working on a machine language program that will

For quick reference help with AT commands, you can also use the /H after a command.
For example: AT+SEMODE/H

IO Settings modes: TODO

	MP3A	MP3B	REG	FM	Midi	Synth
Rd 0xFF20	At command	At command	At command	At command	At command	
Wr	At command	At command	At command	At command	At command	
Rd 0xFF21			Uart status		0xFF21	
Wr			Uart control		= Value	
Rd 0xFF22			RX uart Data		0xFF22	
Wr			TX uart Data		= Value	
Rd 0xFF23		Status byte *1	Status byte *1		0xFF23	
Wr			-		= Value	
Rd 0xFF24					0xFF24	
Wr					= Value	
Rd 0xFF25		Read Files data	Usb data		0xFF25	
Wr					= Value	
Rd 0xFF26		Vu left	Cmd response		0xFF26	
Wr			Cmd Function		= Value	
Rd 0xFF27		Vu Right	Cmd Command		Cmd Command	
Wr			Cmd Data		Cmd Data	

*1 status byte

bit 3 End of MUSIC playing
bit 4 USB Error
bit 5 USB busy
bit 6 Fifo buffer Full
bit 7 Fifo buffer emty

AT command reference:

CARTRIDGE

AT+CARTRIDGE

SEONE/M only MP3
SEONE/R only Radio
SEONE/MR MP3 and Radio

Get cartridge type

send: AT+CARTRIDGE?<13>
response: AT+CARTRIDGE=SEONE/MR<13><10> options: SEONE/M, SEONE/R, SEONE/MR
OK<13><10>

AT+SEMODE

MP3A = Sunrise MSXMP3
MP3B = TMTLOGIC MP3
REG = Register mode
TEST = Test mode

Change mode from SE-ONE

send: AT+SEMODE=FM<13> options: FM, MP3A, MP3B, REG, TEST
response: OK<13><10> options: OK, ERR, ILL

Get SE-ONE mode

send: AT+SEMODE?<13>
response: AT+SEMODE=FM<13> options: FM, MP3A, MP3B, REG, TEST
OK<13><10> options: OK, ERR, ILL

AT+SELOGBOOK

Get SE-ONE system logbook

send: AT+SELOGBOOK?<13>
response: AT+SELOGBOOK=TEXT<13> options: OK, ERR, ILL
OK<13><10>

AT+SEVERSION

1 = Radio
2 = MP3
3 = Radio and MP3

Get SE-ONE system SE-ONE version number

send: AT+SEVERSION?<13>
response: AT+SEVERSION=1<13> options: 1,2,3
OK<13><10> options: OK, ERR, ILL

AT+SEFIRMWARE

Get firmware dfu file name

send: AT+SEFIRMWARE?<13>
response: AT+SEFIRMWARE= SEONEddmmyyyy.DFU<13> options: SEONE[date].DFU
OK<13><10> options: OK, ERR, ILL

AT+SETUNE

Set startup tune

Set or Reset (ON/OFF) factory start-tune, this will be store in the ARM flash memory
When set OFF, the SE-ONE play the factory start-tune not when system startup.

send: AT+SETTUNE=ON<13> options: ON,OFF
response: OK<13><10> options: OK,ERR,ILL

AT+SEPLAYTUNE

Plays then intro tune from TMTLOGIC

send: AT+SEPLAYTUNE<13>
response: OK<13><10> options: OK,ERR

AT+SEFIRMWARECHK

Check if the firmware must be updated.

When firmware check is false, you must updating your SE-ONE !

send: AT+SEFIRMWARECHK=DDMMYYYY<13>
response: AT+SEFIRMWARECHK=TRUE <13> options: true, false
OK<13><10> options: OK,ERR,ILL

AT+SEFAULTNR

Get the actual error number.

send: AT+SEFAULTNR ?<13>
response: AT+ SEFAULTNR=12 <13><10> options: 0-29

0	"Succeeded	"
1	"Disk IO error	"
2	"Assertion failed	"
3	"The physical drive cannt work	"
4	"Could not find the file	"
5	"Could not find the path	"
6	"Invalid path name	"
7	"Access denied directory full	"
8	"Access denied	"
9	"Invalid file/directory	"
10	"Write protected	"
11	"Invalid logical drive	"
12	"The volume has no work area	"
13	"No valid FAT volume	"
14	"The f_mkfs() aborted	"
15	"Could not to access volume	"
16	"Rejected according sharing policy	"
17	"LFN could not be allocated	"
18	"Number of open files	"
19	"Invalid parameter	"
20	"Unknow at commando	"
21	"Syntax error	"
22	"Usb busy	"
23	"Invalid at command	"
24	"Usb already installed	"
25	"AT parameter error	"
26	"AT not in uppercase	"
27	"ARM flash error	"
28	"Invalid input	"
29	"Wrong SE mode	"

AT+SEFAULTTXT

Get the actual error text.

send: AT+SEFAULTTXT? <13>
response: AT+ SEFAULTTXT= Invalid input <13><10>

DSP Digital Sound Processor / Audio Processor

AT+DSPTYPE	Get the DSP chip type
AT+DSPVOLL	Set the left volume level
AT+DSPVOLR	Set the right volume level
AT+DSPBASS	Set the bass level
AT+DSPTREBLE	Set the treble volume level
AT+DSPVOLUP	Increase the volume by 10 steps
AT+DSPVOLDOWN	Decrease the volume by 10 steps
AT+DSPMUTE	Set DSP mute

AT+DSPTYPE

Get the DSP chip type

send: AT+DSPTYPE?<13>
response: AT+DSPTYPE=TDA8425<13> options: TDA8425
OK<13><10> options: OK,ERR,ILL

AT+DSPVOLL

Set the left volume level

send: AT+DSPVOLL=80<13> options: 0-100
response: OK<13><10> options: OK,ERR,ILL

Get the left volume level

send: AT+DSPVOLL?<13>
response: AT+DSPVOLL=66<13> options: 0-100
OK<13><10> options: OK,ERR,ILL

AT+DSPVOLR

Set the right volume level

send: AT+DSPVOLR=80<13>
 response: OK<13><10> options: 0-100
 options: OK,ERR,ILL

Get the left volume level

send: AT+DSPVOLR?<13>
 response: AT+DSPVOLR=66<13>
 OK<13><10> options: 0-100
 options: OK,ERR,ILL

AT+DSPBASS

Set the bass level

send: AT+DSPBASS=80<13>
 response: OK<13><10> options: 0-100
 options: OK,ERR,ILL

Get the bass level

send: AT+DSPBASS?<13>
 response: AT+DSPBASS=66<13>
 OK<13><10> options: 0-100
 options: OK,ERR,ILL

AT+DSPTREBLE

Set the treble volume level

send: AT+DSPTREBLE=80<13>
 response: OK<13><10> options: 0-100
 options: OK,ERR,ILL

Get the left volume level

send: AT+DSPTREBLE?<13>
 response: AT+DSPTREBLE=66<13>
 OK<13><10> options: 0-100
 options: OK,ERR,ILL

AT+DSPVOLUP

Increase the volume by 10 steps

send: AT+DSPVOLUP<13>
 response: OK<13><10> options: OK,ERR

AT+DSPVOLDOWN

Decrease the volume by 10 steps

send: AT+DSPVOLDOWN<13>
 response: OK<13><10> options: OK,ERR

AT+DSPMUTE

Set DSP mute

send: AT+DSPMUTE=ON<13>
 response: OK<13><10> options: ON,OFF
 options: OK,ERR,ILL

Get DSP mute state

send: AT+DSPMUTE?<13>
 response: AT+DSPMUTE=ON<13>
 OK<13><10> options: ON,OFF
 options: OK,ERR,ILL

MP3 DSP chip

AT+MP3TYPE Get the mp3 chip type
 AT+MP3LOWLEVEL Set the low-level bit from status register inp(&hFF23) bit 7
 AT+MP3HIGHLEVEL Set the high-level bit from status register inp(&hFF23) bit 6

AT+MP3TYPE

Get the mp3 chip type

send: AT+MP3TYPE?<13>
 response: AT+ MP3TYPE=VS1053<13>
 OK<13><10> options: VS1053

AT+MP3LOWLEVEL

Set the low-level bit from status register inp(&hff23) bit 7

Note: this works only in MP3B mode !

Bit 7 will be set when the content of the audio-stream FiFo lower is than this value

send: AT+MP3LOWLEVEL= 32<13>
 response: OK<13><10> options: 0-65535
 options: OK,ERR,ILL

AT+MP3HIGHLEVEL

Set the high-level bit from status register inp(&hff23) bit 6

Note: this works only in MP3B mode !

Bit 6 will be set when the content of the audio-stream FiFo higher is than this value.

send: AT+MP3HIGHLEVEL= 12232<13>
 response: OK<13><10> options: 0-65535
 options: OK,ERR,ILL

VU meter

AT+VULEFT Set the left VU byte
AT+VURIGHT Get the left VU byte
AT+VULN Get the left VU number
AT+VURN Get the right VU number

rserve at+vutype

AT+VULEFT

Set the left VU byte

send: AT+VULEFT=80<13> options: 0-255
response: OK<13><10> options: OK,ERR,ILL

Get the left VU byte

send: AT+VULEFT?<13>
response: AT+VULEFT=66<13> options: 0-255
 OK<13><10> options: OK,ERR,ILL

AT+VURIGHT

Get the left VU byte

send: AT+VURIGHT=80<13> options: 0-255
response: OK<13><10> options: OK,ERR,ILL

Get the left VU byte

send: AT+VURIGHT?<13>
response: AT+VURIGHT=66<13> options: 0-255
 OK<13><10> options: OK,ERR,ILL

AT+VULN

Get the left VU number

send: AT+VULN?<13>
response: AT+VULN=7<13> options: 0-8
 OK<13><10> options: OK,ERR,ILL

Get IO base: PRINT INP(&Hff26)

AT+VURN

Get the right VU number

send: AT+VURN?<13>
response: AT+VURN=7<13> options: 0-8
 OK<13><10> options: OK,ERR,ILL

Get IO base: PRINT INP(&Hff27)

USB universal serial bus

AT+USBINIT Install usb port
AT+USBDEVICE Get which USB device is connected

AT+USBINIT

Note: only in MP3B mode
Install usb port

When USB is good installed, the blue led is burn on the SE-ONE

send: AT+USBINIT<13>
response: OK<13><10> options: OK,ERR,ILL

AT+USBDEVICE

Note: only in MP3B mode

NONE: no device
MSD: Mass storge device (usb stick)

Get which USB device is connected

send: AT+USBDEVICE?<13>
response: AT+USBDEVICE =MSD<13> options: NONE,MSD
 OK<13><10> options: OK,ERR,ILL

MSD mass storage device (only MP3B & REG mode)

AT+MSDPATH Set the directory from the Mass Storage Device (usb stick)
AT+MSDFILES? Get the available files and directories in the current directory.
AT+MSDPLAY Play an music file from Mass Storage Device (usb stick)
AT+MSDPAUSE Pause playing
AT+MSDSTART Continue music playing
AT+MSDSTOP Stop playing
AT+MSDTALK Play a speech sample from usb /setalk

AT+MSDPATH

Note: only in MP3B mode

Set the directory from the Mass Storage Device (usb stick)

send: AT+MSDPATH=/MAP/MAP<13>
response: OK<13><10> options: OK,ERR,ILL

Get the directory from the Mass Storage Device (usb stick)

send: AT+MSDPATH?<13>
response: AT+MSDPATH=/MAP/MAP<13>
OK<13><10> options: OK,ERR,ILL

AT+MSDPATHUP

Set the directory above from the Mass Storage Device (usb stick) (dos syntax "..")

send: AT+MSDPATHUP<13>
response: OK<13><10>

AT+MSDFILES?

Note: only in MP3B mode

Get the available files and directories in the current directory.

send: AT+MSDFILES?<13>
response: OK<13><10> options: OK,ERR,ILL

for read the files run this program:

```
1000 D = INP(&HFF25)
1010 PRINT CHR$(D);
1020 IF D = 10 THEN END
1030 GOTO 1000
```

AT+MSDPLAY

Note: only in MP3B mode
This function works stand-alone in the SE_ONE

Play an music file from Mass Storage Device (usb stick)

send: AT+MSDPLAY=MUSIC.MP3<13>
response: OK<13><10> options: OK,ERR,ILL

AT+MSDPAUSE

Note: only in MP3B mode

Pause playing

send: AT+MSDPAUSE<13>
response: OK<13><10> options: OK,ERR,ILL

AT+MSDSTART

Note: only in MP3B mode

Continue music playing

send: AT+MSDSTART<13>
response: OK<13><10> options: OK,ERR,ILL

AT+MSDSTOP

Note: only in MP3B mode
Stop playing

send: AT+MSDSTOP<13>
response: OK<13><10> options: OK,ERR,ILL

AT+MSDTALK

Note: only in MP3B and REG mode
Play app switch to streaming mode.
Speech samples can be download from www.tmtlogic.com > support > sf3 > sd system files
Place the samples in \SETALK directory or the SD card

Play a speech sample from sd /setalk

send: AT+MSDTALK=44<13> 0-57 *
response: OK<13><10> options: OK,ERR,ILL

wait for usb 140 if (inp(&hFF23) AND 32) = 32) then goto 140 bit 5 USB_busy

see data sheet of the SP0256

<http://www.futurebots.com/spo256.pdf>

hex ??oct	word	time
00 000	PA1 PAUSE	10MS
01 001	PA2 PAUSE	30MS
02 002	PA3 PAUSE	50MS
03 003	PA4 PAUSE	100MS
04 004	PA5 PAUSE	200MS
05 005	/OY/ BOY	420MS
06 006	/AY/ Sky	260MS
07 007	/EH/ End	70MS
08 010	/KK3/ Comb	120MS
09 011	/PP/ Pow	210MS
0A 012	/JH/ Dodge	140MS
0B 013	/NN1/ Thin	140MS
0C 014	/IH/ Sit	70MS
0D 015	/TT2/ To	140MS
0E 016	/RR1/ Rural	170MS

0F 017	/AX/	Succeed	70MS	
10 020	/MM/	Milk		180MS
11 021	/TT1/	Part		100MS
12 022	/DH1/	They		290MS
13 023	/IY/	See		250MS
14 024	/EY/	Beige		280MS
15 025	/DD1/	Could		70MS
16 026	/UW1/	To		100MS
17 027	/AO/	Aught		100MS
18 030	/AA/	Hot		100MS
19 031	/YY2/	Yes		180MS
1A 032	/AE/	Hat		120MS
1B 033	/HH1/	He		130MS
1C 034	/BB1/	Business	80MS	
1D 035	/TH/	Thin		180MS
1E 036	/UH/	Book		100MS
1F 037	/UW2/	Food		260MS
20 040	/AW/	Out		370MS
21 041	/DD2/	Do		160MS
22 042	/GG3/	Wig		140MS
23 043	/VV/	Vest		190MS
24 044	/GG1/	Got		80MS
25 045	/SH/	Ship		160MS
26 046	/ZH/	Azure		190MS
27 047	/RR2/	Brain		120MS
28 050	/FF/	Food		150MS
29 051	/KK2/	Sky		190MS
2A 052	/KK1/	Can't		160MS
2B 053	/ZZ/	Zoo		210MS
2C 054	/NG/	Anchor		220MS
2D 055	/LL/	Lake		110MS
2E 056	/WW/	Wool		180MS
2F 057	/XR/	Repair		360MS
30 060	/WH/	Whig		200MS
31 061	/YY1/	Yes		130MS
32 062	/CH/	Church		190MS
33 063	/ER1/	Fir		160MS
34 064	/ER2/	Fir		300MS
35 065	/OW/	Beau		240MS
36 066	/DH2/	They		240MS
37 067	/SS/	Vest		90MS
38 070	/NN2/	No		190MS
39 071	/HH2/	Hoe		180MS
3A 072	/OR/	Store		330MS
3B 073	/AR/	Alarm		290MS
3C 074	/YR/	Clear		350MS
3D 075	/GG2/	Guest		40MS
3E 076	/EL/	Saddle		190MS
3F 077	/BB2/	Business	50MS	

AT+MSDSETSAMPLE

Note: only in MP3B and REG mode
Store sample filename in memory 0-9.
Place the samples in \SAMPLE directory

Play a speech sample from usb /setalk

send: AT+MSDSETSAMPLE=1@TOETER.MP3<13>
response: OK<13><10> options: OK,ERR,ILL

AT+MSDSAMPLE

Note: only in MP3B and REG mode
Play app switch to streaming mode.
Play sample from USB/SAMPLE
Place the samples in \SAMPLE directory

Play a speech sample from usb /setalk

send: AT+MSDSAMPLE=1<13> 0-9
response: OK<13><10> options: OK,ERR,ILL

REG register mode (beta)

Before test this mode run this command :AT+SEMODE=REG

(AT commands are active in REG mode)

I/O port setting see begin of this manual

CALL ARM response

```
1000 I = INP(&hFF26)
1010 IF I = 1 THEN GOTO 1010
1020 IF I = 2 THEN PRINT "Error" : end
1030 RETURN
```

Audio DSP

F48	cmdDspVolUp	17 sept 2018
F49	cmdDspVolDown	17 sept 2018
F50	cmdSetVolLeftRight	17 sept 2018
F51	cmdGetVolLeftRightStereo	18 sept 2018
F52	cmdSetBassTreableStereo	18 sept 2018
F53	cmdGetBassTreableStereoMute	18 sept 2018

F48 **cmdDspVolUp** 17 sept 2018
100 out &hFF26,48 activate function

	130 gosub ARM response	
F49	cmdDspVolDown	17 sept 2018
	100 out &hFF26,49	activate function
	130 gosub ARM response	
F50	cmdSetVolLeftRight	17 sept 2018
	value 0-100d	
	100 out &hFF26,0	
	110 out &hFF27,45 volume left	
	120 out &hFF27,55 volume Right	
	130 out &hFF26,50	
	140 gosub ARM response	
F51	cmdGetVolLeftRightStereo	18 sept 2018
	100 out &hFF26,51	
	110 gosub ARM response	
	120 print inp(&hFF27)	
	130 print inp(&hFF27)	
F52	cmdSetBassTreableStereo	18 sept 2018
	bass,treable value 0-100d	
	stereo type value 0-3	
	0 = Fored mono	
	1 = linear Stereo	
	2 = pseudo Stereo	
	3 = spatial Stereo	
	mute value 0-1	
	100 out &hFF26,0	
	110 out &hFF27,45	bass
	120 out &hFF27,55	treable
	130 out &hFF27,2	
	140 out &hFF27,1 Mute	Stereo type
	150 out &hFF26,52	
	160 gosub ARM response	
F53	cmdGetBassTreableStereoMute	18 sept 2018
	100 out &hFF26,53	
	110 gosub ARM response	
	120 print inp(&hFF27)	
	130 print inp(&hFF27)	
	140 print inp(&hFF27)	
	150 print inp(&hFF27)	

VU

F64	cmdSetVuEnable	18 sept 2018
F66	cmdSetVuLeftRight	18 sept 2018
F67	cmdGetVuLeftRight	18 sept 2018
F68	cmdSetVuMaxLeds	18 sept 2018
F69	cmdGetVuLeftRightNumber	18 sept 2018
F70	cmdGetEqualizer (todo)	19 sept 2018
F64	cmdSetVuEnable	18 sept 2018
	0 = OFF	
	1 = ON	
	100 out &hFF26,0	
	110 out &hFF27,1	= ON
	120 out &hFF26,64	
	130 gosub ARM response	
F66	cmdSetVuLeftRight	18 sept 2018
	100 out &hFF26,0	
	110 out &hFF27,1 left byte	
	120 out &hFF27,1 right byte	
	130 out &hFF26,66	
	140 gosub ARM response	
F67	cmdGetVuLeftRight	18 sept 2018
	100 out &hFF26,67	
	110 gosub ARM response	
	120 print inp(&hFF27)	Vu left byte leds
	130 print inp(&hFF27)	Vu right byte leds
F68	cmdSetVuMaxLeds	18 sept 2018
	value is 1-32	
	100 out &hFF26,0	
	110 out &hFF27,8 Default 8	
	120 out &hFF26,68	
	130 gosub ARM response	
F69	cmdGetVuLeftRightNumber	18 sept 2018
	is the same as in MP3B inp(&hFF26) & inp(&hFF27)	
	value is: 0 - 8	
	100 out &hFF26,69	
	110 gosub ARM response	

```

120 print inp(&hFF27)          Vu left  n leds
130 print inp(&hFF27)          Vu right n leds

F70      cmdGetEqualizer  (todo)          19 sept 2018

Not for FM radio sound

100 out &hFF26,&hFF46          function cmdGetEqualizer
110 gosub ARM response

todo      16x print inp(&hFF27)
          print inp(&hFF27)          Vu right n leds

=====
FAT32
reserve 50-5F

case 50:  addsCmdInitFatfs();
case 51:  addsCmdReadSector();
case 52:  addsCmdWriteLock();
case 53:  addsCmdWriteSector();
case 54:  addsCmdFatfsInfo();

=====
SYSTEM:

60h      cmdGetFirmwareVersion          18 sept 2018

Attention! another structure then AT+SEFIRMWARE !

YEAR
YEAR
YEAR
YEAR
MOUNT
MOUNT
DAY
DAY

out &hFF26,&hFF60          function cmdGetFirmwareVersion

inp(&hFF26)          0=idle, 1= busy, 2=Error
wait for ilde..
while(inp(&hFF26) == 1);

print chr$(inp(&hFF27))    2
print chr$(inp(&hFF27))    0
print chr$(inp(&hFF27))    1
print chr$(inp(&hFF27))    8
print chr$(inp(&hFF27))    0
print chr$(inp(&hFF27))    9
print chr$(inp(&hFF27))    1
print chr$(inp(&hFF27))    8
print inp(&hFF27)          <0>

BASIC example Firmware check

10 OUT &hFF26,&hFF60
20 IF INP(&hFF26) = 1 THEN GOTO 20
21 IF INP(&hFF26) = 2 THEN PRINT"ERROR":END
30 F$ = ""
40 I = INP(&hFF27)
50 IF I = 0 THEN GOTO 80
60 F$=F$+CHR$(I)
70 GOTO 40
80 C = 20170909      :rem Year Mount Day
90 F = VAL(F$)
99 IF F < C THEN PRINT"Update your SE-ONE" :END

=====
USB MSD:

70h      cmdUsbMsdInit          18 sept 2018

out &hFF26,&h70          function cmdUsbMsdInit

inp(&hFF26)          0=idle, 1= busy, 2=Error
wait for ilde..
while(inp(&hFF26) == 1);

72h      cmdUsbMsdAvailable

74h      cmdSetMSDPLAY          19 sept 2018

out &hFF26,0
out &hFF27,asc("F")
out &hFF27,asc("I")
out &hFF27,asc("L")
out &hFF27,asc("E")
out &hFF27,asc("N")
out &hFF27,asc("A")
out &hFF27,asc("M")
out &hFF27,asc("E")
out &hFF27,asc(".")
out &hFF27,asc("B")
out &hFF27,asc("A")
out &hFF27,asc("S")

out &hFF26,&h72          Function cmdSetMSDPLAY

inp(&hFF26)          0=idle, 1= busy, 2=Error
wait for ilde..
while(inp(&hFF26) == 1);

```

75h	cmdGetMSDPLAY	19 sept 2018
	out &hFF26,&h75	function cmdGetMSDPLAY
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
	print chr\$(inp(&hFF27)) F print chr\$(inp(&hFF27)) I print chr\$(inp(&hFF27)) L print chr\$(inp(&hFF27)) E print chr\$(inp(&hFF27)) N print chr\$(inp(&hFF27)) A print chr\$(inp(&hFF27)) M print chr\$(inp(&hFF27)) . print chr\$(inp(&hFF27)) B print chr\$(inp(&hFF27)) A print chr\$(inp(&hFF27)) S print inp(&hFF27) <0>	
76h	cmdSetMSDPATH	19 sept 2018
	out &hFF26,0 out &hFF27,asc("&hFF27") out &hFF27,asc("M") out &hFF27,asc("A") out &hFF27,asc("P") out &hFF26,&h76	Function cmdSetMSDPATH
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
77h	cmdGetMSDPATH	19 sept 2018
	out &hFF26,&h77	function cmdGetMSDPATH
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
	print chr\$(inp(&hFF27)) \ print chr\$(inp(&hFF27)) M print chr\$(inp(&hFF27)) A print chr\$(inp(&hFF27)) P print inp(&hFF27) <0>	
78h	cmdMsdStart	19 sept 2018
	out &hFF26,&h78	Function cmdMsdStart
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
79h	cmdMsdStop	19 sept 2018
	out &hFF26,&h79	Function cmdMsdStop
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
7Ah	cmdMSDPAUSE	19 sept 2018
	out &hFF26,&h79	Function cmdMsdPause
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
7Bh	cmdMSDFILES	19 sept 2018
	8 characters + ' . ' + 3 characters + <TAB> out &hFF26,&h7B	Function cmdMsdFiles
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
	10 i = inp(&hFF25) 20 if i = 0 then end 30 print chr\$(i); 40 goto 10	note: 25h !
	f i l e , m p 3 <32> <32> <32> <32> <9> f i l e n	
7Ch	cmdGetMSDSIZE	19 sept 2018
	Gives the size of the file after play commando	
	32 bits binair value	
	out &hFF26,&h77	function cmdGetMSDPATH
	inp(&hFF26) wait for ilde.. while(inp(&hFF26) == 1);	0=idle, 1= busy, 2=Error
	print chr\$(inp(&hFF27)) \ print chr\$(inp(&hFF27)) M print chr\$(inp(&hFF27)) A print chr\$(inp(&hFF27)) P print inp(&hFF27) <0>	
7Dh	cmdGetMSDCNT	19 sept 2018

F126 cmdSetMSDTALK

value 0 - 58

F127 **cmdSetMSDSETSAMPLE**

value 0 - 9

F128 cmdSetMSDSAMPLE

value 0 - 9

Examples

How to use AT commands:

AT.BAS (CPC)

How to play form USB stick?

How to get the files list form USB stick?

```
AT+SEMODE=MP3B
AT+USBINIT          The blue led must burn, than the USB port is active
AT+MSDFILES?
```

BASIC:

for read the files run this program:

```
1000 D = INP(&hFF25)
1010 PRINT CHR$(D);
1020 IF D = 10 THEN END
1030 GOTO 1000
```

Troubleshooting

No Sound

```
AT+DSPVOLL=100
AT+DSPVOLR=100
```

USB don't work

After mode MP3B and USBINIT the blue led must be burn on, otherwise reset the SE-ONE and try again.

The usb stick is not good formatted, format it with FAT or FAT32

Appendix:

Symbol/ace III conversion table:														
CPC/BASE SF3			ENTERPRISE			PCW			MSX					
SF3 SEONE	&FD40 - FD4F &FF20 - &FF27		&40 - &4F &20 - &27			&40B0 - &4FB0 &20B0 - &27B0			&40 - &4F &20 - &27					
PinName	PinNumber	CPC >> SF3	PinName	PinNumber	ENT >> SF3	PinName	PinNumber	PCW >> SF3	PinName	PinNumber	PCW >> MSX			
Z80_A0	18	O	Z80_A0	a10	O	Z80_A0	23	O	Z80_A0	28	O			
Z80_A1	17	O	Z80_A1	b10	O	Z80_A1	24	O	Z80_A1	27	O			
Z80_A2	16	O	Z80_A2	a11	O	Z80_A2	21	O	Z80_A2	30	O			
Z80_A3	15	O	Z80_A3	b11	O	Z80_A3	22	O	Z80_A3	29	O			
Z80_A4	14	O	Z80_A4	a12	O	Z80_A4	19	O	Z80_A4	32	O			
Z80_A5	13	O	Z80_A5	b12	O	Z80_A5	20	O	Z80_A5	31	O			
Z80_A6	12	O	Z80_A6	a13	O	Z80_A6	17	O	Z80_A6	22	O			
Z80_A7	11	O	Z80_A7	b13	O	Z80_A7	18	O	Z80_A7	21	O			
Z80_A8	10	O	Z80_A8	a6	O	Z80_A8	15	O	Z80_A8	24	O			
Z80_A9	9	O	Z80_A9	b6	O	Z80_A9	16	O	Z80_A9	17	O			
Z80_A10	8	O	Z80_A10	a7	O	Z80_A10	13	O	Z80_A10	20	O			
Z80_A11	7	O	Z80_A11	b7	O	Z80_A11	14	O	Z80_A11	19	O			
Z80_A12	6	O	Z80_A12	a8	O	Z80_A12	11	O	Z80_A12	23	O			
Z80_A13	5	O	Z80_A13	b8	O	Z80_A13	12	O	Z80_A13	26	O			
Z80_A14	4	O	Z80_A14	a9	O	Z80_A14	9	O	Z80_A14	25	O			
Z80_A15	3	O	Z80_A15	b9	O	Z80_A15	10	O	Z80_A15	18	O			
Z80_D0	26	BI	Z80_D0	a14	BI	Z80_D0	31	BI	Z80_D0	34	BI			
Z80_D1	25	BI	Z80_D1	b14	BI	Z80_D1	32	BI	Z80_D1	33	BI			
Z80_D2	24	BI	Z80_D2	a15	BI	Z80_D2	29	BI	Z80_D2	36	BI			
Z80_D3	23	BI	Z80_D3	b15	BI	Z80_D3	30	BI	Z80_D3	35	BI			
Z80_D4	22	BI	Z80_D4	a16	BI	Z80_D4	27	BI	Z80_D4	38	BI			
Z80_D5	21	BI	Z80_D5	b16	BI	Z80_D5	28	BI	Z80_D5	37	BI			
Z80_D6	20	BI	Z80_D6	a17	BI	Z80_D6	25	BI	Z80_D6	40	BI			
Z80_D7	19	BI	Z80_D7	b17	BI	Z80_D7	26	BI	Z80_D7	39	BI			
Z80_nM1	29	O	Z80_nM1	a20	O	Z80_nM1	34	O	Z80_nM1	9	O			
Z80_nIORQ	31	O	Z80_nIORQ	b3	O	Z80_nIORQ	42	O	Z80_nIORQ	11	O			
Z80_nMREQ	28	O	Z80_nMREQ	a5	O	Z80_nMREQ	40	O	Z80_nMREQ	12	O			
Z80_nRFSH	30	O	Z80_nRFSH	a2	O	Z80_nRFSH		O	Z80_nRFSH	6	O			
Z80_nvR	33	O	Z80_nvR	b2	O	Z80_nvR	39	O	Z80_nvR	13	O			
Z80_nRD	32	O	Z80_nRD	a3	O	Z80_nRD	41	O	Z80_nRD	14	O			
Z80_nINT	35	I	Z80_nINT	b18	I	Z80_nINT	36	I	Z80_nINT	8	I			
Z80_nNMI	36	I	Z80_nNMI	b5	I	Z80_nNMI		I						
Z80_nvREADY	39	I	Z80_nvWAIT	a19	I	Z80_nvWAIT	38	I	Z80_nvWAIT	7	I			
Z80_nRESET	41	O	Z80_nRESET	a18	O	Z80_nRESET	33	O	Z80_nRESET	15	O			
Z80_CLK	50	O	ENT_A21	a29	O			O	MSX_CLK	42	O			
CPC_ROMDIS	43	O	ENT_A17	a27	O	nBUSRQ	35	I	MSX_nSLTSL	4	O			
CPC_RAMDIS	45	O	ENT_A19	a28	O	nBUSAK	37	O	MSX_BUSDIR	10	O			
CPC_nRAMPD	44	O	ENT_A18	b27	O	nMDIS	45	I	MSX_nCS1	1	O			
CPC_EXP	48	I	ENT_A20	b28	O	32Mhz	47	O	MSX_nCS2	2	O			
CPC_ROMEN	42	O	ENT_A16	b26	O	4Mhz	48		MSX_nCS12	3	O			
CPC_nBUSRES	40	I												
SV0	27	O	SV0	a4.b4(direct+3v a33.b33	O	SV0	5,6	O	SV0	45,47	O			
GND	2	O	GND	b19.b20.b21.b22.b23	O	GND	3,4,49,50	O	GND	41,43	O			
GND	49	O	GND		O			O			O			
TP3		O	BUSDIR	2.1:TP3 2.1f->:internally	O			O			O			
									MSX_SW1		TO SW2			
									MSX_SW2		TO SW1			