Z80 computer test programs

Contents

Z80 simple test programs, mainly for memory tests
- Very simple Z80 test program
- Z80 test program that toggles the MEMSEL signal
- Z80 program to test RAM

Z80 test programs, mainly for i/o device tests
- Test i/o devices
- Test most parts of Z80 computer
- Test most parts of Z80 computer including memory
- Test Z80 computer memory, i/o devices and interrupt
- Testing with Raspberry Pi showing output from SIO channel A and B

Z80 simple test programs, mainly for memory tests

2021-05-17

- Z80 Assembler Syntax (http://www.z80.info/z80syntx.htm)
- Z80 CPU User Manual - um0080.pdf (http://www.zilog.com/docs/z80/um0080.pdf)
- Z80 Family CPU Peripherals - um0081.pdf (http://www.zilog.com/docs/z80/um0081.pdf)

Create simple test programs and assemble, program and run them with a logic analyser.

The assembler z80asm is used: AlbertVeli/z80asm: Clone of z80asm (original is on Savannah) (https://github.com/AlbertVeli/z80asm)

Code, schematics etc. uploaded to GitHub: hansake/Z80_Computer_board: A simple Z80 based computer board (https://github.com/hansake/Z80_Computer_board)

Makefile for assembling the test programs:

```
hal@Perceval:~/z80_computer/tstprg$ cat Makefile
stest01.bin : stest01.z80
        z80asm --list=stest01.lst --output=stest01.bin stest01.z80

stest02.bin : stest02.z80 memsel.bin
        z80asm --list=stest02.lst --output=stest02.bin stest02.z80

memsel.bin : memsel.z80
        z80asm --list=memsel.lst --output=memsel.bin memsel.z80
```

Very simple Z80 test program

2021-05-17

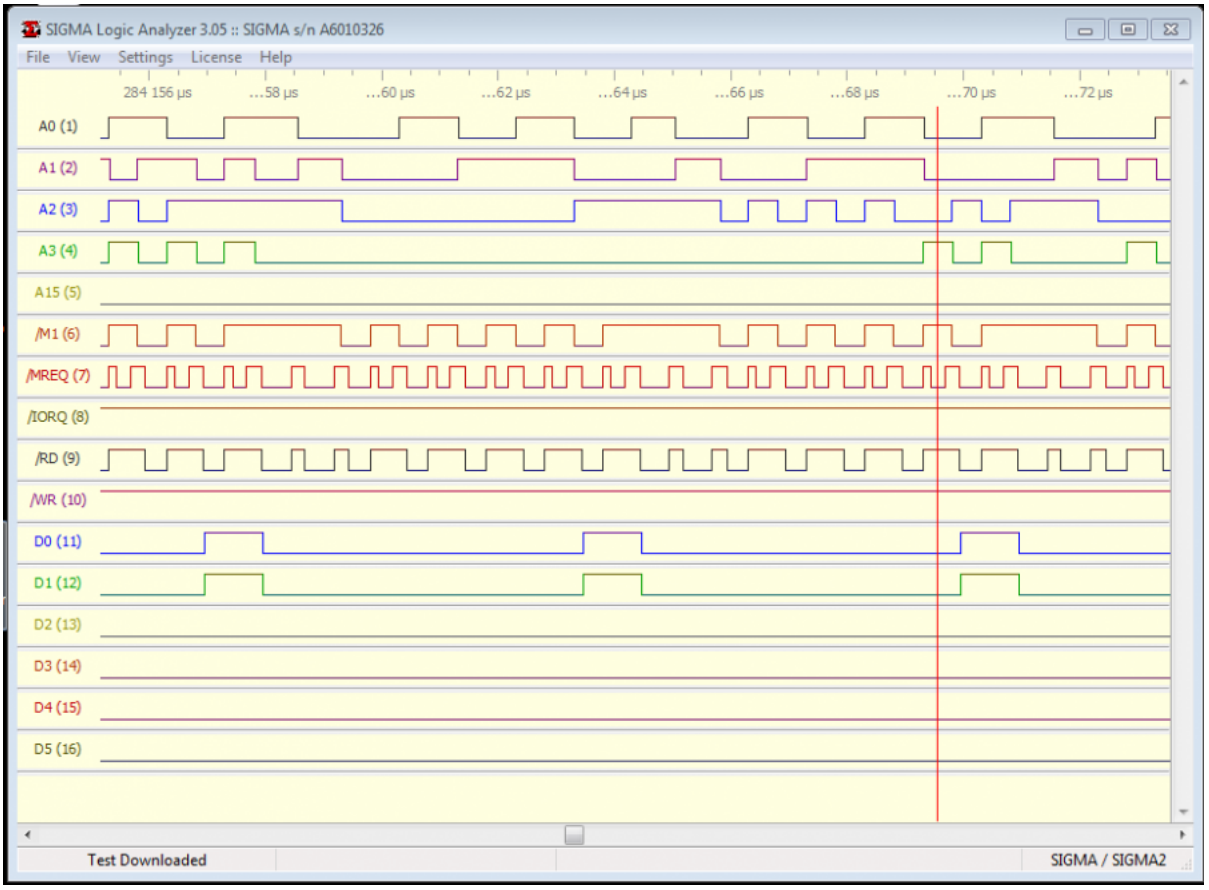
Build a very simple program:

```
hal@Perceval:~/z80_computer/tstprg$ make
z80asm --list=stest01.lst --output=stest01.bin stest01.z80
hal@Perceval:~/z80_computer/tstprg$ cat stest01.lst
# File stest01.z80
0000                    boot:
0000 00                         nop
0001 00                         nop
0002 00                         nop
0003 00                         nop
0004 c3 00 00                   jp boot
0007
0007 ..                         db "stest01.z80 version 0.1"
001e
# End of file stest01.z80
001e
```

Programming EPROM:

```
L0046: Selected device: STMicroelectronics M27C256B.
L0047: Buffer checksum in range of [0h..7FFFh]: 007F8000h - Byte sum (x8), Straight
L0048:
L0049: >> 17.05.2021, 15:05:54
L0050: Buffer checksum type is set to "Byte sum (x8), Straight"
L0051: Buffer block(s) excluded from checksum calculation: Disabled
L0052:
L0053: >> Log file created at 17.05.2021 15:05:54
L0054: Log file name: C:\Users\hal\AppData\Roaming\Elnec\Pg4uw\report.rep
L0055: Log file mode: Append
L0056:
L0057: Buffer is erased with value FFh.
L0058:
L0059:
L0060: Buffer is erased with value FFh.
L0061: >> 17.05.2021, 15:06:45
L0062: Loading file (from "Load file" dialog window): C:\Users\hal\Desktop\EPROM\stest01.bin
L0063: File format selection: automatic
L0064: File format: Binary
L0065: File loading successful. Bytes loaded 30 (0000001Eh).
L0066:
L0067: Buffer operation "Checksum", time elapsed: 15 ms
L0068: Buffer checksum in range of [0h..7FFFh]: 007F6A5Ah - Byte sum (x8), Straight
L0069:
L0070: >> 17.05.2021, 15:07:01
L0071: Opened window "Program?" (parent window "PG4UW v3.15h/06.2015 - universal control...").
L0072:
L0073: >> 17.05.2021, 15:07:07
L0074: Closed window "Program?" (by pressing "Yes").
L0075:
L0076: >> 17.05.2021, 15:07:07
L0077: Programming device: STMicroelectronics M27C256B.
L0078: Buffer checksum in range of [0h..7FFFh]: 007F6A5Ah - Byte sum (x8), Straight
L0079: Checking device ID ...
L0080: Programming device ...
L0081: Verifying device at VCCmax. ...
L0082: Verifying device at VCCmin. ...
L0083: Programming device - O.K.
L0084: Elapsed time: 0:00:20.2
L0085: Statistics info: Success:1  Failure:0  Other failure:0  Total:1
```

Logic Analyzer stest01



Z80 test program that toggles the MEMSEL signal

2021-05-18

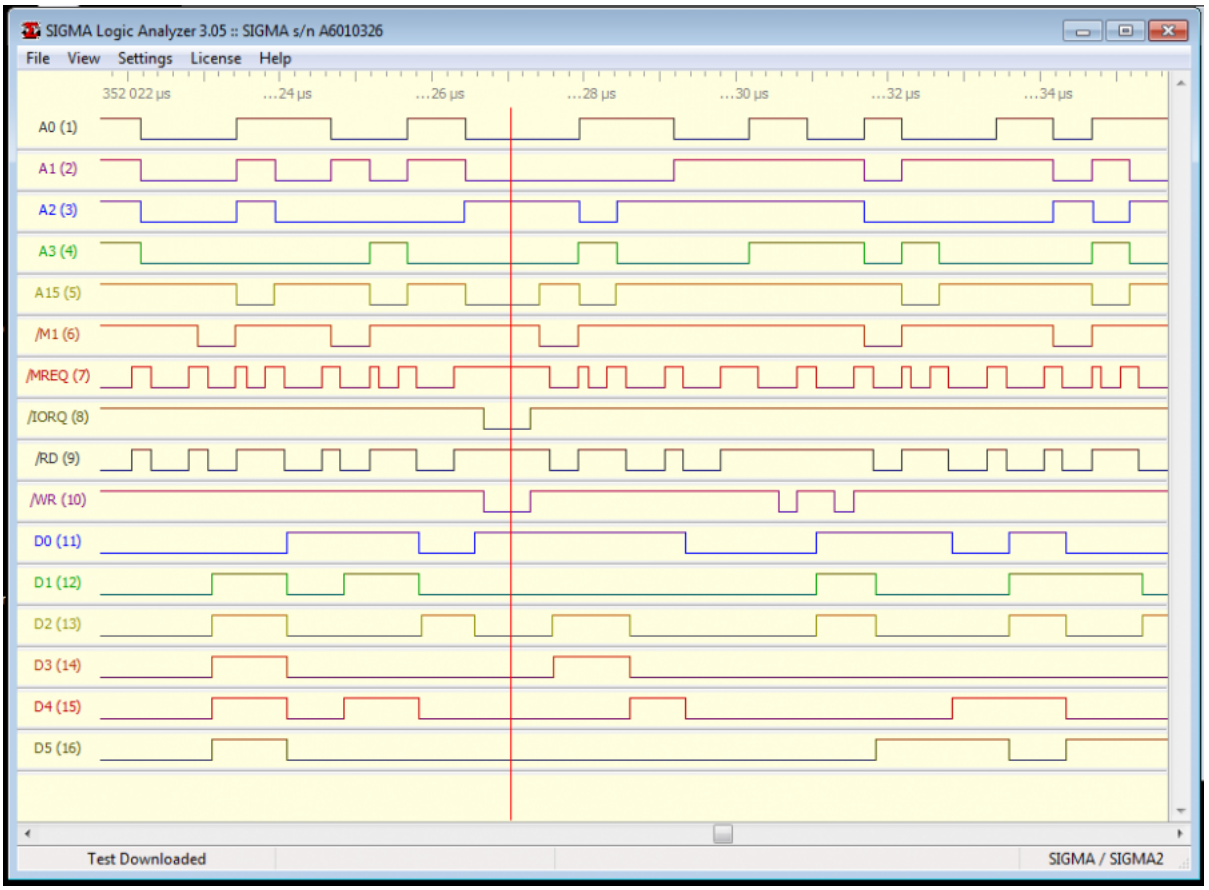
```
hal@Perceval:~/z80_computer/tstprg$ make stest02.bin
z80asm --list=memsel.lst --output=memsel.bin memsel.z80
z80asm --list=stest02.lst --output=stest02.bin stest02.z80
hal@Perceval:~/z80_computer/tstprg$ cat memsel.lst
# File memsel.z80
0000                    ; Test program that toggles the MEMSEL signal in the PLD
0000                    ; and thus also the LED that shows MEMSEL state.
0000                    ; This code is first copied from EPROM to upper RAM
0000                    ; by the program code in EPROM.
0000
0000                            org 0x8000
8000                    upperram:
8000 31 00 00                   ld sp,0x0000    ; initialize SP, the high byte written to the stack
8003                                            ; by a call or push will be to address 0xffff
8003                    selblink:
8003 3e 01                      ld a,1          ; value is ignored when writing
8005                                            ; it is for the benefit of the logic analyzer
8005 d3 04                      out (0x04),a    ; select RAM in lower 32KB address range, LED on
8007 cd 14 80                   call delay
800a 3e 00                      ld a,0
800c d3 00                      out (0x00),a    ; select EPROM in lower 32KB address range, LED off
800e cd 14 80                   call delay
8011 c3 03 80                   jp selblink
8014
8014                    delay:
8014 21 20 4e                   ld hl,20000     ; number of loops to delay between blinks
8017 22 27 80                   ld (loopcnt),hl
801a                    delayloop:
801a 2a 27 80                   ld hl,(loopcnt)
801d 2b                         dec hl
801e 7c                         ld a,h
801f b5                         or l
8020 c8                         ret z
8021 22 27 80                   ld (loopcnt),hl
8024 c3 1a 80                   jp delayloop
8027
8027                    loopcnt:
```

```
8027 00 00                      dw 0
8029
# End of file memsel.z80
8029
hal@Perceval:~/z80_computer/tstprg$ cat stest02.lst
# File stest02.z80
0000                    ; Test program that copies memsel.bin program to upper RAM
0000                    ; and executes it.
0000                    ; memsel is switching between RAM and EPROM in lower memory
0000                    ; and indicates that RAM is selected with the LED
0000                    boot:
0000 01 29 00                   ld bc,prgend - prgstart
0003 21 16 00                   ld hl,prgstart
0006 11 00 80                   ld de,0x8000
0009                    cploop:
0009 78                         ld a,b
000a b1                         or c
000b ca 00 80                   jp z,0x8000
000e 7e                         ld a,(hl)
000f 23                         inc hl
0010 12                         ld (de),a
0011 13                         inc de
0012 0b                         dec bc
0013 c3 09 00                   jp cploop
0016
0016                    prgstart:
0016                            incbin "memsel.bin"
003f                    prgend:
003f
003f ..                         db "stest02.z80 version 0.2"
# End of file stest02.z80
0056
```

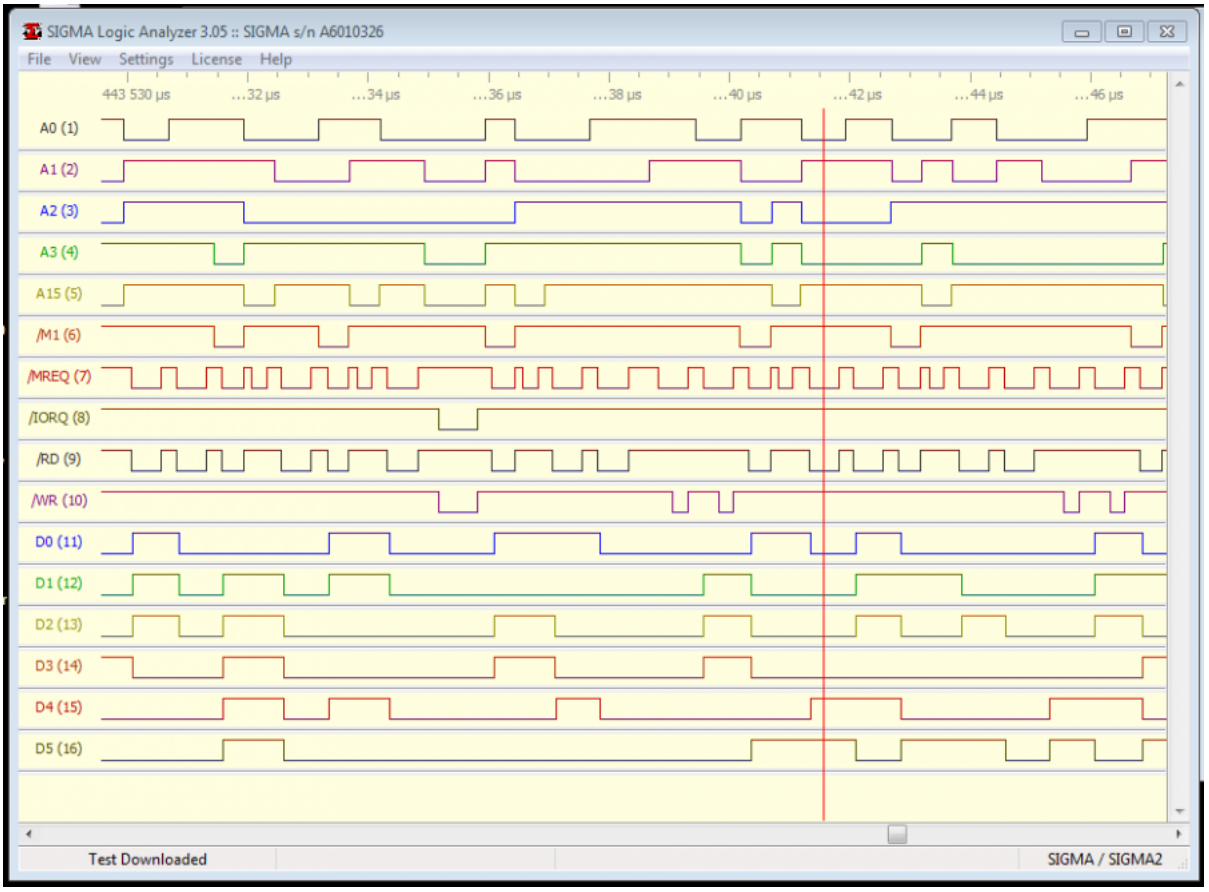
Programming EPROM

```
L0046: Selected device: STMicroelectronics M27C256B.
L0047: Buffer checksum in range of [0h..7FFFh]: 007F8000h - Byte sum (x8), Straight
L0048:
L0049: >> 17.05.2021, 16:40:31
L0050: Buffer checksum type is set to "Byte sum (x8), Straight"
L0051: Buffer block(s) excluded from checksum calculation: Disabled
L0052:
L0053: >> Log file created at 17.05.2021 16:40:31
L0054: Log file name: C:\Users\hal\AppData\Roaming\Elnec\Pg4uw\report.rep
L0055: Log file mode: Append
L0056:
L0057: Buffer is erased with value FFh.
L0058:
L0059:
L0060: Buffer is erased with value FFh.
L0061: >> 17.05.2021, 16:43:02
L0062: Loading file (from "Load file" dialog window): C:\Users\hal\Desktop\EPROM\stest02.bin
L0063: File format selection: automatic
L0064: File format: Binary
L0065: File loading successful. Bytes loaded 83 (00000053h).
L0066:
L0067: Buffer operation "Checksum", time elapsed: 16 ms
L0068: Buffer checksum in range of [0h..7FFFh]: 007F4673h - Byte sum (x8), Straight
L0069:
L0070: >> 17.05.2021, 16:50:08
L0071: Opened window "Program?" (parent window "PG4UW v3.15h/06.2015 - universal control...").
L0072:
L0073: >> 17.05.2021, 16:50:12
L0074: Closed window "Program?" (by pressing "Yes").
L0075:
L0076: >> 17.05.2021, 16:50:12
L0077: Programming device: STMicroelectronics M27C256B.
L0078: Buffer checksum in range of [0h..7FFFh]: 007F4673h - Byte sum (x8), Straight
L0079: Checking device ID ...
L0080: Programming device ...
L0081: Verifying device at VCCmax. ...
L0082: Verifying device at VCCmin. ...
L0083: Programming device - O.K.
L0084: Elapsed time: 0:00:20.9
L0085: Statistics info: Success:1  Failure:0  Other failure:0  Total:1
```

Logic Analyzer stest02, select RAM in low memory



Logic Analyzer stest02, select EPROM in low memory



Z80 program to test RAM

2021-05-19

```
hal@Perceval:~/z80_computer/tstprg$ make stest03.bin
z80asm --list=tstmem.lst --output=tstmem.bin tstmem.z80
z80asm --list=stest03.lst --output=stest03.bin stest03.z80
hal@Perceval:~/z80_computer/tstprg$ cat tstmem.lst
# File tstmem.z80
0000                    ; Test program that makes a simple memory test of low and high RAM
0000                    ; The LED is on while testing low RAM and off while testing
0000                    ; high RAM. If there is an error the LED will flash rapidly.
0000                    ; This code is first copied from EPROM to upper RAM
0000                    ; by the program code in EPROM.
0000
0000                            org 0x8000
8000                    tststart:
8000 31 00 00                   ld sp,0x0000    ; initialize SP, the high byte written to the stack
8003                                            ; by a call or push will be to address 0xffff
8003
8003 3e 0a                      ld a,10         ; start each test cycle with fast LED flashes
8005 cd 99 80                   call errsig
8008
8008                    ; Test low RAM
8008 3e 00                      ld a,0          ; reset error flag
800a 32 ce 80                   ld (ramerr),a
800d 3e 01                      ld a,1
800f d3 04                      out (0x04),a    ; select RAM in lower 32KB address range, LED on
8011 3e 05                      ld a,5          ; Test a couple of times
8013 32 cb 80                   ld (tests),a
8016                    tstlolop:
8016 01 00 80                   ld bc,0x8000    ; number of bytes to test
8019 21 00 00                   ld hl,0x0000    ; start address of test
801c                    tstlo:
801c 1e 00                      ld e,0x00
801e 73                         ld (hl),e
801f 7e                         ld a,(hl)
8020 bb                         cp e
8021 ca 29 80                   jp z,tstloff
8024 3e 01                      ld a,1
8026 32 ce 80                   ld (ramerr),a
8029                    tstloff:
8029 1e ff                      ld e,0xff
802b 73                         ld (hl),e
802c 7e                         ld a,(hl)
802d bb                         cp e
802e ca 36 80                   jp z,tstlonxt
8031 3e 01                      ld a,1
8033 32 ce 80                   ld (ramerr),a
8036                    tstlonxt:
8036 23                         inc hl
8037 0b                         dec bc
8038 78                         ld a,b
8039 b1                         or c
803a 20 e0                      jr nz,tstlo
803c 3a cb 80                   ld a,(tests)
803f 3d                         dec a
8040 32 cb 80                   ld (tests),a
8043 b7                         or a
8044 20 d0                      jr nz,tstlolop
8046
8046 3a ce 80                   ld a,(ramerr)   ; Test if there was an error
8049 b7                         or a
804a 3e 64                      ld a,100        ; flash for a while if error
804c c4 99 80                   call nz,errsig
804f
804f                    ; Test high RAM
804f 3e 00                      ld a,0          ; reset error flag
8051 32 ce 80                   ld (ramerr),a
8054 3e 00                      ld a,0
8056 d3 00                      out (0x00),a    ; select EPROM in lower 32KB address range, LED on
8058 3e 05                      ld a,5          ; Test a couple of times
805a 32 cb 80                   ld (tests),a
805d                    tsthilop:
805d 01 00 70                   ld bc,0x7000    ; number of bytes to test, leave room for stack
8060 21 d0 80                   ld hl,ramtst    ; start address of test, after code and variables
8063                    tsthi:
8063 1e 00                      ld e,0x00
8065 73                         ld (hl),e
8066 7e                         ld a,(hl)
8067 bb                         cp e
```

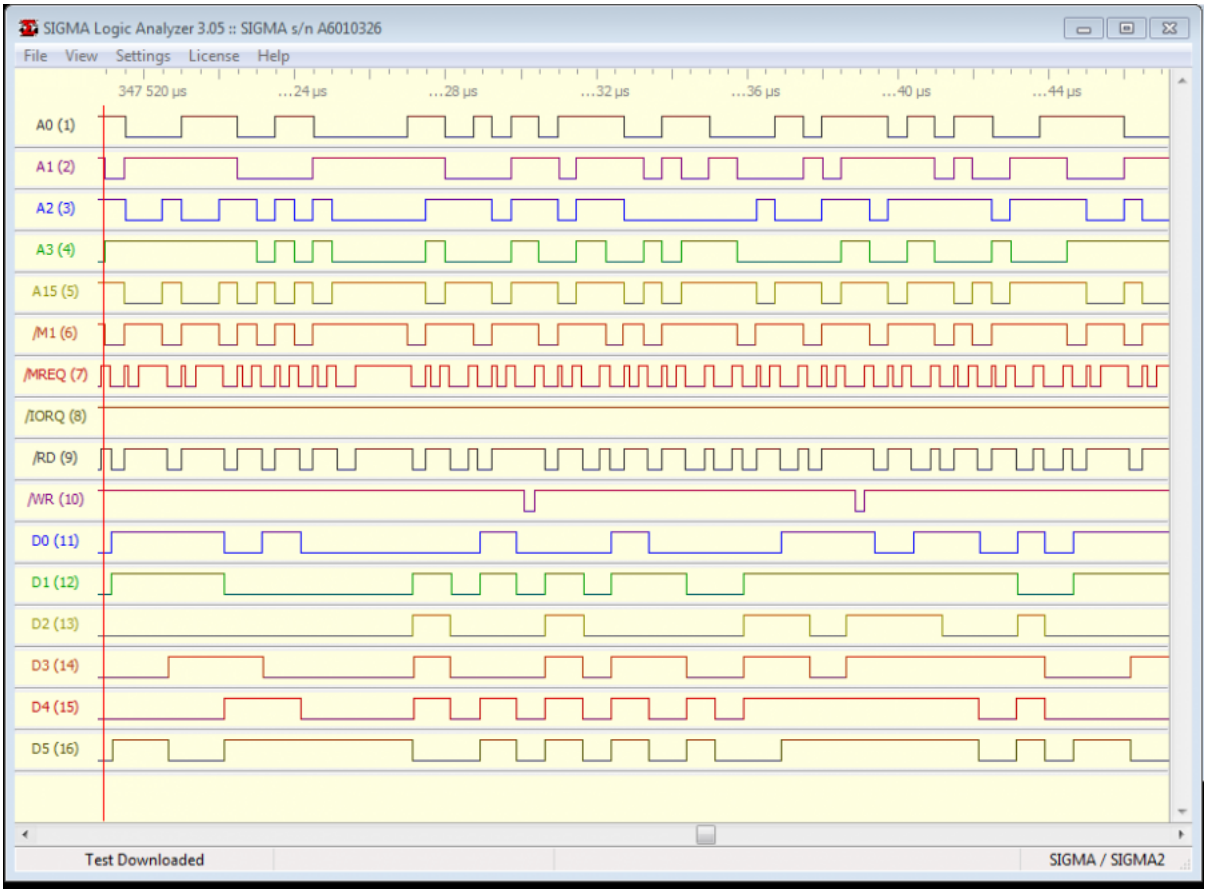
```
8068 ca 70 80                   jp z,tsthiff
806b 3e 01                      ld a,1
806d 32 ce 80                   ld (ramerr),a
8070                    tsthiff:
8070 1e ff                      ld e,0xff
8072 73                         ld (hl),e
8073 7e                         ld a,(hl)
8074 bb                         cp e
8075 ca 7d 80                   jp z,tsthinxt
8078 3e 01                      ld a,1
807a 32 ce 80                   ld (ramerr),a
807d                    tsthinxt:
807d 23                         inc hl
807e 0b                         dec bc
807f 78                         ld a,b
8080 b1                         or c
8081 20 e0                      jr nz,tsthi
8083 3a cb 80                   ld a,(tests)
8086 3d                         dec a
8087 32 cb 80                   ld (tests),a
808a b7                         or a
808b 20 d0                      jr nz,tsthilop
808d
808d 3a ce 80                   ld a,(ramerr)
8090 b7                         or a
8091 3e 64                      ld a,100        ; flash for a while if error
8093 c4 99 80                   call nz,errsig
8096
8096 c3 00 80                   jp tststart     ; run tests "forever"
8099
8099                    ; Blink LED to signal error a number of times
8099                    errsig:
8099 32 cf 80                   ld (errblinks),a
809c                    blinkled:
809c cd a9 80                   call blink
809f 3a cf 80                   ld a,(errblinks)
80a2 3d                         dec a
80a3 32 cf 80                   ld (errblinks),a
80a6 c8                         ret z
80a7 18 f3                      jr blinkled
80a9
80a9                    ; Blink MEMSEL LED once
80a9                    blink:
80a9 3e 01                      ld a,1          ; value is ignored when writing
80ab                                            ; it is for the benefit of the logic analyzer
80ab d3 04                      out (0x04),a    ; select RAM in lower 32KB address range, LED on
80ad cd b8 80                   call delay
80b0 3e 00                      ld a,0
80b2 d3 00                      out (0x00),a    ; select EPROM in lower 32KB address range, LED off
80b4 cd b8 80                   call delay
80b7 c9                         ret
80b8
80b8                    ; Suitable delay for blinking LED
80b8                    delay:
80b8 21 40 1f                   ld hl,8000      ; number of loops to delay between blinks
80bb 22 cc 80                   ld (loopcnt),hl
80be                    delayloop:
80be 2a cc 80                   ld hl,(loopcnt)
80c1 2b                         dec hl
80c2 7c                         ld a,h
80c3 b5                         or l
80c4 c8                         ret z
80c5 22 cc 80                   ld (loopcnt),hl
80c8 c3 be 80                   jp delayloop
80cb
80cb                    tests:
80cb 00                         db 0
80cc                    loopcnt:
80cc 00 00                      dw 0
80ce                    ramerr:
80ce 00                         db 0
80cf                    errblinks:
80cf 00                         db 0
80d0
80d0                    ; Start testing high RAM here
80d0                    ramtst:
80d0
# End of file tstmem.z80
80d0
hal@Perceval:~/z80_computer/tstprg$ cat stest03.lst
# File stest03.z80
```

```
0000                    ; Test program that copies tstmem.bin program to upper RAM
0000                    ; and executes it.
0000                    ; tstmem makes a simple memory test of lo and high RAM
0000                    ; The LED is on while testing low RAM and off while testing
0000                    ; high RAM. If there is an error the LED will flash rapidly
0000                    boot:
0000 01 d0 00                   ld bc,prgend - prgstart
0003 21 16 00                   ld hl,prgstart
0006 11 00 80                   ld de,0x8000
0009                    cploop:
0009 78                         ld a,b
000a b1                         or c
000b ca 00 80                   jp z,0x8000
000e 7e                         ld a,(hl)
000f 23                         inc hl
0010 12                         ld (de),a
0011 13                         inc de
0012 0b                         dec bc
0013 c3 09 00                   jp cploop
0016
0016                    prgstart:
0016                            incbin "tstmem.bin"
00e6                    prgend:
00e6
00e6 ..                         db "stest03.z80 version 0.3"
# End of file stest03.z80
00fd
```

Program EPROM

```
L0159: Buffer is erased with value FFh.
L0160: >> 18.05.2021, 19:27:19
L0161: Loading file (from "Load file" dialog window): C:\Users\hal\Desktop\EPROM\stest03.bin
L0162: File format selection: automatic
L0163: File format: Binary
L0164: File loading successful. Bytes loaded 253 (000000FDh).
L0165:
L0166: Buffer operation "Checksum", time elapsed: 0 ms
L0167: Buffer checksum in range of [0h..7FFFh]: 007EE77Ah - Byte sum (x8), Straight
L0168:
L0169: >> 18.05.2021, 19:27:33
L0170: Opened window "Program?" (parent window "PG4UW v3.15h/06.2015 - universal control...").
L0171:
L0172: >> 18.05.2021, 19:27:38
L0173: Closed window "Program?" (by pressing "Yes").
L0174:
L0175: >> 18.05.2021, 19:27:38
L0176: Programming device: STMicroelectronics M27C256B.
L0177: Buffer checksum in range of [0h..7FFFh]: 007EE77Ah - Byte sum (x8), Straight
L0178: Checking device ID ...
L0179: Programming device ...
L0180: Verifying device at VCCmax. ...
L0181: Verifying device at VCCmin. ...
L0182: Programming device - O.K.
L0183: Elapsed time: 0:00:21.9
L0184: Statistics info: Success:4  Failure:0  Other failure:0  Total:4
```

Logic Analyzer stest03, testing RAM in high memory



Z80 test programs, mainly for i/o device tests

Test i/o devices

2021-05-24

Some references:

- Z80 Family perip.book - um0081.pdf (http://www.z80.info/zip/um0081.pdf)
- LM80C: Z80 SIO and serial communication – LEONARDOMILIANI.com (https://www.leonardomiliani.com/en/2019/lm80c-z80-sio-e-comunicazione-seriale/)
- LM80C: Z80 CTC and interrupts – LEONARDOMILIANI.com (https://www.leonardomiliani.com/en/2019/english-lm80c-z80-ctc-and-interrupts/)
- How to program the z80 CTC | Manualzz (https://manualzz.com/doc/14618543/how-to-program-the-z80-ctc)
- ASCII Character Chart with Decimal, Binary and Hexadecimal Conversions (https://www.eso.org/~ndelmott/ascii.html)
- Z80_Build_Circuits_Index_0 (http://zed80.com/Z80-RETRO/Resources/Circuits/Z80_Circuits_Index_0.htm)
- Z80_computer_schematic_2021-04-26.pdf (http://192.168.42.21/mediawiki/images/f/ff/Z80_computer_schematic_2021-04-26.pdf)

The SIO code is based on:

- antbern/z80-monitor: A Z80 Monitor written in assembly (https://github.com/antbern/z80-monitor)

This test program outputs text to SIO channel A and B.

Makefile

```
hal@Perceval:~/z80_computer/tstprg$ cat Makefile
iotst.bin : iotst.z80 blink.bin
        z80asm --list=iotst.lst --output=iotst.bin iotst.z80

blink.bin : blink.z80
        z80asm --list=blink.lst --output=blink.bin blink.z80

stest01.bin : stest01.z80
        z80asm --list=stest01.lst --output=stest01.bin stest01.z80

stest02.bin : stest02.z80 memsel.bin
        z80asm --list=stest02.lst --output=stest02.bin stest02.z80

memsel.bin : memsel.z80
        z80asm --list=memsel.lst --output=memsel.bin memsel.z80

stest03.bin : stest03.z80 tstmem.bin
        z80asm --list=stest03.lst --output=stest03.bin stest03.z80

tstmem.bin : tstmem.z80
        z80asm --list=tstmem.lst --output=tstmem.bin tstmem.z80
```

Test program to send text on SIO A Tx

```
hal@Perceval:~/z80_computer/tstprg$ cat iotst.z80
; Test i/o devices on Z80 computer

;
; Character definitions
;
EOS:            equ 0x00        ; End Of String
CR:             equ 0x0d        ; Carriage Return (ENTER)
LF:             equ 0x0a        ; Line Feed
SPACE:          equ 0x20        ; Space
TAB:            equ 0x09        ; Tabulator

; Port definitions for the SIO/0 chip
SIO_BASE:       equ 0x08
SIO_A_DATA:     equ SIO_BASE + 0 + 0
SIO_A_CTRL:     equ SIO_BASE + 0 + 2
SIO_B_DATA:     equ SIO_BASE + 1 + 0
SIO_B_CTRL:     equ SIO_BASE + 1 + 2

; Port definitions for the CTC chip
CTC_BASE:       equ 0x0c
CTC_CH0:        equ CTC_BASE + 0
CTC_CH1:        equ CTC_BASE + 1
CTC_CH2:        equ CTC_BASE + 2
CTC_CH3:        equ CTC_BASE + 3

; Divide constant to get an approximate baudrate of 9600
; To get 9600 baud with a 4MHz xtal oscillator the divide constant
; should be 4000000/(9600*2*16) = 13.0208
; Using the CTC divider constant set to 13 will give a baud-rate
; of 4000000/(2*16*13) = 9615 baud which hopefully is close enough.
; If this is not exact enough, another xtal oscillator must be selected,
; it should have the frequency: 3.6864 MHz
; The divide constant should then be set to 12 which gives the baudrate
; of 3686400/(2*16*12) = 9600 baud.
BAUDDIV:        equ 13

; the program starts here at boot
boot:
        ld sp,0x0000    ; initialize SP, the high byte written to the stack
                        ; by a call or push will be to address 0xffff

        ld bc,prgend - prgstart
        ld hl,prgstart
        ld de,0x8000
cploop:
        ld a,b
```

```
        or c
        jp z,cpend
        ld a,(hl)
        inc hl
        ld (de),a
        inc de
        dec bc
        jp cploop
cpend:
        ld a,1
        call blink

        call ctc_ch0_init
        ld a,10
        call delay
        ld a,2
        call blink

        call sio_init
        ld a,10
        call delay
        ld a,3
        call blink

loop:
        ld a, SIO_A_DATA
        ld (sio_data), a
        ld a, SIO_A_CTRL
        ld (sio_ctrl), a
        ld hl, ver_msg
        call print_string
        call print_newline
        ld hl, a_msg
        call print_string
        call print_newline
        ld a,5
        call delay
        ld a,5
        call blink

        ld a, SIO_B_DATA
        ld (sio_data), a
        ld a, SIO_B_CTRL
        ld (sio_ctrl), a
        ld hl, ver_msg
        call print_string
        call print_newline
        ld hl, b_msg
        call print_string
        call print_newline
        ld a,5
        call delay
        ld a,5
        call blink
        jp loop

; ctc_ch0_init: initializes the CTC Channel 0 for baudrate clock to SIO/0
; input TRG0 is supplied by the BCLK signal which is the system clock
; divided by 2 by the ATF22V10C
; affects: A
ctc_ch0_init:
        ld a, 01000111b         ; int off, counter on, prescaler don't care,
                                ; edge don't care, time trigger don't care,
                                ; time constant follows, sw reset,
                                ; this is a ctrl cmd
        out (CTC_CH0), a
        ld a, BAUDDIV           ; divide constant to get baudrate
        out (CTC_CH0), a
        ret

; sio_init: initializes the SIO/0 for serial communication
; affects: HL, B, C
sio_init:
        ; load B with number of bytes
        ld b, sio_init_data_end - sio_init_data
        ld hl, sio_init_data    ; HL points to start of data
        ld c, SIO_A_CTRL        ; I/O-port A for write
        otir                    ; block write of B bytes to [C] starting from HL

        ; load B with number of bytes
        ld b, sio_init_data_end - sio_init_data
```

```
        ld hl, sio_init_data    ; HL points to start of data
        ld c, SIO_B_CTRL        ; I/O-port B for write
        otir                    ; block write of B bytes to [C] starting from HL

        ret

sio_init_data:
        db 00110000b            ; write to WR0: error reset
        db 00011000b            ; write to WR0: channel reset
        db 0x04, 01000100b      ; write to WR4: clkx16, 1 stop bit, no parity
        db 0x05, 01101000b      ; write to WR5: DTR inactive, enable TX 8bit,
                                ; BREAK off, TX on, RTS inactive
        db 0x01, 00000000b      ; write to WR1: no interrupts enabled
        db 0x03, 11000001b      ; write to WR3: enable RX 8bit
sio_init_data_end:

; tx_ready: waits for transmitt buffer to become empty
; affects: none
sio_tx_ready:
        push af
        push bc
sio_tx_ready_loop:
        ld a, (sio_ctrl)
        ld c, a
        in a, (c)               ; read RR0
        bit 2, a                ; check if bit 2 is set
        jr z, sio_tx_ready_loop ; if no - check again
        pop bc
        pop af
        ret

; rx_ready: waits for a character to become available
; affects: none
sio_rx_ready:
        push af
        push bc
sio_rx_ready_loop:
        ld a, (sio_ctrl)
        ld c, a
        in a, (c)               ; read RR0
        bit 0, a                ; check if bit 0 is set
        jr z, sio_rx_ready_loop ; if no - rx buffer has no data => check again
        pop bc
        pop af
        ret

; sends byte in reg A
; affects: none
putc:
        push bc
        push af
        call sio_tx_ready
        ld a, (sio_data)
        ld c, a
        pop af
        out (c), a              ; write character
        pop bc
        ret

; getc: waits for a byte to be available and reads it
; returns: A - read byte
getc:
        push bc
        call sio_rx_ready
        ld a, (sio_data)
        ld c, a
        in a, (c)               ; read character
        pop bc
        ret

; print_newline: prints a CR/LF pair to advance to the next line
; affects: none
print_newline:
        push af
        ld a, CR                ; print Carriage Return
        call putc
        ld a, LF                ; print Line Feed
        call putc
        pop af
        ret
```

```
; print_string: prints a string which starts at adress HL
; and is terminated by EOS-character
; affects: none
print_string:
        push af
        push hl
print_string_1:
        ld a,(hl)               ; load next character
        cp 0                    ; is it en End Of String - character?
        jr z, print_string_2    ; yes - return
        call putc               ; no - print character
        inc hl                  ; HL++
        jr print_string_1       ; do it again
print_string_2:
        pop hl
        pop af
        ret

; Blink and delay routines to load in high RAM
prgstart:
        incbin "blink.bin"
prgend:
blink:
        jp 0x8000
delay:
        jp 0x8003

; Messages to send on serial channels
a_msg:
        db "Z80 computer SIO channal A", 0
b_msg:
        db "Z80 computer SIO channal B", 0
ver_msg:
        db "This is: iotst.z80 version 0.3", 0

        org 0x9000      ; high RAM
sio_ctrl:
        db 0
sio_data:
        db 0
```

Blink routine in high RAM to have a clue of what is happening

```
hal@Perceval:~/z80_computer/tstprg$ cat blink.lst
# File blink.z80
0000                    ; blink.z80: the blink routine blinks the MEMSEL LED
0000                    ; the number of times in register A
0000                    ; Must run in upper 32KB memory
0000                    ; This code is first copied from EPROM to upper RAM
0000                    ; by the program code in EPROM.
0000
0000                            org 0x8000
8000 c3 06 80                   jp blink
8003 c3 16 80                   jp dwait
8006
8006                    ; Blink LED a number of times
8006                    ; using: A
8006                    blink:
8006 32 52 80                   ld (ledblinks),a
8009                    blinkloop:
8009 cd 29 80                   call blinkled
800c 3a 52 80                   ld a,(ledblinks)
800f 3d                         dec a
8010 32 52 80                   ld (ledblinks),a
8013 c8                         ret z
8014 18 f3                      jr blinkloop
8016
8016                    ; Make delay wait a number of times
8016                    ; using: A
8016                    dwait:
8016 32 53 80                   ld (delays),a
8019                    dwaitloop:
8019 cd 38 80                   call delay
801c cd 38 80                   call delay
801f 3a 53 80                   ld a,(delays)
8022 3d                         dec a
8023 32 53 80                   ld (delays),a
```

```
8026 c8                         ret z
8027 18 f0                      jr dwaitloop
8029
8029                    ; Blink MEMSEL LED once
8029                    blinkled:
8029 3e 01                      ld a,1          ; value is ignored when writing
802b                                            ; it is for the benefit of the logic analyzer
802b d3 04                      out (0x04),a    ; select RAM in lower 32KB address range, LED on
802d cd 38 80                   call delay
8030 3e 00                      ld a,0
8032 d3 00                      out (0x00),a    ; select EPROM in lower 32KB address range, LED off
8034 cd 38 80                   call delay
8037 c9                         ret
8038
8038                    ; Suitable delay for blinking LED
8038                    delay:
8038 e5                         push hl
8039 21 40 1f                   ld hl,8000      ; number of loops to delay between blinks
803c 22 50 80                   ld (loopcnt),hl
803f                    delayloop:
803f 2a 50 80                   ld hl,(loopcnt)
8042 2b                         dec hl
8043 7c                         ld a,h
8044 b5                         or l
8045 ca 4e 80                   jp z,leaveloop
8048 22 50 80                   ld (loopcnt),hl
804b c3 3f 80                   jp delayloop
804e                    leaveloop:
804e e1                         pop hl
804f c9                         ret
8050
8050                    loopcnt:
8050 00 00                      dw 0
8052                    ledblinks:
8052 00                         db 0
8053                    delays:
8053 00                         db 0
# End of file blink.z80
8054
```

Terminal output SIO channel B

```
...
This is: iotst.z80 version 0.7, Built 2021-05-23 14:07
Z80 computer SIO channal B
This is: iotst.z80 version 0.7, Built 2021-05-23 14:07
Z80 computer SIO channal B
This is: iotst.z80 version 0.7, Built 2021-05-23 14:07
Z80 computer SIO channal B
qwe <- input on SIO channal B
This is: iotst.z80 version 0.7, Built 2021-05-23 14:07
Z80 computer SIO channal B
...
```

Test most parts of Z80 computer

2021-05-24

Interrupt and memory tests still to be added.

Makefile:

```
hal@Perceval:~/z80_computer/tstprg$ cat Makefile
z80test.bin : z80test.z80 z80hightest.bin
        z80asm --list=z80test.lst --output=z80test.bin z80test.z80

z80hightest.bin : z80hightest.z80
        date +'    db ", Built %F %R"' > built.z80
        z80asm --list=z80hightest.lst --output=z80hightest.bin z80hightest.z80

iotst.bin : iotst.z80 blink.bin
```

```
        date +'    db ", Built %F %R"' > built.z80
        z80asm --list=iotst.lst --output=iotst.bin iotst.z80

blink.bin : blink.z80
        z80asm --list=blink.lst --output=blink.bin blink.z80

stest01.bin : stest01.z80
        z80asm --list=stest01.lst --output=stest01.bin stest01.z80

stest02.bin : stest02.z80 memsel.bin
        z80asm --list=stest02.lst --output=stest02.bin stest02.z80

memsel.bin : memsel.z80
        z80asm --list=memsel.lst --output=memsel.bin memsel.z80

stest03.bin : stest03.z80 tstmem.bin
        z80asm --list=stest03.lst --output=stest03.bin stest03.z80

tstmem.bin : tstmem.z80
        z80asm --list=tstmem.lst --output=tstmem.bin tstmem.z80
```

Assemble the code:

```
hal@Perceval:~/z80_computer/tstprg$ make
date +'    db ", Built %F %R"' > built.z80
z80asm --list=z80hightest.lst --output=z80hightest.bin z80hightest.z80
z80asm --list=z80test.lst --output=z80test.bin z80test.z80
```

Main test program and loader to high RAM:

```
hal@Perceval:~/z80_computer/tstprg$ cat z80test.lst
# File z80test.z80
0000                    ; Test Z80 computer
0000
0000                    HIRAM:  equ 0x8000
0000
0000                    ; The program starts here at boot
0000                    boot:
0000
0000                    ; copy the main program to high RAM at 0x8000
0000 01 a2 0a                   ld bc,prgend - prgstart
0003 21 19 00                   ld hl,prgstart
0006 11 00 80                   ld de,HIRAM
0009                    cploop:
0009 78                         ld a,b
000a b1                         or c
000b ca 16 00                   jp z,cpend
000e 7e                         ld a,(hl)
000f 23                         inc hl
0010 12                         ld (de),a
0011 13                         inc de
0012 0b                         dec bc
0013 c3 09 00                   jp cploop
0016                    cpend:
0016 c3 00 80                   jp HIRAM        ; jump to the copied code
0019
0019                    ; Main program to load in high RAM
0019                    prgstart:
0019                            incbin "z80hightest.bin"
0abb                    prgend:
# End of file z80test.z80
0abb
```

Test program copied to high RAM:

```
hal@Perceval:~/z80_computer/tstprg$ cat z80hightest.lst
# File z80hightest.z80
0000                    ; Test i/o devices on home built Z80 computer board
0000
0000                    ;
0000                    ; Character definitions
0000                    ;
```

```
0000                    EOS:            equ 0x00        ; End Of String
0000                    CR:             equ 0x0d        ; Carriage Return (ENTER)
0000                    LF:             equ 0x0a        ; Line Feed
0000                    SPACE:          equ 0x20        ; Space
0000                    TAB:            equ 0x09        ; Tabulator
0000
0000                    ; Port definitions for switching between low EPROM and RAM
0000                    MEMEPROM:       equ 0x00
0000                    MEMLORAM:       equ 0x04
0000
0000                    ; Port definitions for the SIO/0 chip
0000                    SIO_BASE:       equ 0x08
0000                    SIO_A_DATA:     equ SIO_BASE + 0 + 0
0000                    SIO_A_CTRL:     equ SIO_BASE + 0 + 2
0000                    SIO_B_DATA:     equ SIO_BASE + 1 + 0
0000                    SIO_B_CTRL:     equ SIO_BASE + 1 + 2
0000
0000                    ; Port definitions for the CTC chip
0000                    CTC_BASE:       equ 0x0c
0000                    CTC_CH0:        equ CTC_BASE + 0
0000                    CTC_CH1:        equ CTC_BASE + 1
0000                    CTC_CH2:        equ CTC_BASE + 2
0000                    CTC_CH3:        equ CTC_BASE + 3
0000
0000                    ; Port definitions for the PIO chip
0000                    PIO_BASE:       equ 0x10
0000                    PIO_A_DATA:     equ PIO_BASE + 0 + 0
0000                    PIO_A_CTRL:     equ PIO_BASE + 0 + 2
0000                    PIO_B_DATA:     equ PIO_BASE + 1 + 0
0000                    PIO_B_CTRL:     equ PIO_BASE + 1 + 2
0000
0000                    ; Divide constant in CTC to get an approximate baudrate of 9600
0000                    ; To get 9600 baud with a 4MHz xtal oscillator the divide constant
0000                    ; should be 4000000/(9600*2*16) = 13.0208
0000                    ; Using the CTC divider constant set to 13 will give a baud-rate
0000                    ; of 4000000/(2*16*13) = 9615 baud which hopefully is close enough.
0000                    ; This is tested and works with a 9600 baudrate connection to a Linux PC.
0000                    ; If this is not exact enough, another xtal oscillator must be selected,
0000                    ; it should have theOutput Mode (Mode 0) frequency: 3.6864 MHz
0000                    ; The divide constant should then be set to 12 which gives the baudrate
0000                    ; of 3686400/(2*16*12) = 9600 baud.
0000                    BAUDDIV:        equ 13
0000
0000                    ; The program is copied to high RAM where it is executed
0000                    HIRAM:  equ 0x8000
0000
0000                    ; The program starts here at boot
0000                            org HIRAM
8000
8000                    boot:
8000 31 a2 92                   ld sp,stacktop  ; initialize stack pointer
8003
8003 3e 01                      ld a,1          ; one LED blink after initial start
8005 cd ab 81                   call blink
8008 3e 05                      ld a,5
800a cd bb 81                   call delay
800d
800d cd ef 80                   call ctc_init
8010 3e 02                      ld a,2          ; two LED blinks after CTC init
8012 cd ab 81                   call blink
8015 3e 05                      ld a,5
8017 cd bb 81                   call delay
801a
801a cd 08 81                   call sio_init
801d 3e 03                      ld a,3          ; three LED blinks after SIO init
801f cd ab 81                   call blink
8022 3e 05                      ld a,5
8024 cd bb 81                   call delay
8027
8027 cd de 80                   call pio_init
802a 3e 04                      ld a,4          ; four LED blinks after PIO init
802c cd ab 81                   call blink
802f 3e 05                      ld a,5
8031 cd bb 81                   call delay
8034
8034 3e 11                      ld a, 00010001b ; bit pattern to output on PIO
8036 32 9c 82                   ld (pio_out),a
8039                    testloop:
8039 cd 25 81                   call sel_a_sio
803c 21 61 82                   ld hl, ver_msg
803f cd 9b 81                   call print_string
```

```
8042 cd 8e 81                   call print_newline
8045 21 f5 81                   ld hl, a_msg
8048 cd 9b 81                   call print_string
804b cd 8e 81                   call print_newline
804e 3e 00                      ld a, 0
8050 32 9d 82                   ld (keyin), a
8053                    nxt_a_key:
8053 cd 77 81                   call getkey     ; test if any input character available
8056 b7                         or a
8057 ca 65 80                   jp z, no_more_a_keys
805a cd 5d 81                   call putc
805d 3e 01                      ld a, 1
805f 32 9d 82                   ld (keyin), a
8062 c3 53 80                   jp nxt_a_key
8065                    no_more_a_keys:
8065 3a 9d 82                   ld a, (keyin)
8068 b7                         or a
8069 ca 75 80                   jp z, no_a_key
806c 21 10 82                   ld hl, a_in_msg
806f cd 9b 81                   call print_string
8072 cd 8e 81                   call print_newline
8075                    no_a_key:
8075 3e 05                      ld a,5          ; five LED blinks after sending on SIO A
8077 cd ab 81                   call blink
807a 3e 05                      ld a,5
807c cd bb 81                   call delay
807f
807f 3a 9c 82                   ld a, (pio_out)
8082 d3 10                      out (PIO_A_DATA), a
8084 d3 11                      out (PIO_B_DATA), a
8086 07                         rlca
8087 32 9c 82                   ld (pio_out), a
808a
808a cd 32 81                   call sel_b_sio
808d 21 61 82                   ld hl, ver_msg
8090 cd 9b 81                   call print_string
8093 cd 8e 81                   call print_newline
8096 21 2b 82                   ld hl, b_msg
8099 cd 9b 81                   call print_string
809c cd 8e 81                   call print_newline
809f 3e 00                      ld a, 0
80a1 32 9d 82                   ld (keyin), a
80a4                    nxt_b_key:
80a4 cd 77 81                   call getkey     ; test if any input character available
80a7 b7                         or a
80a8 ca b6 80                   jp z, no_more_b_keys
80ab cd 5d 81                   call putc
80ae 3e 01                      ld a, 1
80b0 32 9d 82                   ld (keyin), a
80b3 c3 a4 80                   jp nxt_b_key
80b6                    no_more_b_keys:
80b6 3a 9d 82                   ld a, (keyin)
80b9 b7                         or a
80ba ca c6 80                   jp z, no_b_key
80bd 21 46 82                   ld hl, b_in_msg
80c0 cd 9b 81                   call print_string
80c3 cd 8e 81                   call print_newline
80c6                    no_b_key:
80c6 3e 06                      ld a,6          ; six LED blinks after sending on SIO B
80c8 cd ab 81                   call blink
80cb 3e 05                      ld a,5
80cd cd bb 81                   call delay
80d0
80d0 3a 9c 82                   ld a, (pio_out)
80d3 d3 10                      out (PIO_A_DATA), a
80d5 d3 11                      out (PIO_B_DATA), a
80d7 07                         rlca
80d8 32 9c 82                   ld (pio_out), a
80db
80db c3 39 80                   jp testloop
80de
80de                    ; pio_init: initialize PIO channel A and B for output (Mode 0)
80de                    ; affects: A
80de                    pio_init:
80de 3e 0f                      ld a, 00001111b         ; mode 0
80e0 d3 12                      out (PIO_A_CTRL), a
80e2 3e 07                      ld a, 00000111b         ; int disable
80e4 d3 12                      out (PIO_A_CTRL), a
80e6 3e 0f                      ld a, 00001111b         ; mode 0
80e8 d3 13                      out (PIO_B_CTRL), a
80ea 3e 07                      ld a, 00000111b         ; int disable
```

```
80ec d3 13                      out (PIO_B_CTRL), a
80ee c9                         ret
80ef
80ef                    ; ctc_init: initializes the CTC channel 0 for baudrate clock to SIO/0
80ef                    ; initializes also CTC channels 1 & 2
80ef                    ; input TRG0-2 is supplied by the BCLK signal which is the system clock
80ef                    ; divided by 2 by the ATF22V10C
80ef                    ; affects: A
80ef                    ctc_init:
80ef                            ; CTC chan 0
80ef 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
80f1                                                    ; edge don't care, time trigger don't care,
80f1                                                    ; time constant follows, sw reset,
80f1                                                    ; this is a ctrl cmd
80f1 d3 0c                      out (CTC_CH0), a
80f3 3e 0d                      ld a, BAUDDIV           ; divide constant to get baudrate
80f5 d3 0c                      out (CTC_CH0), a
80f7                            ; CTC chan 1
80f7 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
80f9                                                    ; edge don't care, time trigger don't care,
80f9                                                    ; time constant follows, sw reset,
80f9                                                    ; this is a ctrl cmd
80f9 d3 0d                      out (CTC_CH1), a
80fb 3e 0a                      ld a, 10                ; divide BCLK by 10
80fd d3 0d                      out (CTC_CH1), a
80ff                            ; CTC chan 2
80ff 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
8101                                                    ; edge don't care, time trigger don't care,
8101                                                    ; time constant follows, sw reset,
8101                                                    ; this is a ctrl cmd
8101 d3 0e                      out (CTC_CH2), a
8103 3e 64                      ld a, 100               ; divide BCLK by 100
8105 d3 0e                      out (CTC_CH2), a
8107 c9                         ret
8108
8108                    ; sio_init: initializes the SIO/0 for serial communication
8108                    ; affects: HL, B, C
8108                    sio_init:
8108                            ; load B with number of bytes
8108 06 0a                      ld b, sio_init_data_end - sio_init_data
810a 21 1b 81                   ld hl, sio_init_data    ; HL points to start of data
810d 0e 0a                      ld c, SIO_A_CTRL        ; I/O-port A for write
810f ed b3                      otir                    ; block write of B bytes to [C] starting from HL
8111
8111                            ; load B with number of bytes
8111 06 0a                      ld b, sio_init_data_end - sio_init_data
8113 21 1b 81                   ld hl, sio_init_data    ; HL points to start of data
8116 0e 0b                      ld c, SIO_B_CTRL        ; I/O-port B for write
8118 ed b3                      otir                    ; block write of B bytes to [C] starting from HL
811a
811a c9                         ret
811b
811b                    sio_init_data:
811b 30                         db 00110000b            ; write to WR0: error reset
811c 18                         db 00011000b            ; write to WR0: channel reset
811d 04 44                      db 0x04, 01000100b      ; write to WR4: clkx16, 1 stop bit, no parity
811f 05 68                      db 0x05, 01101000b      ; write to WR5: DTR inactive, enable TX 8bit,
8121                                                    ; BREAK off, TX on, RTS inactive
8121 01 00                      db 0x01, 00000000b      ; write to WR1: no interrupts enabled
8123 03 c1                      db 0x03, 11000001b      ; write to WR3: enable RX 8bit
8125                    sio_init_data_end:
8125
8125                    ; sel_a_sio: selects SIO channel A for i/o
8125                    ; affects: none
8125                    sel_a_sio:
8125 f5                         push af
8126 3e 08                      ld a, SIO_A_DATA
8128 32 9b 82                   ld (sio_data), a
812b 3e 0a                      ld a, SIO_A_CTRL
812d 32 9a 82                   ld (sio_ctrl), a
8130 f1                         pop af
8131 c9                         ret
8132
8132                    ; sel_b_sio: selects SIO channel B for i/o
8132                    ; affects: none
8132                    sel_b_sio:
8132 f5                         push af
8133 3e 09                      ld a, SIO_B_DATA
8135 32 9b 82                   ld (sio_data), a
8138 3e 0b                      ld a, SIO_B_CTRL
813a 32 9a 82                   ld (sio_ctrl), a
```

```
813d f1                         pop af
813e c9                         ret
813f
813f                    ; tx_ready: waits for transmitt buffer to become empty
813f                    ; affects: none
813f                    sio_tx_ready:
813f f5                         push af
8140 c5                         push bc
8141                    sio_tx_ready_loop:
8141 3a 9a 82                   ld a, (sio_ctrl)
8144 4f                         ld c, a
8145 ed 78                      in a, (c)               ; read RR0
8147 cb 57                      bit 2, a                ; check if bit 2 is set
8149 28 f6                      jr z, sio_tx_ready_loop ; if no - check again
814b c1                         pop bc
814c f1                         pop af
814d c9                         ret
814e
814e                    ; rx_ready: waits for a character to become available
814e                    ; affects: none
814e                    sio_rx_ready:
814e f5                         push af
814f c5                         push bc
8150                    sio_rx_ready_loop:
8150 3a 9a 82                   ld a, (sio_ctrl)
8153 4f                         ld c, a
8154 ed 78                      in a, (c)               ; read RR0
8156 cb 47                      bit 0, a                ; check if bit 0 is set
8158 28 f6                      jr z, sio_rx_ready_loop ; if no - rx buffer has no data => check again
815a c1                         pop bc
815b f1                         pop af
815c c9                         ret
815d
815d                    ; sends byte in reg A
815d                    ; affects: none
815d                    putc:
815d c5                         push bc
815e f5                         push af
815f cd 3f 81                   call sio_tx_ready
8162 3a 9b 82                   ld a, (sio_data)
8165 4f                         ld c, a
8166 f1                         pop af
8167 ed 79                      out (c), a              ; write character
8169 c1                         pop bc
816a c9                         ret
816b
816b                    ; getc: waits for a byte to be available and reads it
816b                    ; returns: A - read byte
816b                    getc:
816b c5                         push bc
816c cd 4e 81                   call sio_rx_ready
816f 3a 9b 82                   ld a, (sio_data)
8172 4f                         ld c, a
8173 ed 78                      in a, (c)               ; read character
8175 c1                         pop bc
8176 c9                         ret
8177
8177                    ; getkey: gets a byte if available and reads it
8177                    ; returns: A - read byte or 0 if no byte available
8177                    getkey:
8177 c5                         push bc
8178 3a 9a 82                   ld a, (sio_ctrl)
817b 4f                         ld c, a
817c ed 78                      in a, (c)               ; read RR0
817e cb 47                      bit 0, a                ; check if bit 0 is set
8180 28 08                      jr z, no_key            ; if no - rx buffer has no data => return 0
8182 3a 9b 82                   ld a, (sio_data)
8185 4f                         ld c, a
8186 ed 78                      in a, (c)               ; read character
8188 c1                         pop bc
8189 c9                         ret
818a                    no_key:
818a 3e 00                      ld a, 0
818c c1                         pop bc
818d c9                         ret
818e
818e                    ; print_newline: prints a CR/LF pair to advance to the next line
818e                    ; affects: none
818e                    print_newline:
818e f5                         push af
818f 3e 0d                      ld a, CR                ; print Carriage Return
```

```
8191 cd 5d 81                   call putc
8194 3e 0a                      ld a, LF                ; print Line Feed
8196 cd 5d 81                   call putc
8199 f1                         pop af
819a c9                         ret
819b
819b                    ; print_string: prints a string which starts at adress HL
819b                    ; and is terminated by EOS-character
819b                    ; affects: none
819b                    print_string:
819b f5                         push af
819c e5                         push hl
819d                    print_string_1:
819d 7e                         ld a,(hl)               ; load next character
819e fe 00                      cp 0                    ; is it en End Of String - character?
81a0 28 06                      jr z, print_string_2    ; yes - return
81a2 cd 5d 81                   call putc               ; no - print character
81a5 23                         inc hl                  ; HL++
81a6 18 f5                      jr print_string_1       ; do it again
81a8                    print_string_2:
81a8 e1                         pop hl
81a9 f1                         pop af
81aa c9                         ret
81ab
81ab                    ; blink: the blink routine blinks the MEMSEL LED
81ab                    ; the number of times to blink in register A
81ab                    ; Must run in upper 32KB memory
81ab
81ab                    ; Blink LED a number of times
81ab                    ; using: A
81ab                    blink:
81ab 32 a0 82                   ld (ledblinks),a
81ae                    blinkloop:
81ae cd ce 81                   call blinkled
81b1 3a a0 82                   ld a,(ledblinks)
81b4 3d                         dec a
81b5 32 a0 82                   ld (ledblinks),a
81b8 c8                         ret z
81b9 18 f3                      jr blinkloop
81bb
81bb                    ; Make delay wait a number of times
81bb                    ; using: A
81bb                    delay:
81bb 32 a1 82                   ld (delays),a
81be                    delayloop:
81be cd dd 81                   call bdelay
81c1 cd dd 81                   call bdelay
81c4 3a a1 82                   ld a,(delays)
81c7 3d                         dec a
81c8 32 a1 82                   ld (delays),a
81cb c8                         ret z
81cc 18 f0                      jr delayloop
81ce
81ce                    ; Blink MEMSEL LED once
81ce                    blinkled:
81ce 3e 01                      ld a,1                  ; value is ignored when writing
81d0                                                    ; it is for the benefit of the logic analyzer
81d0 d3 04                      out (MEMLORAM),a        ; select RAM in lower 32KB address range, LED on
81d2 cd dd 81                   call bdelay
81d5 3e 00                      ld a,0
81d7 d3 00                      out (MEMEPROM),a        ; select EPROM in lower 32KB address range, LED
off
81d9 cd dd 81                   call bdelay
81dc c9                         ret
81dd
81dd                    ; Suitable delay for blinking LED
81dd                    bdelay:
81dd e5                         push hl
81de 21 40 1f                   ld hl,8000      ; number of loops to delay between blinks
81e1 22 9e 82                   ld (loopcnt),hl
81e4                    bdelayloop:
81e4 2a 9e 82                   ld hl,(loopcnt)
81e7 2b                         dec hl
81e8 7c                         ld a,h
81e9 b5                         or l
81ea ca f3 81                   jp z,bleaveloop
81ed 22 9e 82                   ld (loopcnt),hl
81f0 c3 e4 81                   jp bdelayloop
81f3                    bleaveloop:
81f3 e1                         pop hl
81f4 c9                         ret
```

```
81f5
81f5                    ; Messages to send on serial channels
81f5                    a_msg:
81f5 .. 00                      db "Z80 computer SIO channal A", 0
8210                    a_in_msg:
8210 .. 00                      db " <- input on SIO channal A", 0
822b                    b_msg:
822b .. 00                      db "Z80 computer SIO channal B", 0
8246                    b_in_msg:
8246 .. 00                      db " <- input on SIO channal B", 0
8261                    ver_msg:
8261 ..                         db "This is: z80test.z80 version 0.9"
8281                            include "built.z80"
8281 ..                     db ", Built 2021-05-24 14:09"
# End of file built.z80
8299 00                         db 0
829a
829a                    ; Variables
829a                    sio_ctrl:
829a 00                         db 0
829b                    sio_data:
829b 00                         db 0
829c                    pio_out:
829c 00                         db 0
829d                    keyin:
829d 00                         db 0
829e                    loopcnt:
829e 00 00                      dw 0
82a0                    ledblinks:
82a0 00                         db 0
82a1                    delays:
82a1 00                         db 0
82a2
82a2                    ; Reserve space for stack
82a2 00...                      ds 4096
92a2                    stacktop:
92a2
92a2                    ; Start of high RAM test
92a2
92a2                    hiramstart:
92a2
92a2
# End of file z80hightest.z80
92a2
```

Program EPROM:

```
L0001: Welcome to Elnec programmers control program PG4UW.
L0002: Version 3.15h/06.2015.
L0003:
L0004: Today is 24.05.2021, 14:29:01.
L0005:
L0006: Processor: Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz  Frequency: 4000\3990,56 MHz
L0007: CID: 0826-FFC6:08-00-27-75-6B-8A::57F928CB
L0008: Operating system: Windows 7 x64 (v6.1, Build 7601: Service Pack 1).
L0009: Physical RAM memory: 4096 MB or more.
L0010:
L0011:
L0012: >> 24.05.2021, 14:29:02
L0013: Default programmer: PREPROM-02aLV.
L0014: Scanning LPT1 (378h) port(s) for PREPROM-02aLV ..... found at port LPT1 (378h).
L0015: Establishing connection ... done.
L0016: Communication speed rate: reached 100% of maximum in this communication mode.
L0017:
L0018:
L0019:
L0020: >> 24.05.2021, 14:29:09
L0021: Selected device: STMicroelectronics M27C256B.
L0022:
L0023: Buffer operation "Checksum", time elapsed: 16 ms
L0024: Buffer checksum in range of [0h..7FFFh]: 007F8000h - Byte sum (x8), Straight
L0025:
L0026: >---------- Begin of options list ---------- (Program startup)
L0027:
L0028:  |>---------- Device operation options ----------
L0029:  |  ----------  Addresses  ----------
L0030:  |  Device start: "0000000000" h
L0031:  |  Device end: "0000007FFF" h
```

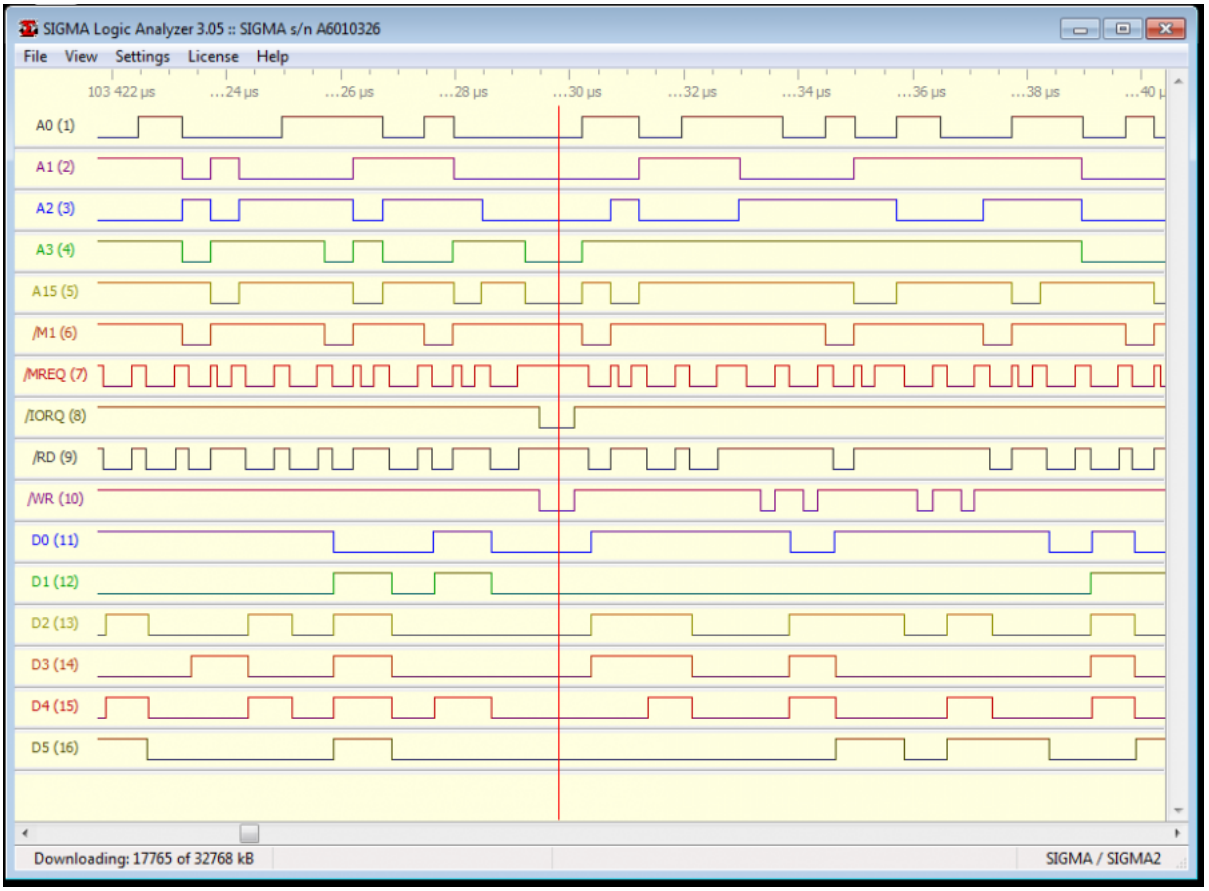
```
L0032:  |  Buffer start: "0000000000" h
L0033:  |  Split: "None"
L0034:  |  ----------  Insertion test and/or ID check  ----------
L0035:  |  Insertion test: "Not supported"
L0036:  |  Device ID check error terminates the operation: "Enable"
L0037:  |  ----------  Command execution  ----------
L0038:  |  Blank check before programming: "Disable"
L0039:  |  Verify after reading: "Enable"
L0040:  |  Verify: "Twice"
L0041:  |  Verify options: " 4.20V - 6.00V"
L0042:  |<---------- Device operation options ----------
L0043:
L0044: <---------- End of options list ---------- (Program startup)
L0045:
L0046: Selected device: STMicroelectronics M27C256B.
L0047: Buffer checksum in range of [0h..7FFFh]: 007F8000h - Byte sum (x8), Straight
L0048:
L0049: >> 24.05.2021, 14:29:09
L0050: Buffer checksum type is set to "Byte sum (x8), Straight"
L0051: Buffer block(s) excluded from checksum calculation: Disabled
L0052:
L0053: >> Log file created at 24.05.2021 14:29:09
L0054: Log file name: C:\Users\hal\AppData\Roaming\Elnec\Pg4uw\report.rep
L0055: Log file mode: Append
L0056:
L0057: Buffer is erased with value FFh.
L0058:
L0059:
L0060: Buffer is erased with value FFh.
L0061: >> 24.05.2021, 14:29:30
L0062: Loading file (from "Load file" dialog window): C:\Users\hal\Desktop\EPROM\z80test.bin
L0063: File format selection: automatic
L0064: File format: Binary
L0065: File loading successful. Bytes loaded 4795 (000012BBh).
L0066:
L0067: Buffer operation "Checksum", time elapsed: 15 ms
L0068: Buffer checksum in range of [0h..7FFFh]: 006E0151h - Byte sum (x8), Straight
L0069:
L0070: >> 24.05.2021, 14:29:52
L0071: Opened window "Program?" (parent window "PG4UW v3.15h/06.2015 - universal control...").
L0072:
L0073: >> 24.05.2021, 14:29:55
L0074: Closed window "Program?" (by pressing "Yes").
L0075:
L0076: >> 24.05.2021, 14:29:55
L0077: Programming device: STMicroelectronics M27C256B.
L0078: Buffer checksum in range of [0h..7FFFh]: 006E0151h - Byte sum (x8), Straight
L0079: Checking device ID ...
L0080: Programming device ...
L0081: Verifying device at VCCmax. ...
L0082: Verifying device at VCCmin. ...
L0083: Programming device - O.K.
L0084: Elapsed time: 0:01:03.8
L0085: Statistics info: Success:1  Failure:0  Other failure:0  Total:1
```

Output to serial SIO channel:

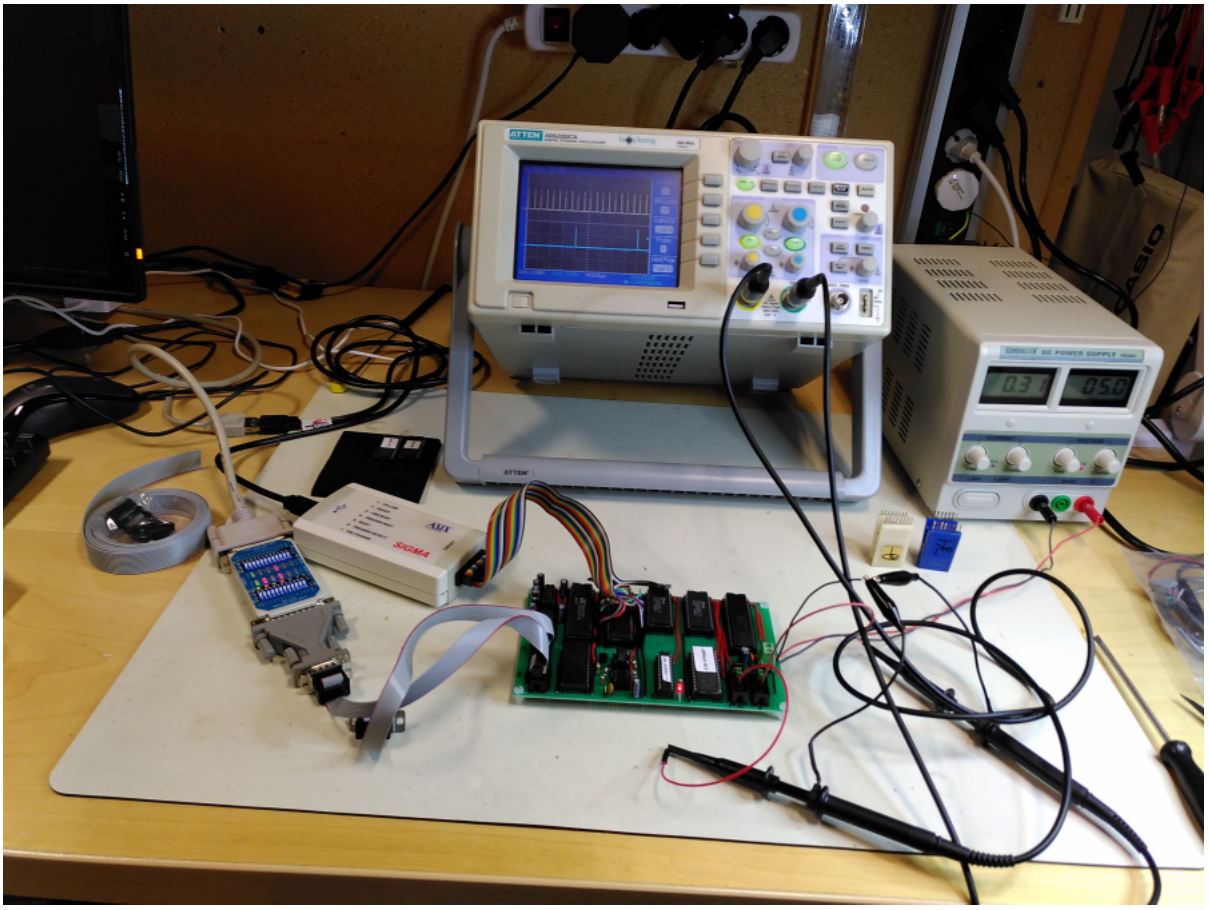
```
...
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal A
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal A
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal A
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal A
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal A
...
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal B
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal B
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal B
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
Z80 computer SIO channal B
This is: z80test.z80 version 0.9, Built 2021-05-24 14:09
```

```
Z80 computer SIO channal B
...
```

Logic analyzer:



Test setup:



Test most parts of Z80 computer including memory

2021-05-25

Makefile

```
z80test.bin : z80test.z80 z80hightest.bin
        z80asm --list=z80test.lst --output=z80test.bin z80test.z80

z80hightest.bin : z80hightest.z80
        date +'    db ", Built %F %R"' > built.z80
        z80asm --list=z80hightest.lst --output=z80hightest.bin z80hightest.z80
```

make

```
hal@Perceval:~/z80_computer/tstprg$ make
date +'    db ", Built %F %R"' > built.z80
z80asm --list=z80hightest.lst --output=z80hightest.bin z80hightest.z80
z80asm --list=z80test.lst --output=z80test.bin z80test.z80
```

z80test

```
hal@Perceval:~/z80_computer/tstprg$ cat z80test.lst
# File z80test.z80
0000                    ; Test Z80 computer hardware, copies code from EPROM to
0000                    ; RAM and then runs it.
0000                    ;
0000                    ; You are free to use, modify, and redistribute
0000                    ; this source code. The software is provided "as is",
0000                    ; without warranty of any kind.
0000                    ; Hastily Cobbled Together 2021 by Hans-Ake Lund.
0000                    ;
```

```
0000
0000                    HIRAM:  equ 0x8000
0000
0000                            org 0x0000
0000
0000                    ; The program starts here at boot
0000                    boot:
0000
0000                    ; copy the main program to high RAM at 0x8000
0000 01 54 14                   ld bc,prgend - prgstart
0003 21 19 00                   ld hl,prgstart
0006 11 00 80                   ld de,HIRAM
0009                    cploop:
0009 78                         ld a,b
000a b1                         or c
000b ca 16 00                   jp z,cpend
000e 7e                         ld a,(hl)
000f 23                         inc hl
0010 12                         ld (de),a
0011 13                         inc de
0012 0b                         dec bc
0013 c3 09 00                   jp cploop
0016                    cpend:
0016 c3 00 80                   jp HIRAM        ; jump to the copied code
0019
0019                    ; Main program to load in high RAM
0019                    prgstart:
0019                            incbin "z80hightest.bin"
146d                    prgend:
# End of file z80test.z80
146d
```

z80hightest

```
hal@Perceval:~/z80_computer/tstprg$ cat z80hightest.lst
# File z80hightest.z80
0000                    ; Test memory and i/o devices on home built Z80 computer board
0000                    ;
0000                    ; You are free to use, modify, and redistribute
0000                    ; this source code. The software is provided "as is",
0000                    ; without warranty of any kind.
0000                    ; Hastily Cobbled Together 2021 by Hans-Ake Lund.
0000                    ;
0000
0000                    ; Character definitions
0000                    ;
0000                    EOS:            equ 0x00        ; End Of String
0000                    CR:             equ 0x0d        ; Carriage Return (ENTER)
0000                    LF:             equ 0x0a        ; Line Feed
0000                    SPACE:          equ 0x20        ; Space
0000                    TAB:            equ 0x09        ; Tabulator
0000
0000                    ; Port definitions for switching between low EPROM and RAM
0000                    MEMEPROM:       equ 0x00
0000                    MEMLORAM:       equ 0x04
0000
0000                    ; Port definitions for the SIO/0 chip
0000                    SIO_BASE:       equ 0x08
0000                    SIO_A_DATA:     equ SIO_BASE + 0 + 0
0000                    SIO_A_CTRL:     equ SIO_BASE + 0 + 2
0000                    SIO_B_DATA:     equ SIO_BASE + 1 + 0
0000                    SIO_B_CTRL:     equ SIO_BASE + 1 + 2
0000
0000                    ; Port definitions for the CTC chip
0000                    CTC_BASE:       equ 0x0c
0000                    CTC_CH0:        equ CTC_BASE + 0
0000                    CTC_CH1:        equ CTC_BASE + 1
0000                    CTC_CH2:        equ CTC_BASE + 2
0000                    CTC_CH3:        equ CTC_BASE + 3
0000
0000                    ; Port definitions for the PIO chip
0000                    PIO_BASE:       equ 0x10
0000                    PIO_A_DATA:     equ PIO_BASE + 0 + 0
0000                    PIO_A_CTRL:     equ PIO_BASE + 0 + 2
0000                    PIO_B_DATA:     equ PIO_BASE + 1 + 0
0000                    PIO_B_CTRL:     equ PIO_BASE + 1 + 2
0000
0000                    ; Divide constant in CTC to get an approximate baudrate of 9600
```

```
0000                    ; To get 9600 baud with a 4MHz xtal oscillator the divide constant
0000                    ; should be 4000000/(9600*2*16) = 13.0208
0000                    ; Using the CTC divider constant set to 13 will give a baud-rate
0000                    ; of 4000000/(2*16*13) = 9615 baud which hopefully is close enough.
0000                    ; This is tested and works with a 9600 baudrate connection to a Linux PC.
0000                    ;
0000                    ; If this is not exact enough, another xtal oscillator must be selected,
0000                    ; it should have the frequency: 3.6864 MHz
0000                    ; The divide constant will then be set to 12 which gives the baudrate
0000                    ; of 3686400/(2*16*12) = 9600 baud.
0000                    BAUDDIV:        equ 13
0000
0000                    ; The program is copied to high RAM where it is executed
0000                    HIRAM:  equ 0x8000
0000
0000                    ; The program starts here when copied from EPROM at boot
0000                            org HIRAM
8000
8000                    boot:
8000 31 54 94                   ld sp,stacktop  ; initialize stack pointer
8003
8003 3e 01                      ld a,1          ; one LED blink after initial start
8005 cd b5 82                   call blink
8008 3e 05                      ld a,5
800a cd c5 82                   call delay
800d
800d cd eb 81                   call ctc_init
8010 3e 02                      ld a,2          ; two LED blinks after CTC init
8012 cd b5 82                   call blink
8015 3e 05                      ld a,5
8017 cd c5 82                   call delay
801a
801a cd 04 82                   call sio_init
801d 3e 03                      ld a,3          ; three LED blinks after SIO init
801f cd b5 82                   call blink
8022 3e 05                      ld a,5
8024 cd c5 82                   call delay
8027
8027 cd da 81                   call pio_init
802a 3e 04                      ld a,4          ; four LED blinks after PIO init
802c cd b5 82                   call blink
802f 3e 05                      ld a,5
8031 cd c5 82                   call delay
8034
8034 3e 11                      ld a, 00010001b ; bit pattern to output on PIO
8036 32 4c 84                   ld (pio_out),a
8039                    testloop:
8039                    ; test SIO channel A
8039 cd 21 82                   call sel_a_sio
803c cd 8a 82                   call print_newline
803f 21 05 83                   ld hl, ver_msg
8042 cd 97 82                   call print_string
8045 cd 8a 82                   call print_newline
8048 21 45 83                   ld hl, a_msg
804b cd 97 82                   call print_string
804e cd 8a 82                   call print_newline
8051 3e 00                      ld a, 0
8053 32 4d 84                   ld (keyin), a
8056                    nxt_a_key:
8056 cd 73 82                   call getkey     ; test if any input character available
8059 b7                         or a
805a ca 68 80                   jp z, no_more_a_keys
805d cd 59 82                   call putc
8060 3e 01                      ld a, 1
8062 32 4d 84                   ld (keyin), a
8065 c3 56 80                   jp nxt_a_key
8068                    no_more_a_keys:
8068 3a 4d 84                   ld a, (keyin)
806b b7                         or a
806c ca 78 80                   jp z, no_a_key
806f 21 5d 83                   ld hl, a_in_msg
8072 cd 97 82                   call print_string
8075 cd 8a 82                   call print_newline
8078                    no_a_key:
8078 3e 05                      ld a,5          ; five LED blinks after sending on SIO A
807a cd b5 82                   call blink
807d 3e 05                      ld a,5
807f cd c5 82                   call delay
8082
8082 cd a7 82                   call pio_bits_out
8085
```

```
8085                    ; test SIO channel B
8085 cd 2e 82                   call sel_b_sio
8088 cd 8a 82                   call print_newline
808b 21 05 83                   ld hl, ver_msg
808e cd 97 82                   call print_string
8091 cd 8a 82                   call print_newline
8094 21 78 83                   ld hl, b_msg
8097 cd 97 82                   call print_string
809a cd 8a 82                   call print_newline
809d 3e 00                      ld a, 0
809f 32 4d 84                   ld (keyin), a
80a2                    nxt_b_key:
80a2 cd 73 82                   call getkey     ; test if any input character available
80a5 b7                         or a
80a6 ca b4 80                   jp z, no_more_b_keys
80a9 cd 59 82                   call putc
80ac 3e 01                      ld a, 1
80ae 32 4d 84                   ld (keyin), a
80b1 c3 a2 80                   jp nxt_b_key
80b4                    no_more_b_keys:
80b4 3a 4d 84                   ld a, (keyin)
80b7 b7                         or a
80b8 ca c4 80                   jp z, no_b_key
80bb 21 90 83                   ld hl, b_in_msg
80be cd 97 82                   call print_string
80c1 cd 8a 82                   call print_newline
80c4                    no_b_key:
80c4 3e 06                      ld a, 6                 ; six LED blinks after sending on SIO B
80c6 cd b5 82                   call blink
80c9 3e 05                      ld a, 5
80cb cd c5 82                   call delay
80ce
80ce cd a7 82                   call pio_bits_out
80d1
80d1                    ; Test RAM
80d1 21 ab 83                   ld hl, lo_ram_tst_start
80d4 cd 21 82                   call sel_a_sio
80d7 cd 97 82                   call print_string
80da cd 8a 82                   call print_newline
80dd cd 2e 82                   call sel_b_sio
80e0 cd 97 82                   call print_string
80e3 cd 8a 82                   call print_newline
80e6 cd 5c 81                   call test_low_ram
80e9 21 c2 83                   ld hl, lo_ram_tst_ok
80ec 3a 52 84                   ld a,(ramerr)           ; was there an error?
80ef b7                         or a
80f0 ca f6 80                   jp z, lo_ram_prt        ; no error
80f3 21 dc 83                   ld hl, lo_ram_tst_err
80f6                    lo_ram_prt:
80f6 cd 21 82                   call sel_a_sio
80f9 cd 97 82                   call print_string
80fc cd 8a 82                   call print_newline
80ff cd 2e 82                   call sel_b_sio
8102 cd 97 82                   call print_string
8105 cd 8a 82                   call print_newline
8108 3e 07                      ld a, 7                 ; seven LED blinks after testing low RAM
810a cd b5 82                   call blink
810d 3e 05                      ld a,5
810f cd c5 82                   call delay
8112
8112 cd a7 82                   call pio_bits_out
8115
8115 21 f9 83                   ld hl, hi_ram_tst_start
8118 cd 21 82                   call sel_a_sio
811b cd 97 82                   call print_string
811e cd 8a 82                   call print_newline
8121 cd 2e 82                   call sel_b_sio
8124 cd 97 82                   call print_string
8127 cd 8a 82                   call print_newline
812a cd 9b 81                   call test_high_ram
812d 21 11 84                   ld hl, hi_ram_tst_ok
8130 3a 52 84                   ld a,(ramerr)           ; was there an error?
8133 b7                         or a
8134 ca 3a 81                   jp z, hi_ram_prt        ; no error
8137 21 2c 84                   ld hl, hi_ram_tst_err
813a                    hi_ram_prt:
813a cd 21 82                   call sel_a_sio
813d cd 97 82                   call print_string
8140 cd 8a 82                   call print_newline
8143 cd 2e 82                   call sel_b_sio
8146 cd 97 82                   call print_string
```

```
8149 cd 8a 82                   call print_newline
814c 3e 08                      ld a, 8                 ; eight LED blinks after testing high RAM
814e cd b5 82                   call blink
8151 3e 05                      ld a, 5
8153 cd c5 82                   call delay
8156
8156 cd a7 82                   call pio_bits_out
8159
8159 c3 39 80                   jp testloop
815c
815c                    ; test_low_ram: test lower 32K RAM memory
815c                    test_low_ram:
815c 3e 00                      ld a, 0                 ; reset error flag
815e 32 52 84                   ld (ramerr), a
8161 3e 01                      ld a, 1
8163 d3 04                      out (MEMLORAM), a       ; select RAM in lower 32KB address range, LED on
8165 3e 05                      ld a, 5                 ; test a couple of times
8167 32 53 84                   ld (tests), a
816a                    tstlolop:
816a 01 00 80                   ld bc, 0x8000           ; number of bytes to test
816d 21 00 00                   ld hl, 0x0000           ; start address of test
8170                    tstlo:
8170 1e 00                      ld e, 0x00
8172 73                         ld (hl), e
8173 7e                         ld a, (hl)
8174 bb                         cp e
8175 ca 7d 81                   jp z, tstloff
8178 3e 01                      ld a, 1
817a 32 52 84                   ld (ramerr), a
817d                    tstloff:
817d 1e ff                      ld e, 0xff
817f 73                         ld (hl), e
8180 7e                         ld a, (hl)
8181 bb                         cp e
8182 ca 8a 81                   jp z, tstlonxt
8185 3e 01                      ld a, 1
8187 32 52 84                   ld (ramerr), a
818a                    tstlonxt:
818a 23                         inc hl
818b 0b                         dec bc
818c 78                         ld a, b
818d b1                         or c
818e 20 e0                      jr nz, tstlo
8190 3a 53 84                   ld a, (tests)
8193 3d                         dec a
8194 32 53 84                   ld (tests),a
8197 b7                         or a
8198 20 d0                      jr nz, tstlolop
819a c9                         ret
819b
819b                    ; test_high_ram: test higher 32K RAM memory
819b                    test_high_ram:
819b 3e 00                      ld a, 0                 ; reset error flag
819d 32 52 84                   ld (ramerr), a
81a0 3e 00                      ld a, 0
81a2 d3 00                      out (MEMEPROM), a       ; select EPROM in lower 32KB address range, LED
on
81a4                                                    ; LED indicates that higher RAM is tested
81a4 3e 05                      ld a, 5                 ; test a couple of times
81a6 32 53 84                   ld (tests), a
81a9                    tsthilop:
81a9 01 ab 6b                   ld bc, 0xffff - hiramstart      ; number of bytes to test
81ac 21 54 94                   ld hl, hiramstart       ; start address of test, after code and variables
81af                    tsthi:
81af 1e 00                      ld e, 0x00
81b1 73                         ld (hl), e
81b2 7e                         ld a, (hl)
81b3 bb                         cp e
81b4 ca bc 81                   jp z, tsthiff
81b7 3e 01                      ld a, 1
81b9 32 52 84                   ld (ramerr), a
81bc                    tsthiff:
81bc 1e ff                      ld e, 0xff
81be 73                         ld (hl), e
81bf 7e                         ld a, (hl)
81c0 bb                         cp e
81c1 ca c9 81                   jp z, tsthinxt
81c4 3e 01                      ld a, 1
81c6 32 52 84                   ld (ramerr), a
81c9                    tsthinxt:
81c9 23                         inc hl
```

```
81ca 0b                         dec bc
81cb 78                         ld a, b
81cc b1                         or c
81cd 20 e0                      jr nz, tsthi
81cf 3a 53 84                   ld a, (tests)
81d2 3d                         dec a
81d3 32 53 84                   ld (tests), a
81d6 b7                         or a
81d7 20 d0                      jr nz, tsthilop
81d9 c9                         ret
81da
81da                    ; pio_init: initialize PIO channel A and B for output (Mode 0)
81da                    ; affects: A
81da                    pio_init:
81da 3e 0f                      ld a, 00001111b         ; mode 0
81dc d3 12                      out (PIO_A_CTRL), a
81de 3e 07                      ld a, 00000111b         ; int disable
81e0 d3 12                      out (PIO_A_CTRL), a
81e2 3e 0f                      ld a, 00001111b         ; mode 0
81e4 d3 13                      out (PIO_B_CTRL), a
81e6 3e 07                      ld a, 00000111b         ; int disable
81e8 d3 13                      out (PIO_B_CTRL), a
81ea c9                         ret
81eb
81eb                    ; ctc_init: initializes the CTC channel 0 for baudrate clock to SIO/0
81eb                    ; initializes also CTC channels 1 & 2
81eb                    ; input TRG0-2 is supplied by the BCLK signal which is the system clock
81eb                    ; divided by 2 by the ATF22V10C
81eb                    ; affects: A
81eb                    ctc_init:
81eb                            ; CTC chan 0
81eb 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
81ed                                                    ; edge don't care, time trigger don't care,
81ed                                                    ; time constant follows, sw reset,
81ed                                                    ; this is a ctrl cmd
81ed d3 0c                      out (CTC_CH0), a
81ef 3e 0d                      ld a, BAUDDIV           ; divide constant to get baudrate
81f1 d3 0c                      out (CTC_CH0), a
81f3                            ; CTC chan 1
81f3 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
81f5                                                    ; edge don't care, time trigger don't care,
81f5                                                    ; time constant follows, sw reset,
81f5                                                    ; this is a ctrl cmd
81f5 d3 0d                      out (CTC_CH1), a
81f7 3e 0a                      ld a, 10                ; divide BCLK by 10
81f9 d3 0d                      out (CTC_CH1), a
81fb                            ; CTC chan 2
81fb 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
81fd                                                    ; edge don't care, time trigger don't care,
81fd                                                    ; time constant follows, sw reset,
81fd                                                    ; this is a ctrl cmd
81fd d3 0e                      out (CTC_CH2), a
81ff 3e 64                      ld a, 100               ; divide BCLK by 100
8201 d3 0e                      out (CTC_CH2), a
8203 c9                         ret
8204
8204                    ; sio_init: initializes the SIO/0 for serial communication
8204                    ; affects: HL, B, C
8204                    sio_init:
8204                            ; load B with number of bytes
8204 06 0a                      ld b, sio_init_data_end - sio_init_data
8206 21 17 82                   ld hl, sio_init_data    ; HL points to start of data
8209 0e 0a                      ld c, SIO_A_CTRL        ; I/O-port A for write
820b ed b3                      otir                    ; block write of B bytes to [C] starting from HL
820d
820d                            ; load B with number of bytes
820d 06 0a                      ld b, sio_init_data_end - sio_init_data
820f 21 17 82                   ld hl, sio_init_data    ; HL points to start of data
8212 0e 0b                      ld c, SIO_B_CTRL        ; I/O-port B for write
8214 ed b3                      otir                    ; block write of B bytes to [C] starting from HL
8216
8216 c9                         ret
8217
8217                    sio_init_data:
8217 30                         db 00110000b            ; write to WR0: error reset
8218 18                         db 00011000b            ; write to WR0: channel reset
8219 04 44                      db 0x04, 01000100b      ; write to WR4: clkx16, 1 stop bit, no parity
821b 05 68                      db 0x05, 01101000b      ; write to WR5: DTR inactive, enable TX 8bit,
821d                                                    ; BREAK off, TX on, RTS inactive
821d 01 00                      db 0x01, 00000000b      ; write to WR1: no interrupts enabled
821f 03 c1                      db 0x03, 11000001b      ; write to WR3: enable RX 8bit
```

```
8221                    sio_init_data_end:
8221
8221                    ; sel_a_sio: selects SIO channel A for i/o
8221                    ; affects: none
8221                    sel_a_sio:
8221 f5                         push af
8222 3e 08                      ld a, SIO_A_DATA
8224 32 4b 84                   ld (sio_data), a
8227 3e 0a                      ld a, SIO_A_CTRL
8229 32 4a 84                   ld (sio_ctrl), a
822c f1                         pop af
822d c9                         ret
822e
822e                    ; sel_b_sio: selects SIO channel B for i/o
822e                    ; affects: none
822e                    sel_b_sio:
822e f5                         push af
822f 3e 09                      ld a, SIO_B_DATA
8231 32 4b 84                   ld (sio_data), a
8234 3e 0b                      ld a, SIO_B_CTRL
8236 32 4a 84                   ld (sio_ctrl), a
8239 f1                         pop af
823a c9                         ret
823b
823b                    ; tx_ready: waits for transmitt buffer to become empty
823b                    ; affects: none
823b                    sio_tx_ready:
823b f5                         push af
823c c5                         push bc
823d                    sio_tx_ready_loop:
823d 3a 4a 84                   ld a, (sio_ctrl)
8240 4f                         ld c, a
8241 ed 78                      in a, (c)               ; read RR0
8243 cb 57                      bit 2, a                ; check if bit 2 is set
8245 28 f6                      jr z, sio_tx_ready_loop ; if no - check again
8247 c1                         pop bc
8248 f1                         pop af
8249 c9                         ret
824a
824a                    ; rx_ready: waits for a character to become available
824a                    ; affects: none
824a                    sio_rx_ready:
824a f5                         push af
824b c5                         push bc
824c                    sio_rx_ready_loop:
824c 3a 4a 84                   ld a, (sio_ctrl)
824f 4f                         ld c, a
8250 ed 78                      in a, (c)               ; read RR0
8252 cb 47                      bit 0, a                ; check if bit 0 is set
8254 28 f6                      jr z, sio_rx_ready_loop ; if no - rx buffer has no data => check again
8256 c1                         pop bc
8257 f1                         pop af
8258 c9                         ret
8259
8259                    ; sends byte in reg A
8259                    ; affects: none
8259                    putc:
8259 c5                         push bc
825a f5                         push af
825b cd 3b 82                   call sio_tx_ready
825e 3a 4b 84                   ld a, (sio_data)
8261 4f                         ld c, a
8262 f1                         pop af
8263 ed 79                      out (c), a              ; write character
8265 c1                         pop bc
8266 c9                         ret
8267
8267                    ; getc: waits for a byte to be available and reads it
8267                    ; returns: A - read byte
8267                    getc:
8267 c5                         push bc
8268 cd 4a 82                   call sio_rx_ready
826b 3a 4b 84                   ld a, (sio_data)
826e 4f                         ld c, a
826f ed 78                      in a, (c)               ; read character
8271 c1                         pop bc
8272 c9                         ret
8273
8273                    ; getkey: gets a byte if available and reads it
8273                    ; returns: A - read byte or 0 if no byte available
8273                    getkey:
```

```
8273 c5                         push bc
8274 3a 4a 84                   ld a, (sio_ctrl)
8277 4f                         ld c, a
8278 ed 78                      in a, (c)               ; read RR0
827a cb 47                      bit 0, a                ; check if bit 0 is set
827c 28 08                      jr z, no_key            ; if no - rx buffer has no data => return 0
827e 3a 4b 84                   ld a, (sio_data)
8281 4f                         ld c, a
8282 ed 78                      in a, (c)               ; read character
8284 c1                         pop bc
8285 c9                         ret
8286                    no_key:
8286 3e 00                      ld a, 0
8288 c1                         pop bc
8289 c9                         ret
828a
828a                    ; print_newline: prints a CR/LF pair to advance to the next line
828a                    ; affects: none
828a                    print_newline:
828a f5                         push af
828b 3e 0d                      ld a, CR                ; print Carriage Return
828d cd 59 82                   call putc
8290 3e 0a                      ld a, LF                ; print Line Feed
8292 cd 59 82                   call putc
8295 f1                         pop af
8296 c9                         ret
8297
8297                    ; print_string: prints a string which starts at adress HL
8297                    ; and is terminated by EOS-character
8297                    ; affects: none
8297                    print_string:
8297 f5                         push af
8298 e5                         push hl
8299                    print_string_1:
8299 7e                         ld a,(hl)               ; load next character
829a fe 00                      cp 0                    ; is it en End Of String - character?
829c 28 06                      jr z, print_string_2    ; yes - return
829e cd 59 82                   call putc               ; no - print character
82a1 23                         inc hl                  ; HL++
82a2 18 f5                      jr print_string_1       ; do it again
82a4                    print_string_2:
82a4 e1                         pop hl
82a5 f1                         pop af
82a6 c9                         ret
82a7
82a7                    ; pio_bits_out: output bitpattern on PIO ports, then shift the pattern left
82a7                    ; affects: none
82a7                    pio_bits_out:
82a7 f5                         push af
82a8 3a 4c 84                   ld a, (pio_out)
82ab d3 10                      out (PIO_A_DATA), a
82ad d3 11                      out (PIO_B_DATA), a
82af 07                         rlca
82b0 32 4c 84                   ld (pio_out), a
82b3 f1                         pop af
82b4 c9                         ret
82b5
82b5                    ; blink: the blink routine blinks the MEMSEL LED
82b5                    ; the number of times to blink in register A
82b5                    ; Must run in upper 32KB memory
82b5
82b5                    ; Blink LED a number of times
82b5                    ; using: A
82b5                    blink:
82b5 32 50 84                   ld (ledblinks),a
82b8                    blinkloop:
82b8 cd d8 82                   call blinkled
82bb 3a 50 84                   ld a,(ledblinks)
82be 3d                         dec a
82bf 32 50 84                   ld (ledblinks),a
82c2 c8                         ret z
82c3 18 f3                      jr blinkloop
82c5
82c5                    ; Make delay wait a number of times
82c5                    ; using: A
82c5                    delay:
82c5 32 51 84                   ld (delays),a
82c8                    delayloop:
82c8 cd ed 82                   call bdelay
82cb cd ed 82                   call bdelay
82ce 3a 51 84                   ld a,(delays)
```

```
82d1 3d                         dec a
82d2 32 51 84                   ld (delays),a
82d5 c8                         ret z
82d6 18 f0                      jr delayloop
82d8
82d8                    ; Blink MEMSEL LED once
82d8                    blinkled:
82d8 3e 01                      ld a,1                  ; value is ignored when writing
82da                                                    ; it is for the benefit of the logic analyzer
82da d3 04                      out (MEMLORAM),a        ; select RAM in lower 32KB address range, LED on
82dc cd ed 82                   call bdelay
82df cd ed 82                   call bdelay
82e2 3e 00                      ld a,0
82e4 d3 00                      out (MEMEPROM),a        ; select EPROM in lower 32KB address range, LED
off
82e6 cd ed 82                   call bdelay
82e9 cd ed 82                   call bdelay
82ec c9                         ret
82ed
82ed                    ; Suitable delay for blinking LED and waiting
82ed                    bdelay:
82ed e5                         push hl
82ee 21 40 1f                   ld hl,8000      ; number of loops to delay between blinks
82f1 22 4e 84                   ld (loopcnt),hl
82f4                    bdelayloop:
82f4 2a 4e 84                   ld hl,(loopcnt)
82f7 2b                         dec hl
82f8 7c                         ld a,h
82f9 b5                         or l
82fa ca 03 83                   jp z,bleaveloop
82fd 22 4e 84                   ld (loopcnt),hl
8300 c3 f4 82                   jp bdelayloop
8303                    bleaveloop:
8303 e1                         pop hl
8304 c9                         ret
8305
8305                    ; Messages to send on serial channels
8305                    ver_msg:
8305 ..                         db "Z80 computer board, z80test version 1.0"
832c                            include "built.z80"
832c ..                     db ", Built 2021-05-25 16:12"
# End of file built.z80
8344 00                         db 0
8345                    a_msg:
8345 .. 00                      db "Output on SIO channal A", 0
835d                    a_in_msg:
835d .. 00                      db " <- input on SIO channal A", 0
8378                    b_msg:
8378 .. 00                      db "Output on SIO channal B", 0
8390                    b_in_msg:
8390 .. 00                      db " <- input on SIO channal B", 0
83ab                    lo_ram_tst_start:
83ab .. 00                      db "Testing low RAM memory", 0
83c2                    lo_ram_tst_ok:
83c2 .. 00                      db "Low RAM memory test is ok", 0
83dc                    lo_ram_tst_err:
83dc .. 00                      db "Error in low RAM memory test", 0
83f9                    hi_ram_tst_start:
83f9 .. 00                      db "Testing high RAM memory", 0
8411                    hi_ram_tst_ok:
8411 .. 00                      db "High RAM memory test is ok", 0
842c                    hi_ram_tst_err:
842c .. 00                      db "Error in high RAM memory test", 0
844a
844a                    ; Variables
844a                    sio_ctrl:
844a 00                         db 0
844b                    sio_data:
844b 00                         db 0
844c                    pio_out:
844c 00                         db 0
844d                    keyin:
844d 00                         db 0
844e                    loopcnt:
844e 00 00                      dw 0
8450                    ledblinks:
8450 00                         db 0
8451                    delays:
8451 00                         db 0
8452                    ramerr:
8452 00                         db 0
```

```
8453                    tests:
8453 00                         db 0
8454
8454                    ; Reserve space for stack
8454 00...                      ds 4096
9454                    stacktop:
9454
9454                    ; Start of high RAM test
9454
9454                    hiramstart:
9454
9454
# End of file z80hightest.z80
9454
```

SIO channal A output

```
Z80 computer board, z80test version 1.0, Built 2021-05-25 16:12
Output on SIO channal A
Testing low RAM memory
Low RAM memory test is ok
Testing high RAM memory
High RAM memory test is ok

Z80 computer board, z80test version 1.0, Built 2021-05-25 16:12
Output on SIO channal A
kkk <- input on SIO channal A
Testing low RAM memory
Low RAM memory test is ok
Testing high RAM memory
High RAM memory test is ok
```

SIO channal B output

```
Z80 computer board, z80test version 1.0, Built 2021-05-25 16:12
Output on SIO channal B
Testing low RAM memory
Low RAM memory test is ok
Testing high RAM memory
High RAM memory test is ok

Z80 computer board, z80test version 1.0, Built 2021-05-25 16:12
Output on SIO channal B
hhh <- input on SIO channal B
Testing low RAM memory
Low RAM memory test is ok
Testing high RAM memory
High RAM memory test is ok
```

Test Z80 computer memory, i/o devices and interrupt

2021-05-29

Makefile

```
hal@Perceval:~/z80_computer/tstprg$ cat Makefile
z80test.bin : z80test.z80 z80hightest.bin
        z80asm --list=z80test.lst --output=z80test.bin z80test.z80

z80hightest.bin : z80hightest.z80
        date +'    db ", Built %F %R"' > built.z80
        z80asm --list=z80hightest.lst --output=z80hightest.bin z80hightest.z80
```

Assembling the test program:

```
hal@Perceval:~/z80_computer/tstprg$ make
date +'    db ", Built %F %R"' > built.z80
z80asm --list=z80hightest.lst --output=z80hightest.bin z80hightest.z80
z80asm --list=z80test.lst --output=z80test.bin z80test.z80
```

Main program, copying the test program to RAM and executing it:

```
hal@Perceval:~/z80_computer/tstprg$ cat z80test.lst
# File z80test.z80
0000                    ; Test Z80 computer hardware, copies code from EPROM to
0000                    ; RAM and then runs it.
0000                    ;
0000                    ; You are free to use, modify, and redistribute
0000                    ; this source code. The software is provided "as is",
0000                    ; without warranty of any kind.
0000                    ; Hastily Cobbled Together 2021 by Hans-Ake Lund.
0000                    ;
0000
0000                    HIRAM:  equ 0x8000
0000
0000                            org 0x0000
0000
0000                    ; The program starts here at boot
0000                    boot:
0000
0000                    ; copy the main program to high RAM at 0x8000
0000 01 a7 0d                   ld bc,prgend - prgstart
0003 21 19 00                   ld hl,prgstart
0006 11 00 80                   ld de,HIRAM
0009                    cploop:
0009 78                         ld a,b
000a b1                         or c
000b ca 16 00                   jp z,cpend
000e 7e                         ld a,(hl)
000f 23                         inc hl
0010 12                         ld (de),a
0011 13                         inc de
0012 0b                         dec bc
0013 c3 09 00                   jp cploop
0016                    cpend:
0016 c3 00 80                   jp HIRAM        ; jump to the copied code
0019
0019                    ; Main program to load in high RAM
0019                    prgstart:
0019                            incbin "z80hightest.bin"
0dc0                    prgend:
# End of file z80test.z80
0dc0
```

Test program:

```
hal@Perceval:~/z80_computer/tstprg$ cat z80hightest.lst
# File z80hightest.z80
0000                    ; Test memory and i/o devices on home built Z80 computer board
0000                    ;
0000                    ; You are free to use, modify, and redistribute
0000                    ; this source code. The software is provided "as is",
0000                    ; without warranty of any kind.
0000                    ; Hastily Cobbled Together 2021 by Hans-Ake Lund.
0000                    ;
0000
0000                    ; Character definitions
0000                    ;
0000                    EOS:            equ 0x00        ; End Of String
0000                    CR:             equ 0x0d        ; Carriage Return (ENTER)
0000                    LF:             equ 0x0a        ; Line Feed
0000                    SPACE:          equ 0x20        ; Space
0000                    TAB:            equ 0x09        ; Tabulator
0000
0000                    ; Port definitions for switching between low EPROM and RAM
0000                    MEMEPROM:       equ 0x00
```

```
0000                    MEMLORAM:       equ 0x04
0000
0000                    ; Port definitions for the SIO/0 chip
0000                    SIO_BASE:       equ 0x08
0000                    SIO_A_DATA:     equ SIO_BASE + 0 + 0
0000                    SIO_A_CTRL:     equ SIO_BASE + 0 + 2
0000                    SIO_B_DATA:     equ SIO_BASE + 1 + 0
0000                    SIO_B_CTRL:     equ SIO_BASE + 1 + 2
0000
0000                    ; Port definitions for the CTC chip
0000                    CTC_BASE:       equ 0x0c
0000                    CTC_CH0:        equ CTC_BASE + 0
0000                    CTC_CH1:        equ CTC_BASE + 1
0000                    CTC_CH2:        equ CTC_BASE + 2
0000                    CTC_CH3:        equ CTC_BASE + 3
0000
0000                    ; Port definitions for the PIO chip
0000                    PIO_BASE:       equ 0x10
0000                    PIO_A_DATA:     equ PIO_BASE + 0 + 0
0000                    PIO_A_CTRL:     equ PIO_BASE + 0 + 2
0000                    PIO_B_DATA:     equ PIO_BASE + 1 + 0
0000                    PIO_B_CTRL:     equ PIO_BASE + 1 + 2
0000
0000                    ; Divide constant in CTC to get an approximate baudrate of 9600
0000                    ; To get 9600 baud with a 4MHz xtal oscillator the divide constant
0000                    ; should be 4000000/(9600*2*16) = 13.0208
0000                    ; Using the CTC divider constant set to 13 will give a baud-rate
0000                    ; of 4000000/(2*16*13) = 9615 baud which hopefully is close enough.
0000                    ; This is tested and works with a 9600 baudrate connection to a Linux PC.
0000                    ;
0000                    ; If this is not exact enough, another xtal oscillator must be selected,
0000                    ; it should have the frequency: 3.6864 MHz
0000                    ; The divide constant will then be set to 12 which gives the baudrate
0000                    ; of 3686400/(2*16*12) = 9600 baud.
0000                    BAUDDIV:        equ 13
0000
0000                    ; The program is copied to high RAM where it is executed
0000                    HIRAM:  equ 0x8000
0000
0000                    ; The program starts here when copied from EPROM at boot
0000                            org HIRAM
8000
8000                    boot:
8000 31 a7 8d                   ld sp,stacktop  ; initialize stack pointer
8003
8003 3e 01                      ld a,1          ; one LED blink after initial start
8005 cd fd 82                   call blink
8008 3e 05                      ld a,5
800a cd 0d 83                   call delay
800d
800d cd 35 82                   call ctc_init
8010 3e 02                      ld a,2          ; two LED blinks after CTC init
8012 cd fd 82                   call blink
8015 3e 05                      ld a,5
8017 cd 0d 83                   call delay
801a
801a cd 5a 82                   call sio_init
801d 3e 03                      ld a,3          ; three LED blinks after SIO init
801f cd fd 82                   call blink
8022 3e 05                      ld a,5
8024 cd 0d 83                   call delay
8027
8027 cd 24 82                   call pio_init
802a 3e 04                      ld a,4          ; four LED blinks after PIO init
802c cd fd 82                   call blink
802f 3e 05                      ld a,5
8031 cd 0d 83                   call delay
8034
8034 3e 11                      ld a, 00010001b ; bit pattern to output on PIO
8036 32 9e 85                   ld (pio_out),a
8039
8039 cd 77 82                   call sel_a_sio
803c cd e0 82                   call print_newline
803f 21 1c 84                   ld hl, ver_msg
8042 cd ed 82                   call print_string
8045 cd e0 82                   call print_newline
8048 cd 84 82                   call sel_b_sio
804b cd e0 82                   call print_newline
804e 21 1c 84                   ld hl, ver_msg
8051 cd ed 82                   call print_string
8054 cd e0 82                   call print_newline
```

```
8057
8057                    ; Initialize interupt mode 2 and enable interupt
8057 ed 5e                      im 2
8059 3e 84                      ld a, ivblock / 256
805b ed 47                      ld i, a
805d fb                         ei
805e
805e                    ; This test loop goes on "forever"
805e                    testloop:
805e 3e 00                      ld a, 0         ; reset interrupt indicator
8060 32 a6 85                   ld (gotint), a
8063
8063                    ; test SIO channel A
8063 cd 77 82                   call sel_a_sio
8066 cd e0 82                   call print_newline
8069 21 1c 84                   ld hl, ver_msg
806c cd ed 82                   call print_string
806f cd e0 82                   call print_newline
8072 21 5c 84                   ld hl, a_msg
8075 cd ed 82                   call print_string
8078 cd e0 82                   call print_newline
807b 3e 00                      ld a, 0
807d 32 9f 85                   ld (keyin), a
8080                    nxt_a_key:
8080 cd c9 82                   call getkey     ; test if any input character available
8083 b7                         or a
8084 ca 92 80                   jp z, no_more_a_keys
8087 cd af 82                   call putc
808a 3e 01                      ld a, 1
808c 32 9f 85                   ld (keyin), a
808f c3 80 80                   jp nxt_a_key
8092                    no_more_a_keys:
8092 3a 9f 85                   ld a, (keyin)
8095 b7                         or a
8096 ca a2 80                   jp z, no_a_key
8099 21 74 84                   ld hl, a_in_msg
809c cd ed 82                   call print_string
809f cd e0 82                   call print_newline
80a2                    no_a_key:
80a2 3e 05                      ld a,5          ; five LED blinks after sending on SIO A
80a4 cd fd 82                   call blink
80a7 3e 05                      ld a,5
80a9 cd 0d 83                   call delay
80ac
80ac                    ; test SIO channel B
80ac cd 84 82                   call sel_b_sio
80af cd e0 82                   call print_newline
80b2 21 1c 84                   ld hl, ver_msg
80b5 cd ed 82                   call print_string
80b8 cd e0 82                   call print_newline
80bb 21 8f 84                   ld hl, b_msg
80be cd ed 82                   call print_string
80c1 cd e0 82                   call print_newline
80c4 3e 00                      ld a, 0
80c6 32 9f 85                   ld (keyin), a
80c9                    nxt_b_key:
80c9 cd c9 82                   call getkey     ; test if any input character available
80cc b7                         or a
80cd ca db 80                   jp z, no_more_b_keys
80d0 cd af 82                   call putc
80d3 3e 01                      ld a, 1
80d5 32 9f 85                   ld (keyin), a
80d8 c3 c9 80                   jp nxt_b_key
80db                    no_more_b_keys:
80db 3a 9f 85                   ld a, (keyin)
80de b7                         or a
80df ca eb 80                   jp z, no_b_key
80e2 21 a7 84                   ld hl, b_in_msg
80e5 cd ed 82                   call print_string
80e8 cd e0 82                   call print_newline
80eb                    no_b_key:
80eb 3e 06                      ld a, 6                 ; six LED blinks after sending on SIO B
80ed cd fd 82                   call blink
80f0 3e 05                      ld a, 5
80f2 cd 0d 83                   call delay
80f5
80f5                    ; Test RAM
80f5 21 c2 84                   ld hl, lo_ram_tst_start
80f8 cd 77 82                   call sel_a_sio
80fb cd ed 82                   call print_string
80fe cd e0 82                   call print_newline
```

```
8101 cd 84 82                   call sel_b_sio
8104 cd ed 82                   call print_string
8107 cd e0 82                   call print_newline
810a cd a6 81                   call test_low_ram
810d 21 d9 84                   ld hl, lo_ram_tst_ok
8110 3a a4 85                   ld a,(ramerr)           ; was there an error?
8113 b7                         or a
8114 ca 1a 81                   jp z, lo_ram_prt        ; no error
8117 21 f3 84                   ld hl, lo_ram_tst_err
811a                    lo_ram_prt:
811a cd 77 82                   call sel_a_sio
811d cd ed 82                   call print_string
8120 cd e0 82                   call print_newline
8123 cd 84 82                   call sel_b_sio
8126 cd ed 82                   call print_string
8129 cd e0 82                   call print_newline
812c 3e 07                      ld a, 7                 ; seven LED blinks after testing low RAM
812e cd fd 82                   call blink
8131 3e 05                      ld a,5
8133 cd 0d 83                   call delay
8136
8136 21 10 85                   ld hl, hi_ram_tst_start
8139 cd 77 82                   call sel_a_sio
813c cd ed 82                   call print_string
813f cd e0 82                   call print_newline
8142 cd 84 82                   call sel_b_sio
8145 cd ed 82                   call print_string
8148 cd e0 82                   call print_newline
814b cd e5 81                   call test_high_ram
814e 21 28 85                   ld hl, hi_ram_tst_ok
8151 3a a4 85                   ld a,(ramerr)           ; was there an error?
8154 b7                         or a
8155 ca 5b 81                   jp z, hi_ram_prt        ; no error
8158 21 43 85                   ld hl, hi_ram_tst_err
815b                    hi_ram_prt:
815b cd 77 82                   call sel_a_sio
815e cd ed 82                   call print_string
8161 cd e0 82                   call print_newline
8164 cd 84 82                   call sel_b_sio
8167 cd ed 82                   call print_string
816a cd e0 82                   call print_newline
816d 3e 08                      ld a, 8                 ; eight LED blinks after testing high RAM
816f cd fd 82                   call blink
8172 3e 05                      ld a, 5
8174 cd 0d 83                   call delay
8177
8177                    ; Test if interupt recieved on SIO A
8177 cd 77 82                   call sel_a_sio
817a 21 7d 85                   ld hl, no_int_msg
817d 3a a6 85                   ld a, (gotint)  ; Interupt recieved?
8180 b7                         or a
8181 ca 87 81                   jp z, prtinta    ; no
8184 21 61 85                   ld hl, int_msg
8187                    prtinta:
8187 cd ed 82                   call print_string
818a cd e0 82                   call print_newline
818d
818d                    ; Test if interupt recieved on SIO B
818d cd 84 82                   call sel_b_sio
8190 21 7d 85                   ld hl, no_int_msg
8193 3a a6 85                   ld a, (gotint)  ; Interupt recieved?
8196 b7                         or a
8197 ca 9d 81                   jp z, prtintb    ; no
819a 21 61 85                   ld hl, int_msg
819d                    prtintb:
819d cd ed 82                   call print_string
81a0 cd e0 82                   call print_newline
81a3
81a3 c3 5e 80                   jp testloop
81a6
81a6                    ; test_low_ram: test lower 32K RAM memory
81a6                    test_low_ram:
81a6 3e 00                      ld a, 0                 ; reset error flag
81a8 32 a4 85                   ld (ramerr), a
81ab 3e 01                      ld a, 1
81ad d3 04                      out (MEMLORAM), a       ; select RAM in lower 32KB address range, LED on
81af 3e 05                      ld a, 5                 ; test a couple of times
81b1 32 a5 85                   ld (tests), a
81b4                    tstlolop:
81b4 01 00 80                   ld bc, 0x8000           ; number of bytes to test
81b7 21 00 00                   ld hl, 0x0000           ; start address of test
```

```
81ba                    tstlo:
81ba 1e 00                      ld e, 0x00
81bc 73                         ld (hl), e
81bd 7e                         ld a, (hl)
81be bb                         cp e
81bf ca c7 81                   jp z, tstloff
81c2 3e 01                      ld a, 1
81c4 32 a4 85                   ld (ramerr), a
81c7                    tstloff:
81c7 1e ff                      ld e, 0xff
81c9 73                         ld (hl), e
81ca 7e                         ld a, (hl)
81cb bb                         cp e
81cc ca d4 81                   jp z, tstlonxt
81cf 3e 01                      ld a, 1
81d1 32 a4 85                   ld (ramerr), a
81d4                    tstlonxt:
81d4 23                         inc hl
81d5 0b                         dec bc
81d6 78                         ld a, b
81d7 b1                         or c
81d8 20 e0                      jr nz, tstlo
81da 3a a5 85                   ld a, (tests)
81dd 3d                         dec a
81de 32 a5 85                   ld (tests),a
81e1 b7                         or a
81e2 20 d0                      jr nz, tstlolop
81e4 c9                         ret
81e5
81e5                    ; test_high_ram: test higher 32K RAM memory
81e5                    test_high_ram:
81e5 3e 00                      ld a, 0                 ; reset error flag
81e7 32 a4 85                   ld (ramerr), a
81ea 3e 00                      ld a, 0
81ec d3 00                      out (MEMEPROM), a       ; select EPROM in lower 32KB address range, LED
on
81ee                                                    ; LED indicates that higher RAM is tested
81ee 3e 05                      ld a, 5                 ; test a couple of times
81f0 32 a5 85                   ld (tests), a
81f3                    tsthilop:
81f3 01 58 72                   ld bc, 0xffff - hiramstart      ; number of bytes to test
81f6 21 a7 8d                   ld hl, hiramstart       ; start address of test, after code and variables
81f9                    tsthi:
81f9 1e 00                      ld e, 0x00
81fb 73                         ld (hl), e
81fc 7e                         ld a, (hl)
81fd bb                         cp e
81fe ca 06 82                   jp z, tsthiff
8201 3e 01                      ld a, 1
8203 32 a4 85                   ld (ramerr), a
8206                    tsthiff:
8206 1e ff                      ld e, 0xff
8208 73                         ld (hl), e
8209 7e                         ld a, (hl)
820a bb                         cp e
820b ca 13 82                   jp z, tsthinxt
820e 3e 01                      ld a, 1
8210 32 a4 85                   ld (ramerr), a
8213                    tsthinxt:
8213 23                         inc hl
8214 0b                         dec bc
8215 78                         ld a, b
8216 b1                         or c
8217 20 e0                      jr nz, tsthi
8219 3a a5 85                   ld a, (tests)
821c 3d                         dec a
821d 32 a5 85                   ld (tests), a
8220 b7                         or a
8221 20 d0                      jr nz, tsthilop
8223 c9                         ret
8224
8224                    ; pio_init: initialize PIO channel A and B for output (Mode 0)
8224                    ; affects: A
8224                    pio_init:
8224 3e 0f                      ld a, 00001111b         ; mode 0
8226 d3 12                      out (PIO_A_CTRL), a
8228 3e 07                      ld a, 00000111b         ; int disable
822a d3 12                      out (PIO_A_CTRL), a
822c 3e 0f                      ld a, 00001111b         ; mode 0
822e d3 13                      out (PIO_B_CTRL), a
8230 3e 07                      ld a, 00000111b         ; int disable
```

```
8232 d3 13                      out (PIO_B_CTRL), a
8234 c9                         ret
8235
8235                    ; ctc_init: initializes the CTC channel 0 for baudrate clock to SIO/0
8235                    ; initializes also CTC channels 1 & 2
8235                    ; input TRG0-2 is supplied by the BCLK signal which is the system clock
8235                    ; divided by 2 by the ATF22V10C
8235                    ; affects: A
8235                    ctc_init:
8235                            ; CTC chan 0
8235 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
8237                                                    ; edge don't care, time trigger don't care,
8237                                                    ; time constant follows, sw reset,
8237                                                    ; this is a ctrl cmd
8237 d3 0c                      out (CTC_CH0), a
8239 3e 0d                      ld a, BAUDDIV           ; divide constant to get baudrate
823b d3 0c                      out (CTC_CH0), a
823d                            ; Interupt vector is written to chan 0
823d 3e 10                      ld a, ctciv & 0xf8      ; interupt vector for device
823f d3 0c                      out (CTC_CH0), a
8241
8241                            ; CTC chan 1
8241 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
8243                                                    ; edge don't care, time trigger don't care,
8243                                                    ; time constant follows, sw reset,
8243                                                    ; this is a ctrl cmd
8243 d3 0d                      out (CTC_CH1), a
8245 3e 0a                      ld a, 10                ; divide BCLK by 10
8247 d3 0d                      out (CTC_CH1), a
8249
8249                            ; CTC chan 2
8249 3e 47                      ld a, 01000111b         ; int off, counter on, prescaler don't care,
824b                                                    ; edge don't care, time trigger don't care,
824b                                                    ; time constant follows, sw reset,
824b                                                    ; this is a ctrl cmd
824b d3 0e                      out (CTC_CH2), a
824d 3e 64                      ld a, 100               ; divide BCLK by 100
824f d3 0e                      out (CTC_CH2), a
8251
8251                            ; CTC chan 3 setup
8251 3e c7                      ld a, 11000111b         ; int enabled, counter on, prescaler don't car>
8253                                                    ; edge don't care, time trigger don't care,
8253                                                    ; time constant follows, sw reset,
8253                                                    ; this is a ctrl cmd
8253 d3 0f                      out (CTC_CH3), a
8255 3e fe                      ld a, 254                ; divide BCLK by 254
8257 d3 0f                      out (CTC_CH3), a
8259
8259 c9                         ret
825a
825a                    ; sio_init: initializes the SIO/0 for serial communication
825a                    ; affects: HL, B, C
825a                    sio_init:
825a                            ; load B with number of bytes
825a 06 0a                      ld b, sio_init_data_end - sio_init_data
825c 21 6d 82                   ld hl, sio_init_data    ; HL points to start of data
825f 0e 0a                      ld c, SIO_A_CTRL        ; I/O-port A for write
8261 ed b3                      otir                    ; block write of B bytes to [C] starting from HL
8263
8263                            ; load B with number of bytes
8263 06 0a                      ld b, sio_init_data_end - sio_init_data
8265 21 6d 82                   ld hl, sio_init_data    ; HL points to start of data
8268 0e 0b                      ld c, SIO_B_CTRL        ; I/O-port B for write
826a ed b3                      otir                    ; block write of B bytes to [C] starting from HL
826c
826c c9                         ret
826d
826d                    sio_init_data:
826d 30                         db 00110000b            ; write to WR0: error reset
826e 18                         db 00011000b            ; write to WR0: channel reset
826f 04 44                      db 0x04, 01000100b      ; write to WR4: clkx16, 1 stop bit, no parity
8271 05 68                      db 0x05, 01101000b      ; write to WR5: DTR inactive, enable TX 8bit,
8273                                                    ; BREAK off, TX on, RTS inactive
8273 01 00                      db 0x01, 00000000b      ; write to WR1: no interrupts enabled
8275 03 c1                      db 0x03, 11000001b      ; write to WR3: enable RX 8bit
8277                    sio_init_data_end:
8277
8277                    ; sel_a_sio: selects SIO channel A for i/o
8277                    ; affects: none
8277                    sel_a_sio:
8277 f5                         push af
```

```
8278 3e 08                      ld a, SIO_A_DATA
827a 32 9d 85                   ld (sio_data), a
827d 3e 0a                      ld a, SIO_A_CTRL
827f 32 9c 85                   ld (sio_ctrl), a
8282 f1                         pop af
8283 c9                         ret
8284
8284                    ; sel_b_sio: selects SIO channel B for i/o
8284                    ; affects: none
8284                    sel_b_sio:
8284 f5                         push af
8285 3e 09                      ld a, SIO_B_DATA
8287 32 9d 85                   ld (sio_data), a
828a 3e 0b                      ld a, SIO_B_CTRL
828c 32 9c 85                   ld (sio_ctrl), a
828f f1                         pop af
8290 c9                         ret
8291
8291                    ; tx_ready: waits for transmitt buffer to become empty
8291                    ; affects: none
8291                    sio_tx_ready:
8291 f5                         push af
8292 c5                         push bc
8293                    sio_tx_ready_loop:
8293 3a 9c 85                   ld a, (sio_ctrl)
8296 4f                         ld c, a
8297 ed 78                      in a, (c)               ; read RR0
8299 cb 57                      bit 2, a                ; check if bit 2 is set
829b 28 f6                      jr z, sio_tx_ready_loop ; if no - check again
829d c1                         pop bc
829e f1                         pop af
829f c9                         ret
82a0
82a0                    ; rx_ready: waits for a character to become available
82a0                    ; affects: none
82a0                    sio_rx_ready:
82a0 f5                         push af
82a1 c5                         push bc
82a2                    sio_rx_ready_loop:
82a2 3a 9c 85                   ld a, (sio_ctrl)
82a5 4f                         ld c, a
82a6 ed 78                      in a, (c)               ; read RR0
82a8 cb 47                      bit 0, a                ; check if bit 0 is set
82aa 28 f6                      jr z, sio_rx_ready_loop ; if no - rx buffer has no data => check again
82ac c1                         pop bc
82ad f1                         pop af
82ae c9                         ret
82af
82af                    ; sends byte in reg A
82af                    ; affects: none
82af                    putc:
82af c5                         push bc
82b0 f5                         push af
82b1 cd 91 82                   call sio_tx_ready
82b4 3a 9d 85                   ld a, (sio_data)
82b7 4f                         ld c, a
82b8 f1                         pop af
82b9 ed 79                      out (c), a              ; write character
82bb c1                         pop bc
82bc c9                         ret
82bd
82bd                    ; getc: waits for a byte to be available and reads it
82bd                    ; returns: A - read byte
82bd                    getc:
82bd c5                         push bc
82be cd a0 82                   call sio_rx_ready
82c1 3a 9d 85                   ld a, (sio_data)
82c4 4f                         ld c, a
82c5 ed 78                      in a, (c)               ; read character
82c7 c1                         pop bc
82c8 c9                         ret
82c9
82c9                    ; getkey: gets a byte if available and reads it
82c9                    ; returns: A - read byte or 0 if no byte available
82c9                    getkey:
82c9 c5                         push bc
82ca 3a 9c 85                   ld a, (sio_ctrl)
82cd 4f                         ld c, a
82ce ed 78                      in a, (c)               ; read RR0
82d0 cb 47                      bit 0, a                ; check if bit 0 is set
82d2 28 08                      jr z, no_key            ; if no - rx buffer has no data => return 0
```

```
82d4 3a 9d 85                   ld a, (sio_data)
82d7 4f                         ld c, a
82d8 ed 78                      in a, (c)               ; read character
82da c1                         pop bc
82db c9                         ret
82dc                    no_key:
82dc 3e 00                      ld a, 0
82de c1                         pop bc
82df c9                         ret
82e0
82e0                    ; print_newline: prints a CR/LF pair to advance to the next line
82e0                    ; affects: none
82e0                    print_newline:
82e0 f5                         push af
82e1 3e 0d                      ld a, CR                ; print Carriage Return
82e3 cd af 82                   call putc
82e6 3e 0a                      ld a, LF                ; print Line Feed
82e8 cd af 82                   call putc
82eb f1                         pop af
82ec c9                         ret
82ed
82ed                    ; print_string: prints a string which starts at adress HL
82ed                    ; and is terminated by EOS-character
82ed                    ; affects: none
82ed                    print_string:
82ed f5                         push af
82ee e5                         push hl
82ef                    print_string_1:
82ef 7e                         ld a,(hl)               ; load next character
82f0 fe 00                      cp 0                    ; is it en End Of String - character?
82f2 28 06                      jr z, print_string_2    ; yes - return
82f4 cd af 82                   call putc               ; no - print character
82f7 23                         inc hl                  ; HL++
82f8 18 f5                      jr print_string_1       ; do it again
82fa                    print_string_2:
82fa e1                         pop hl
82fb f1                         pop af
82fc c9                         ret
82fd
82fd                    ; blink: the blink routine blinks the MEMSEL LED
82fd                    ; the number of times to blink in register A
82fd                    ; Must run in upper 32KB memory
82fd
82fd                    ; Blink LED a number of times
82fd                    ; using: A
82fd                    blink:
82fd 32 a2 85                   ld (ledblinks),a
8300                    blinkloop:
8300 cd 20 83                   call blinkled
8303 3a a2 85                   ld a,(ledblinks)
8306 3d                         dec a
8307 32 a2 85                   ld (ledblinks),a
830a c8                         ret z
830b 18 f3                      jr blinkloop
830d
830d                    ; Make delay wait a number of times
830d                    ; using: A
830d                    delay:
830d 32 a3 85                   ld (delays),a
8310                    delayloop:
8310 cd 35 83                   call bdelay
8313 cd 35 83                   call bdelay
8316 3a a3 85                   ld a,(delays)
8319 3d                         dec a
831a 32 a3 85                   ld (delays),a
831d c8                         ret z
831e 18 f0                      jr delayloop
8320
8320                    ; Blink MEMSEL LED once
8320                    blinkled:
8320 3e 01                      ld a,1                  ; value is ignored when writing
8322                                                    ; it is for the benefit of the logic analyzer
8322 d3 04                      out (MEMLORAM),a        ; select RAM in lower 32KB address range, LED on
8324 cd 35 83                   call bdelay
8327 cd 35 83                   call bdelay
832a 3e 00                      ld a,0
832c d3 00                      out (MEMEPROM),a        ; select EPROM in lower 32KB address range, LED
off
832e cd 35 83                   call bdelay
8331 cd 35 83                   call bdelay
8334 c9                         ret
```

```
8335
8335                    ; Suitable delay for blinking LED and waiting
8335                    bdelay:
8335 e5                         push hl
8336 21 40 1f                   ld hl,8000      ; number of loops to delay between blinks
8339 22 a0 85                   ld (loopcnt),hl
833c                    bdelayloop:
833c 2a a0 85                   ld hl,(loopcnt)
833f 2b                         dec hl
8340 7c                         ld a,h
8341 b5                         or l
8342 ca 4b 83                   jp z,bleaveloop
8345 22 a0 85                   ld (loopcnt),hl
8348 c3 3c 83                   jp bdelayloop
834b                    bleaveloop:
834b e1                         pop hl
834c c9                         ret
834d
834d                    ; Interupt routines, most are dummies for now
834d                    ; CTC interupts for CH0 - CH2 not used
834d                    ctcint0:
834d                    ctcint1:
834d                    ctcint2:
834d                    ; PIO interrupt routines, not used for now
834d                    piointa:
834d                    piointb:
834d                    ; SIO interrupt routines, not used for now
834d                    siointa:
834d                    siointb:
834d fb                         ei
834e ed 4d                      reti
8350
8350                    ; CTC interrupt for CH3 used to test interrupt
8350                    ; sets indicator and outputs bit pattern on PIO ports
8350                    ; then shifts the pattern left
8350                    ctcint3:
8350 f5                         push af
8351 3e 01                      ld a, 1
8353 32 a6 85                   ld (gotint), a
8356 3a 9e 85                   ld a, (pio_out)
8359 d3 10                      out (PIO_A_DATA), a
835b d3 11                      out (PIO_B_DATA), a
835d 07                         rlca
835e 32 9e 85                   ld (pio_out), a
8361 f1                         pop af
8362 fb                         ei
8363 ed 4d                      reti
8365
8365                    endofcode:
8365
8365                    ; Interupt vectors for interupt mode 2
8365                    ; make sure that the block is on an even 256 byte address
8365                    if endofcode & 0x00ff
8365 0xff...                    ds 256 - (endofcode & 0x00ff), 0xff
8400                    endif
8400
8400                    ivblock:
8400                    ;
8400                    ; The SIO interupt vector block must be on
8400                    ; an even 16 byte address if "status affects vector" is used
8400
8400                    sioiv:
8400 4d 83                      dw siointa
8402 4d 83                      dw siointa
8404 4d 83                      dw siointa
8406 4d 83                      dw siointa
8408 4d 83                      dw siointb
840a 4d 83                      dw siointb
840c 4d 83                      dw siointb
840e 4d 83                      dw siointb
8410
8410                    ; The CTC interupt vector block must be on
8410                    ; an even 8 byte address
8410                    ctciv:
8410 4d 83                      dw ctcint0
8412 4d 83                      dw ctcint1
8414 4d 83                      dw ctcint2
8416 50 83                      dw ctcint3
8418
8418                    ; The PIO interupt vectors must be on
8418                    ; an even 2 byte address
```

```
8418                    pioaiv:
8418 4d 83                      dw piointa
841a                    piobiv:
841a 4d 83                      dw piointb
841c
841c                    ; Messages to send on serial channels
841c                    ver_msg:
841c ..                         db "Z80 computer board, z80test version 1.1"
8443                            include "built.z80"
8443 ..                     db ", Built 2021-05-29 10:39"
# End of file built.z80
845b 00                         db 0
845c                    a_msg:
845c .. 00                      db "Output on SIO channal A", 0
8474                    a_in_msg:
8474 .. 00                      db " <- input on SIO channal A", 0
848f                    b_msg:
848f .. 00                      db "Output on SIO channal B", 0
84a7                    b_in_msg:
84a7 .. 00                      db " <- input on SIO channal B", 0
84c2                    lo_ram_tst_start:
84c2 .. 00                      db "Testing low RAM memory", 0
84d9                    lo_ram_tst_ok:
84d9 .. 00                      db "Low RAM memory test is ok", 0
84f3                    lo_ram_tst_err:
84f3 .. 00                      db "Error in low RAM memory test", 0
8510                    hi_ram_tst_start:
8510 .. 00                      db "Testing high RAM memory", 0
8528                    hi_ram_tst_ok:
8528 .. 00                      db "High RAM memory test is ok", 0
8543                    hi_ram_tst_err:
8543 .. 00                      db "Error in high RAM memory test", 0
8561                    int_msg:
8561 .. 00                      db "Interupt from CTC channel 3", 0
857d                    no_int_msg:
857d .. 00                      db "No interupt from CTC channel 3", 0
859c
859c                    ; Variables
859c                    sio_ctrl:
859c 00                         db 0
859d                    sio_data:
859d 00                         db 0
859e                    pio_out:
859e 00                         db 0
859f                    keyin:
859f 00                         db 0
85a0                    loopcnt:
85a0 00 00                      dw 0
85a2                    ledblinks:
85a2 00                         db 0
85a3                    delays:
85a3 00                         db 0
85a4                    ramerr:
85a4 00                         db 0
85a5                    tests:
85a5 00                         db 0
85a6                    gotint:
85a6 00                         db 0
85a7
85a7                    ; Reserve space for stack
85a7 0xff...                    ds 2048, 0xff
8da7                    stacktop:
8da7
8da7                    ; Start of high RAM test
8da7
8da7                    hiramstart:
8da7
8da7
# End of file z80hightest.z80
8da7
```

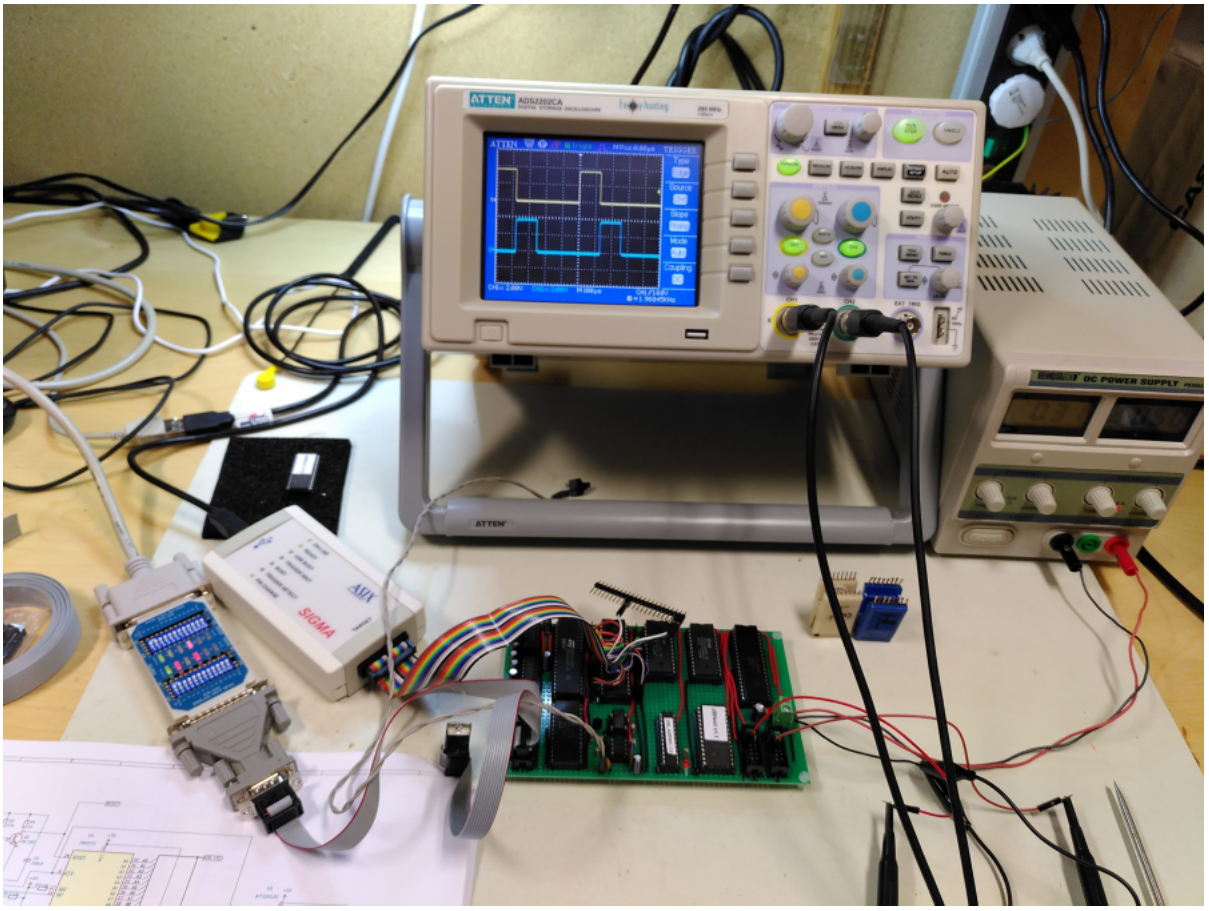
Output on SIO channel A (same on channel B):

```
Z80 computer board, z80test version 1.2, Built 2021-05-29 14:11
Output on SIO channal A
Testing low RAM memory
Low RAM memory test is ok
```

```
Testing high RAM memory
High RAM memory test is ok
Interupt from CTC channel 3

Z80 computer board, z80test version 1.2, Built 2021-05-29 14:11
Output on SIO channal A
asd <- input on SIO channal A
Testing low RAM memory
Low RAM memory test is ok
Testing high RAM memory
High RAM memory test is ok
Interupt from CTC channel 3
```

PIO output

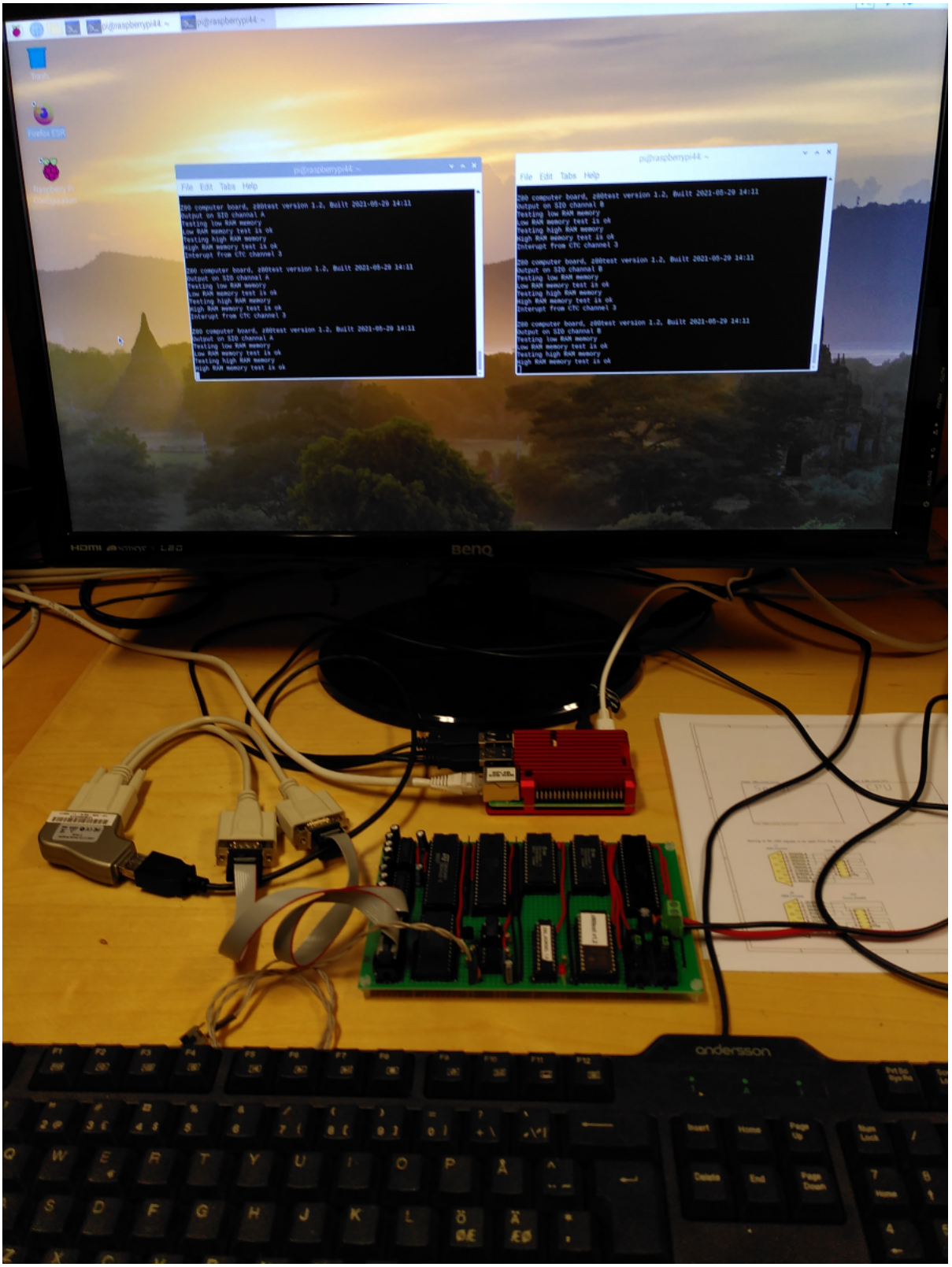


Logic Analyzer interrupt

Testing with Raspberry Pi showing output from SIO channel A and B

2021-05-29

Using a double serial to USB converter connected to the Raspberry Pi.




```
pi@raspberrypi44:~ $ minicom -b 9600 -D /dev/ttyUSB0
...

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB0, 16:31:55

Press CTRL-A Z for help on special keys


Z80 computer board, z80test version 1.2, Built 2021-05-29 14:11

Z80 computer board, z80test version 1.2, Built 2021-05-29 14:11
```

```
Output on SIO channal A
Testing low RAM memory
Low RAM memory test is ok
Testing high RAM memory
High RAM memory test is ok
Interupt from CTC channel 3
```

```
pi@raspberrypi44:~ $ minicom -b 9600 -D /dev/ttyUSB2
...

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB2, 16:25:05

Press CTRL-A Z for help on special keys


Z80 computer board, z80test version 1.2, Built 2021-05-29 14:11

Z80 computer board, z80test version 1.2, Built 2021-05-29 14:11
Output on SIO channal B
Testing low RAM memory
Low RAM memory test is ok
Testing high RAM memory
High RAM memory test is ok
Interupt from CTC channel 3
```