MyNOR.org

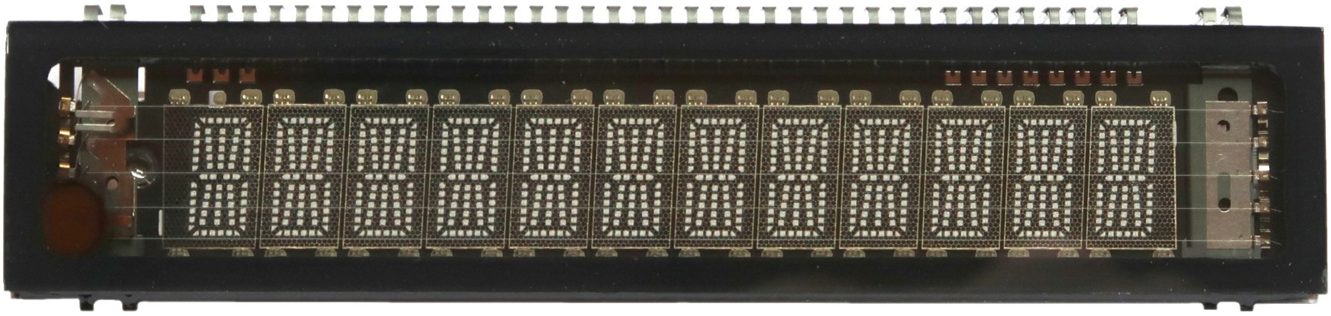
VFD Extension Board Construction Manual

Dennis Kuschel
dennis_k@freenet.de
2021-01-08

A Vacuum Fluorescent Display for MyNOR

VFD tubes have long been used in consumer electronics for displays. They are vacuum tubes that can be seen as a kind of cross between a triode and a low-voltage CRT. Unlike a CRT tube, it operates at very low voltages around 25V, and there are no magnets to deflect the electron beam. Instead, the electrons which are emitted from a heated wire and accelerated through a control grid, are directed to an actively switched anode. This anode is coated with phosphor, which emits light when the electrons hit it. The control grid can be used to switch an entire segment on or off, so that the display segments can be easily multiplexed.

I am using a 12 character wide VFD from Samsung. The part number of the display is “HCS-12SS59T”, it can be obtained from various places on the internet. I bought mine from “Pollin”, a residual stock dealer in Germany:



You need at least MyNOR ROM version 1.2 to be able to operate this display with my provided application programs. With this or a later ROM you can use the VFD display board as an alternative for the MyNOR Calculator. This gives the MyNOR Calculator a much more “vintage” look.

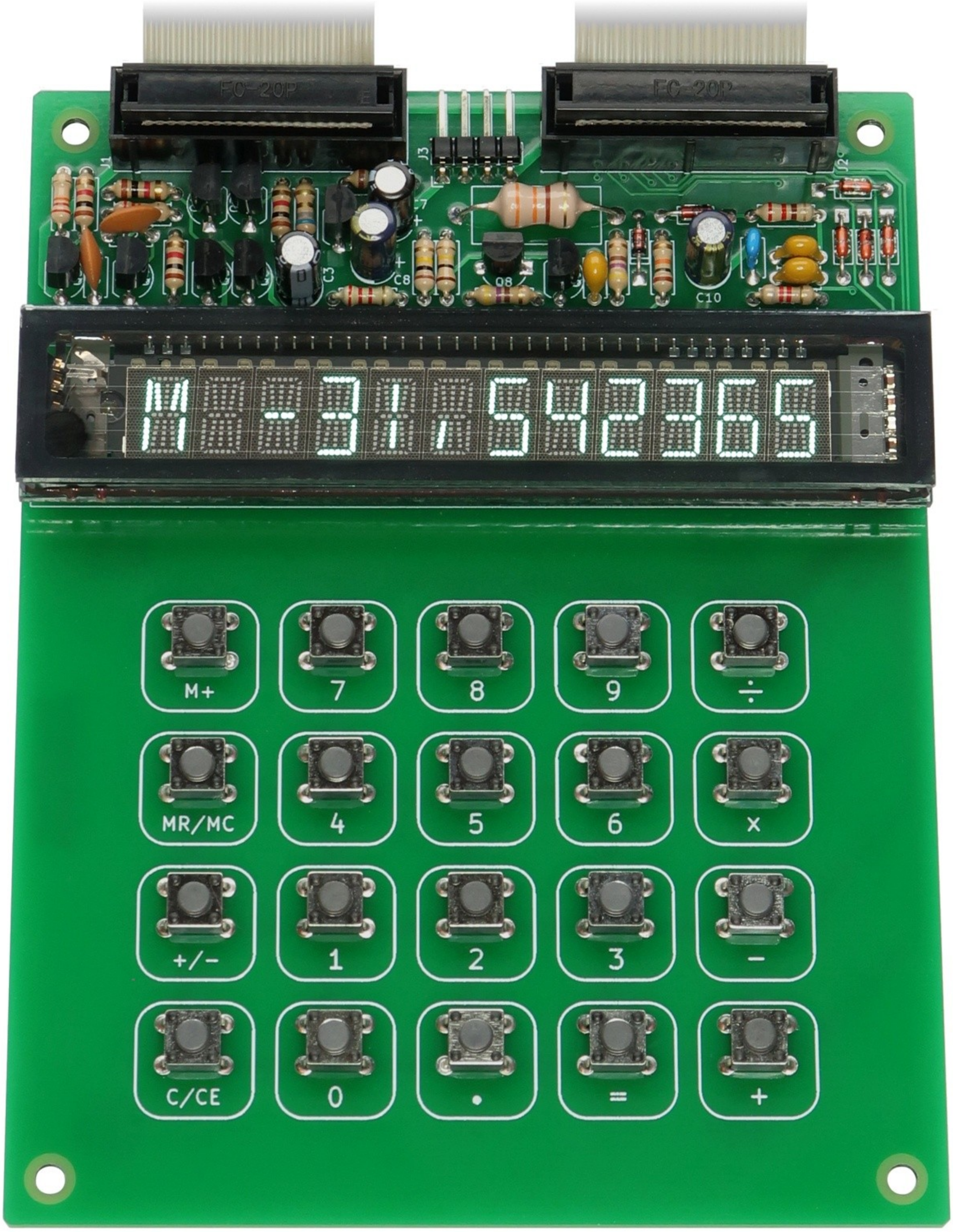
VFD Technical Data:

Interface	SPI with CLK/MOSI/CSn (LSB first) and RSTn
Power Supplies	V_{DD} 4.5V - 5.5V, V_{EE} 30.6V - 37.4V, Filament 3.24V - 3.96V
Display	12 characters, 14 segments per character
Character Set	numbers, upper case letters and special characters plus 16 user definable characters (the ASCII characters 0x20 - 0x5F can be mapped to the VFD character set)

There is a lot of information about this display on the Internet. I downloaded the data sheet at www.pollin.de. You can also take a look at <https://github.com/qrti/VFD-HCS-12SS59T> for more information.

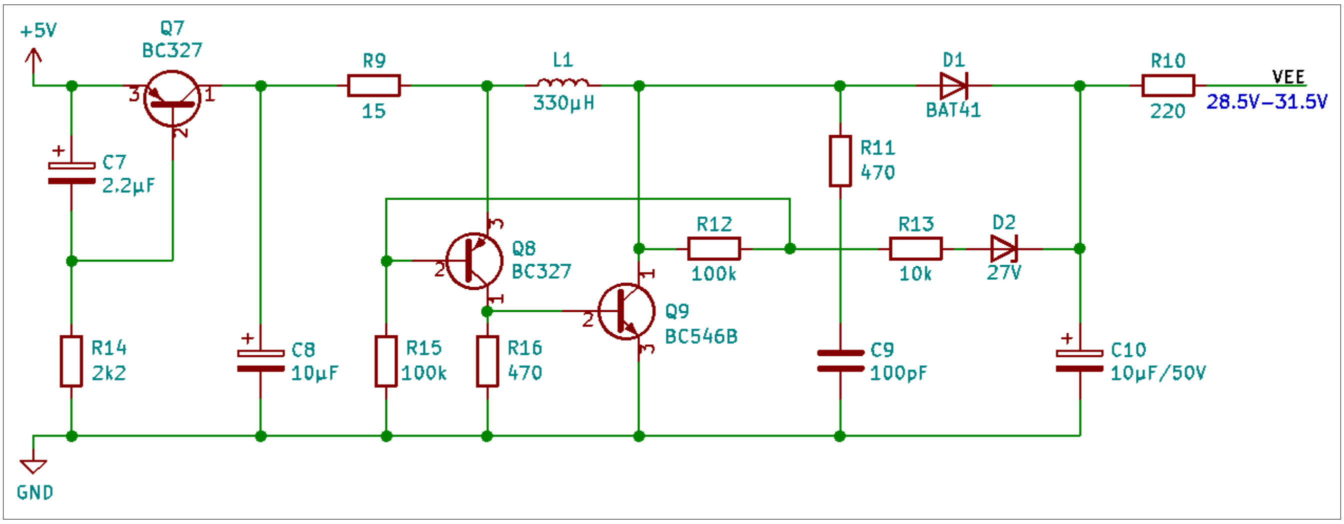
This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License: https://creativecommons.org/licenses/by-sa/4.0/

Here is a picture of the fully assembled VFD board. Please note the vintage looking electronics above the vacuum display tube. These electronics are required for two purposes: First, this vacuum tube needs a “high” voltage for the anode. The voltage converter generates about +29V from the 5V supply. Second, the cathode wires need an AC voltage of approx. 3.6V for heating. This voltage should be alternating to avoid an destructive effect that is known as “cathode poisoning”.



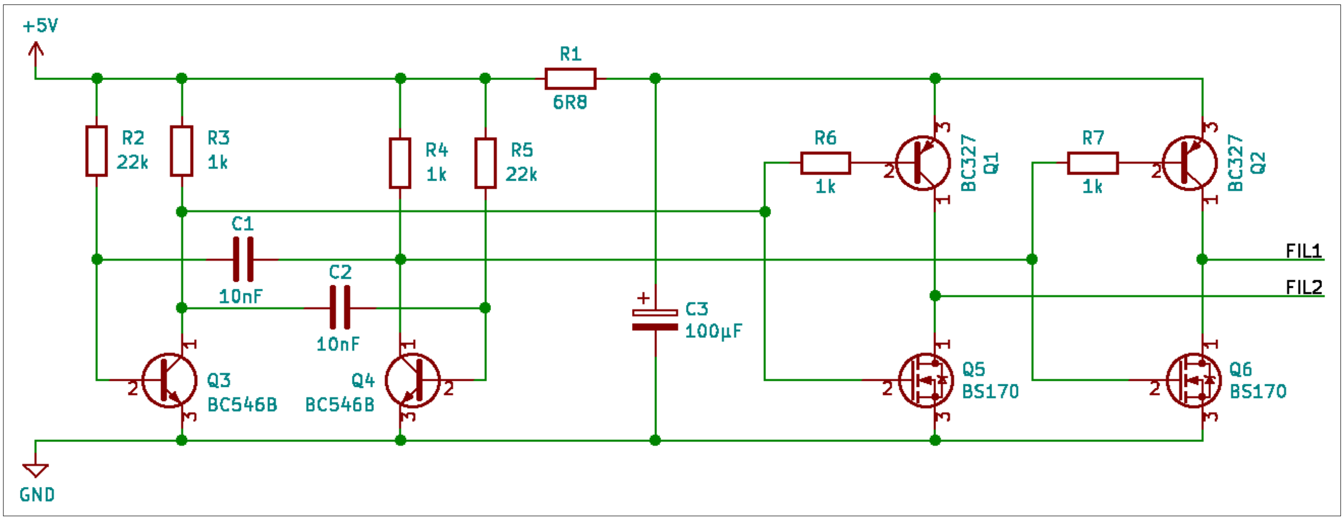
The High Voltage Converter

The anode voltage is generated by a very simple two-transistor boost converter. The converter is built up with the transistors Q8 and Q9, surrounded by the associated components L1, R12, R15, R16, D1 and C10. The diode D2, together with the resistor R13, prevents the output voltage from getting higher than about 30 V. The circuit oscillates at a frequency between 45kHz and 55kHz, with the frequency depending on the actual load. To reduce HF noise, the snubber network R11 / C9 was added to the circuit. R9 is used to limit the peak current through Q9, and C8 buffers the input voltage. And finally Q7, C7 and R14 form a delay circuit that ensures that the power-up sequence of the display is met (V_{EE} must be switched on after V_{DD}).



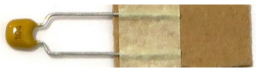
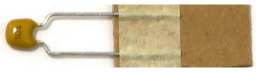
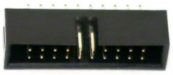
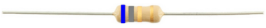
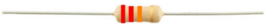
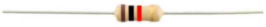
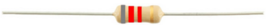
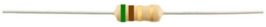
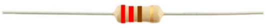
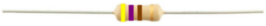
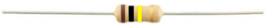
The Cathode Filament Supply

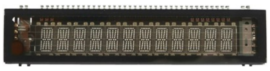
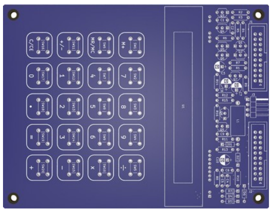
As in any other tube, the electrons in the VFD are emitted from a slightly glowing filament. To avoid “cathode poisoning”, the current through the filament should be alternating. The Samsung VFD requires a filament voltage of 3.6V ±10% at a current of typically 125mA. The circuit below provides a square-wave voltage with a frequency of about 3.4kHz between the connection points FIL1 and FIL2. The oscillator is built up with Q3 and Q4 and the surrounding components. Q1, Q2, Q5 and Q6 form a full bridge circuit to drive the filament. R1 is used to filter the input voltage (or better, prevent the input voltage rail from being disturbed by high current spikes that can occur when the transistors of each half-bridge are both conducting for a very short time). R1 also reduces the voltage across the filament. You may need to change the value of the resistor if the voltage across the filament is higher than 4V.



Required Components

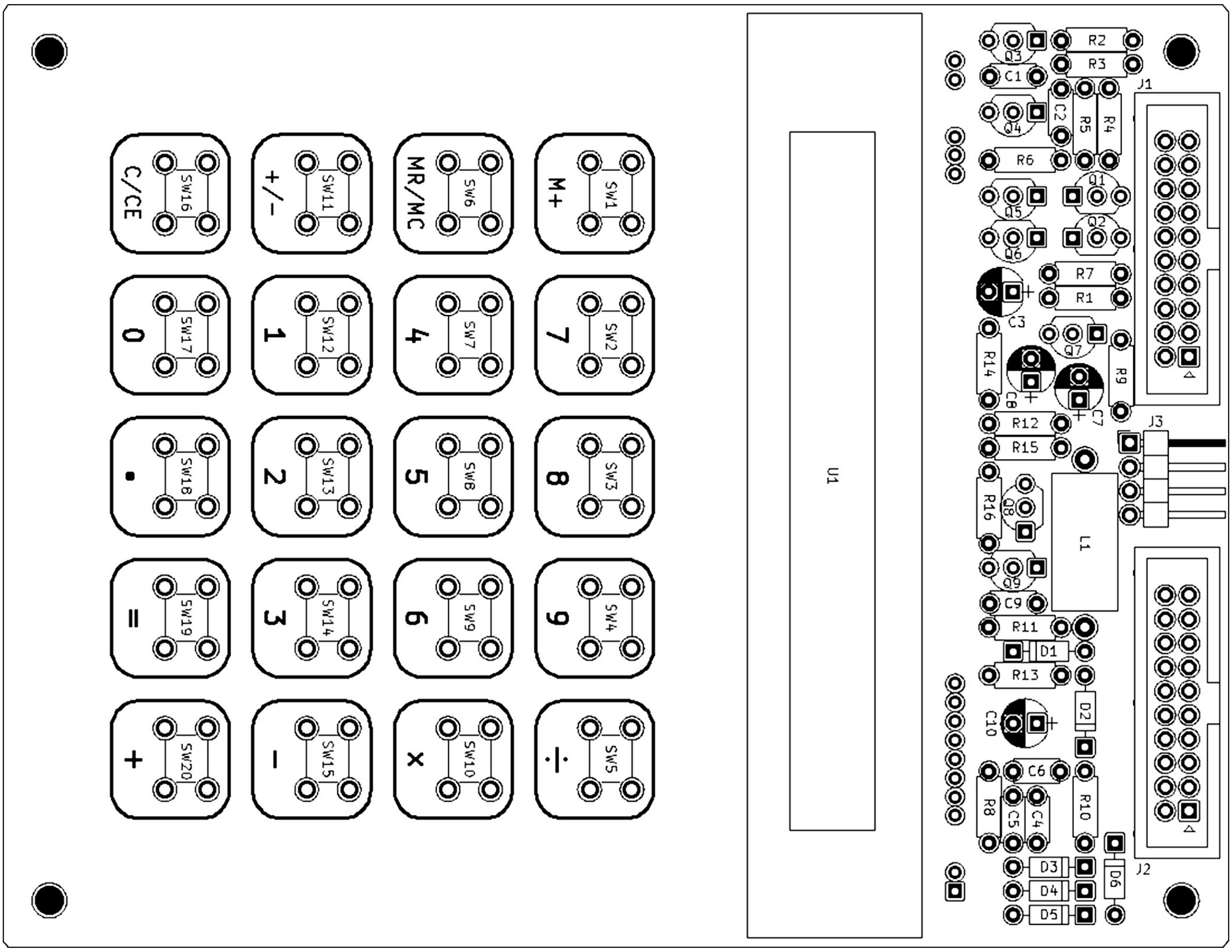
The complete bill of material is listed in the table below. Many of the Mouser part numbers can also be used to order the parts at Digikey or other distributors.

Reference	Qty	Picture	Value	Mouser P/N www.mouser.com	Reichelt P/N www.reichelt.de
C1 C2	2		10 nF (X7R / 5 mm)	SR215C103K	X7R-5 10N
C3	1		100 µF / 16V (2.5 mm)	ESK107M016AE3KA	RAD FC 100/16
C4 C5	2		100 nF (X7R / 5 mm)	SR215C104K	X7R-5 100N
C6	1		82 pF (5mm)	FG28C0G2A820JNT00	KERKO-500 82P
C7	1		2.2µF / 16V (2.5 mm)	UVP2A2R2MED	RAD FC 2,2/50
C8 C10	2		10 µF / 50V (2.5 mm)	ESH106M100AE3AA	RAD FC 10/50
C9	1		100 pF (5 mm)	FG28C0G2A101JNT06	KERKO-500 100P
D2	1		Z-Diode 27V	1N5254B	ZF 27
D1 D3 D4 D5 D6	5		BAT41	BAT41-TAP	BAT 41
J1 J2	2		20 pin header	710-61202021621	WSL 20G
J3	1		4 pin header right-angled	22-28-8043	SL 1X36W 2,54
L1	1		330 µH (min 0.3 A)	B82144A2334J000	L-HBCC 330µ
Q1 Q2 Q7 Q8	4		BC327	BC32725TA	BC 327-25
Q3 Q4 Q9	3		BC546B	BC546BTA	BC 546B
Q5 Q6	2		BS170	BS170D27Z	BS 170
R1	1		6.8 Ohm	CFR-25JR-52-6R8	1/4W 6,8
R2 R5	2		22k	CFR-25JR-52-22K	1/4W 22K
R3 R4 R6 R7	4		1k	CFR-25JR-52-1K	1/4W 1,0K
R8	1		8.2k	CFR-25JR-52-8K2	1/4W 8,2K
R9	1		15 Ohm	CFR-25JR-52-15R	1/4W 15
R10	1		220 Ohm	CFR-25JR-52-220R	1/4W 220
R11 R16	2		470 Ohm	CFR-25JR-52-470R	1/4W 470
R12 R15	2		100k	CFR-25JR-52-100K	1/4W 100K

R13	1		10k	CFR-25JR-52-10K	1/4W 10K
R14	1		2.2k	CFR-25JR-52-2K2	1/4W 2,2K
SW1 - SW20	20		SW_Push 6x4.3mm	TL1105BF160Q	TASTER 3301
U1	1		Samsung HCS-12SS59T	(not available)	(not available)
	4		Spacer 10mm	970100354	DI 10MM
	4		Spacer 15 mm	971150354	DA 15MM
	4		Screw M3 x 8	RM3X8MM-2701	SZK M3X8-200
PCB Raw Card	1		Use provided gerber files (in zip file) and order the PCB at jlcpcb.com		

Board Assembly

The picture below shows the position of each part. Start with soldering the low components (resistors, diodes, buttons and J3). After that, continue with the transistors, the inductor and the capacitors. Finally mount the display and the headers J1 and J2. The display can be fixed with a little hot glue or with a strip of 1.6 mm thick PCB material glued with two-components adhesive.



Software

MyNOR’s operating system does not have built-in support for the VFD (although I updated the ROM to version 1.2 along with the release of this document). This is because there is no space left in the ROM for drivers for new extension boards. The little free space in the ROM is reserved for important improvements or bug fixes. So the driver for the VFD must be part of the application program that is loaded via RS232 or from the EEPROM.

The VFD Calculator

The first and most obvious application I wrote for the VFD board is the calculator. To use this board as calculator, only a small application program must be uploaded to MyNOR. I had to export more API functions from MyNOR’s operating system so that an application program can use the existing calculator code in the ROM. Therefore you must have ROM v1.2 installed to run my applications.

```
@@@@@@:VFD Calculator@@@@*O5A0P7P6X0@1B4@0@1@2D2X3D2@0@0@0@0@0@0@0@2A2B2C2D2E2
F2G2H2I2Q0R0S0T0U0]1C0R0Y0F7[1Y0A4C4A0L0_4A0M0B4Y0A2^1Y0G2^1C0P7N0\0F0\0]0@5A0
L2@0A0Q0O7A0P0[7D0\0M0Q0]0@5C0E0A0O0@0P0P0F0P0]0@0A0E0X7M0E0S0X7W0@4B4K0Q0V0E2
B4A0P0^0]0@0L0N0W0W4B4J0P0U0P0W0H3B4U0L2V0_1B4J0L2X0_1B4I0P0K0N0V0@4B4U0L2A0L2
F0W0_1B4D0P0Y0J2^1X0_1B4C0Q3Y0[1C4X0W2^1A0K2P1A0J2P1U0R0V0M5B4A0K2]0A0T0I2A0U0
@4A0Q0I0E0J0L0S0@4V0R6B4Y0C0]1A0E0P0N0E0F0L0A0M0B4E0Y0\0]1X0T6B4C0P1Y0I1C4J0Q0
D0Q0T0S0W0E7B4C0@0Y0I1C4U0Q0W0V5B4D0I2S0U0W0_7B4C0A0F0H2F0G2A0F2B0F0E2X0]0C4S0
]1W0]0C4A0I2P1A0L0H2A0M0@4E0S0P1W0X0C4J0L0X0M0C4I0L0C0]1G0A0L0@2A0M0@4Y0\2C4X0
Y6[1[0L0[0M0B0L0T0B0M0U0G0J0T0\0M0\0L0Z0[0J0[0K0F0P0Y0X7E1Y0P0F1Y0D0F1X0Y6[1A0
L0C0A0M0B4A0P0@1A0Q0F0X0B3C4A0P0P0A0Q0L0[0J0[0K0Y0X7E1Y0P0F1Y0T4\1F0P0Y0P0F1J0
Q0U0Q0W0L3C4Y0D0F1X0Y6[1[0J0[0K0Y0G7E1D0^0A0E0O7M0E0]0@7A0E0P0N0E0]0@7F0^0C0L3
Y0[1C4C0_2Y0[1C4Y0M2C4C0P3Y0[1C4X0Y6[1
```

You can get the source code of this program from the software section of my website.

The Boot Splash Screen

When you have a larger application for the VFD that needs to be loaded from the EEPROM, you may want to get a “quick” feedback from MyNOR when you turn it on. I have written a very small program that simply shows “BOOTING” on the display and then loads the actual application from the EEPROM. The application must follow directly behind this “boot loader” in the EEPROM. For example, you can store this boot loader on storage location 7 and the application on location 8.

```
@@@@@@:VFD Bootloader@@@@*J3@0@3Z4[0N0Y0G7E1D0^0A0E0O7M0E0]0@7A0E0P0N0E0]0@7F0
^0C0L3Y0L2B4C0_2Y0L2B4C0P3Y0L2B4A0L0^2A0M0B4Y0X7E1A0P0P0Y0P0F1Y0T4\1F0P0Y0P0F1
D0L0S0J3W0U1B4Y0D0F1\0P0X0M2^1[0J0[0K0F0P0Y0X7E1Y0P0F1Y0D0F1X0Y6[1P1P1P1P1P1W0
^0Y0D1_0_0R0
```

Autostart

You may want MyNOR to automatically start the calculator program when MyNOR is turned on. You can accomplish this by simply configuring the calculator application program to autostart (menu point 7 in MyNOR’s main menu). But it may be better to let MyNOR start the calculator program only when the VFD board is connected. This can be accomplished by setting one of the two jumpers on the backside of the VFD board. By soldering a 0805 zero ohm resistor to the appropriate location, you can instruct MyNOR to load the program from location 7 or 8 when power is applied. Note that this autostart is very fast and cannot be interrupted by pressing a key in the terminal window. To stop this autostart mechanism again, de-solder the resistor or use the MyNOR file manager application to delete the files from EEPROM storage locations 7 and 8.

VFD Driver Software

Here is an example program that shows how text can be output to the VFD. The driver software in this example is linked to the standard output functions of MyNOR’s operating system, so the usual API functions can be used to print text and numbers. The driver code follows on the next pages.

(If you have problems with copying the code from the pdf, please use this link: vfd-example.asm)

```
;------------------------------------------------------------------------------
; VFD Example Program
; You can use this program as basis for your own VFD based applications.
;------------------------------------------------------------------------------

.name "VFD Example"               ; name of the program (displayed in EEPROM program list)

#include <mynor/ram-program.hsm>  ; include this to assemble your program for RAM
#include <mynor/api.hsm>          ; include the MyNOR API function definitions

@regs_start

;variables used by the VFD display driver
DISPBUFFER    DS 12  ;display output buffer (note: the content is reversed from right to left)
DISPPOS       DS 1   ;current output position on display / in display buffer (PTR_L)
VFDADDR       DS 1   ;current VFD memory address 0x10-0x1B

;add your own variables here

@regs_end


;Constants
VFD_RESET     SET   0x10  ; port 3.4
VFD_CH_SPC    SET   0x30  ; "space" (segment off)
VFD_CH_COMMA  SET   0x00  ; ","


;Program entry point, jump to main program
          JMP main

;Tables and Strings
;Important: For you own projects, the following 3 tables must come directly behind "JMP main" !
tab_charset   DB  0x00,0x20  ;add the “,”  (you can add more own characters here)

;Your own strings may follow here
text_hello    DB  "VFD Example",0
text_scroll   DB  "This is a scrolling text on the VFD            ",0


;------------------------------------------------------------------------------

main:
          ;initialize the display
          JSR  vfd_init

          ;print first text
          LD   PTR_L,#<text_hello
          LD   PTR_H,#>text_hello
          JSR  print_str

          ;wait 2 seconds
          LDA  #200
          JSR  delay

          ;clear the display and prepare the scroll text
          JSR  vfd_initscrolltext

_loop     ;print the scrolling text
          LD   PTR_L,#<text_scroll
          LD   PTR_H,#>text_scroll
          JSR  print_str
          JMP  _loop
```

```
;-------------------------------------------------------------------------------
; Display Support Routines ("driver") for the VFD "Samsung HCS-12SS59T"
;-------------------------------------------------------------------------------
; Call vfd_init first.
; Then use the standard output functions to print the text (the API print_xxx
; functions like print_str, print_hexbyte, print_decword, etc.)
; To clear the display use function "vfd_clear" instead of "display_clear".
; Note that the cursor positioning functions do not work on the VFD.
;-------------------------------------------------------------------------------

vfd_init:
          ;initialize the display
          PSH  LR_L
          PSH  LR_H
          ;request ROM version 1.2
          LDA  #0x12
          JSR  request_romversion
          ;prepare the SPI bus
          JSR  spi_init
          ;reset the display (hardware reset)
          LDA  OUTP3
          AND  #(~VFD_RESET)&0xFF
          IO   OUT_PORT3
          OR   #VFD_RESET
          IO   OUT_PORT3
          STA  OUTP3
          ;send init commands:
          ;set display to "all off" mode
          LDA  #0x71
          JSR  vfd_cmd
          ;set display width to 12 digits
          LDA  #0x6C
          JSR  vfd_cmd
          ;set duty cycle (brightness) to max value
          ;brightness is limited by the display supply voltage VEE
          LDA  #0x5F
          JSR  vfd_cmd
          ;load additional characters
          JSR  vfd_loadcharset
          ;clear the display
          JSR  vfd_clear
          ;set display to normal mode (switch it on)
          LDA  #0x70
          JSR  vfd_cmd
          ;redirect stdout to the VFD
          LD   R4_L,#<vfd_stdout
          LD   R4_H,#>vfd_stdout
          JSR  set_stdout
          JMP  return

vfd_clear:
          ;clear the display
          LD   DISPPOS,#<DISPBUFFER+12
          LD   VFDADDR,#0x1B
          LD   PTR_L,#<DISPBUFFER
          LD   PTR_H,#>ZEROPAGE
_diclr01  LDA  #VFD_CH_SPC
          SAP
          INC  PTR_L ;hack! assumes that PTR points always into zero-page!
          LDA  PTR_L
          CMP  #(DISPBUFFER+12)&0xFF
          JNF  _diclr01
          LD   PTR_H,#>ZEROPAGE
          LD   PTR_L,#<DISPBUFFER
          JMP  vfd_bufout

vfd_initscrolltext:
          ;prepare for scrolling text
          ;(start shifting in text at the right side of the display)
          PSH  LR_L
          PSH  LR_H
          JSR  vfd_clear
          LD   DISPPOS,#<DISPBUFFER
          JMP  return
```

```
vfd_out:
          ;output a character on the VFD, shift the display to the left.
          PSH  LR_L
          PSH  LR_H
vfd_stdout:
          ;stdout-callback-entry for the OS (LR is pushed to the stack by the caller)
          JSR  psh_ptr
          PSH  R0
          PSH  R1
          ;convert character in ACCU to VFD character set
          LD   R0,ACCU
          ROL  R0
          ROL  R0
          LD   R0,#0x10
          JPF  _diot03  ;is character
          ;is number/special sign
          CMP  #','
          JPF  _diot07
          CMP  #0x0A
          JPF  _diot05
          CMP  #0x0D
          JPF  _diot06
          LD   R0,#0x30
_diot03   AND  #0x1F
          CLC
          ADD  R0
_diot09   ;check if buffer is full (right most position on display = lowest position in the buffer)
          PSH  ACCU
          LDA  DISPPOS
          CMP  #DISPBUFFER
          JNF  _diot01
          ;shift the buffer
          LD   PTR_L,#<DISPBUFFER+10
          LD   PTR_H,#>ZEROPAGE
_diot02   LAP
          INC  PTR_L
          SAP
          DEC  PTR_L
          DEC  PTR_L
          LDA  PTR_L
          CMP  #<DISPBUFFER-1
          JNF  _diot02
          INC  DISPPOS
          LDA  #15
          JSR  delay
          ;output the character and display the buffer
          POP  ACCU
          DEC  DISPPOS
          LD   PTR_L,DISPPOS
          LD   PTR_H,#>ZEROPAGE
          SAP
          LD   PTR_L,#<DISPBUFFER
          JSR  vfd_bufout
          JMP  _diot04
_diot01   ;output only the character
          DEC  DISPPOS
          LD   PTR_L,DISPPOS
          LD   PTR_H,#>ZEROPAGE
          JSR  spi_select
          LD   R0,VFDADDR
          DEC  VFDADDR
          JSR  vfd_spi_xfer
          POP  ACCU
          SAP
          STA  R0
          JSR  vfd_spi_xfer
          JSR  spi_deselect
_diot04   POP  R1
          POP  R0
          JSR  pop_ptr
          JMP  return
_diot07   ;output a comma
          LDA  #VFD_CH_COMMA
          JMP  _diot09
_diot05   ;LF: clear the line and move ptr back to beginning of line
          JSR  vfd_clear
          JMP  _diot04
```

```
_diot06   ;CR: move ptr back to beginning of the line (do not clear the display)
          LD   DISPPOS,#<DISPBUFFER+12
          LD   VFDADDR,#0x1B
          JMP  _diot04

vfd_cmd:
          ;send a command to the VFD display
          PSH  LR_L
          PSH  LR_H
          STA  R0
          JSR  spi_select
          JSR  vfd_spi_xfer
          JSR  spi_deselect
          JMP  return

vfd_loadcharset:
          ;load custom character set into the display
          LD   PTR_L,#<tab_charset
          LD   PTR_H,#>tab_charset
          LD   R0,#0x20
          LD   R1,#1*2  ;add 1 character: ","
          JMP  vfd_transfer

vfd_bufout:
          ;send out 12 bytes from PTR
          LD   R0,#0x10
          LD   R1,#12

vfd_transfer:
          ;send a command in R0 and data from PTR with length in R1 to the display
          PSH  LR_L
          PSH  LR_H
          JSR  spi_select
          JSR  vfd_spi_xfer
_dibo01   LAP
          INC  PTR_L
          STA  R0
          JSR  vfd_spi_xfer
          DEC  R1
          TST  R1
          JNF  _dibo01
_dibo03   JSR  spi_deselect
          JMP  return

vfd_spi_xfer:
          ;Speed optimized SPI transfer for the VFD display:
          ;One-byte transmission with LSB first.
          ;In: R0 = byte to send
          ;Changes: ACCU, R0
          LD   PAR2,#4
          ;output bit, clock falling edge
_spixf1   ROL  OUTP3
          ROR  R0
          ROR  OUTP3
          LDA  OUTP3
          IO   OUT_PORT3
          ;clock rising edge
          LDA  #0x20 ; SPI_SCK
          OR   OUTP3
          IO   OUT_PORT3
          DEC  PAR2
          ;output bit, clock falling edge
          ROL  OUTP3
          ROR  R0
          ROR  OUTP3
          LDA  OUTP3
          IO   OUT_PORT3
          ;clock rising edge
          LDA  #0x20 ; SPI_SCK
          OR   OUTP3
          IO   OUT_PORT3
          TST  PAR2
          JNF  _spixf1
          ;clock falling edge
          LDA  OUTP3
          IO   OUT_PORT3
          RET
```