

# Z180 computer test programs

---

[Z180 Computer](#)

## Contents

---

### **Z180 test programs**

- Test program tools
- Lab setup for tests
- Very simple test program
- Simple test program with MMU enabled
- More thorough test program
- Test program with serial output
- Test program with serial output and input on ports 0 & 1
- Test program with MMU setup and serial output and input on ports 0 & 1
- Test program with MMU setup, RAM access and serial output and input on ports 0 & 1
- Test program of MMU setup
- Test program copied to RAM and running from there
- Test of RAM
- Test of RAM with no wait states
- Test of interrupt

### **AVR test programs**

- Simple blink and serial test

# Z180 test programs

## Test program tools

---

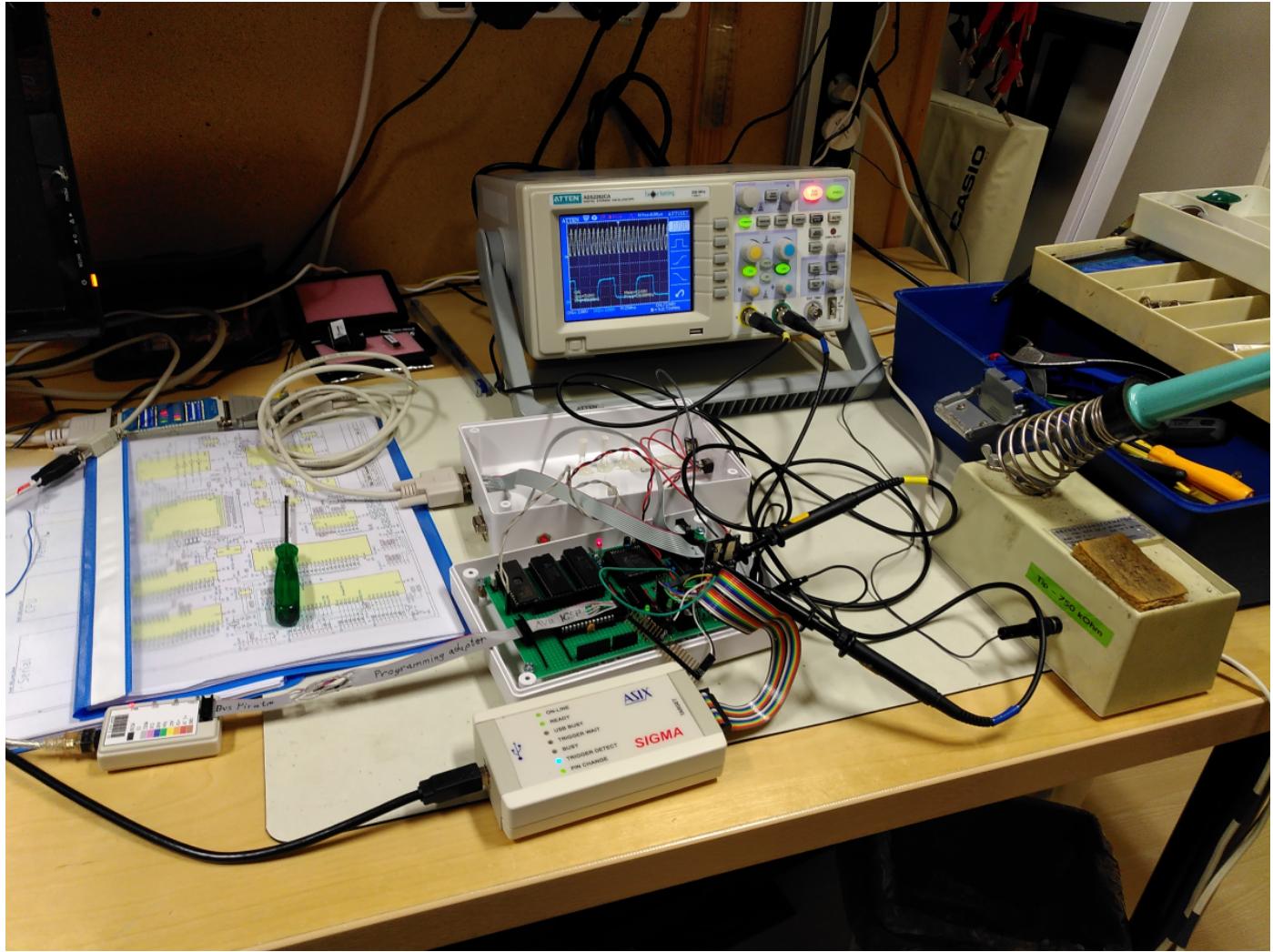
2021-08-05

The assembler z80asm is used: [AlbertVeli/z80asm: Clone of z80asm \(original is on Savannah\)](#) (<https://github.com/AlbertVeli/z80asm>)

## Lab setup for tests

---

2021-08-16



## Very simple test program

2021-08-05

Makefile:

```
blink180.bin : blink180.z80
    date +'    db ", Built %F %R"' > built180.z80
    z80asm --list=blink180.lst --output=blink180.bin blink180.z80
```

blink180.lst, updated 2021-08-09

```
# File blink180.z80
0000          ; Test program that blinks the LED
0000          ; using, no RAM, no interrupt
0000
0000
0000
0000
0000 21 ff ff
0003 3e 01
0005 d3 43
0007 b5
000a c2 07 00
000d 21 ff ff
0010 3e 00
0012 d3 42
0014
0014 2b
0015 7c
0016 b5
0017 c2 14 00
001a
001a c3 00 00
001d

        boot:
        mloop:
        ldi hl,0xffff
        ldi a,1
        out(0x43),a ; LED on
onloop:
        dec hl
        ldi a,h
        or l
        jp nz,onloop
        ldi hl,0xffff
        ldi a,0
        out(0x42),a ; LED off
offloop:
        dec hl
        ldi a,h
        or l
        jp nz,offloop
        jp mloop
```

```
001d          built:  
001d          include "built180.z80"  
001d ..        db ", Built 2021-08-08 17:25"  
# End of file built180.z80  
0035 00        db 0  
0036  
# End of file blink180.z80  
0036
```

## Program EPROM

```
L0001: Welcome to Elnec programmers control program PG4UW.  
L0002: Version 3.15h/06.2015.  
L0003:  
L0004: Today is 05.08.2021, 14:06:08.  
L0005:  
L0006: Processor: Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz Frequency: 3997\3997,77 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: >> 05.08.2021, 14:06:09  
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: >> 05.08.2021, 14:06:15  
L0021: Selected device: STMicroelectronics M27C2001.  
L0022:  
L0023: Buffer operation "Checksum", time elapsed: 16 ms  
L0024: Buffer checksum in range of [0h..3FFFFh]: 03FC0000h - 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: "000003FFFF" 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 M27C2001.  
L0047: Buffer checksum in range of [0h..3FFFFh]: 03FC0000h - Byte sum (x8), Straight  
L0048:  
L0049: >> 05.08.2021, 14:06:15  
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 05.08.2021 14:06:15  
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: >> 05.08.2021, 14:08:31  
L0062: Loading file (from "Load file" dialog window): C:\Users\hal\Desktop\EPROM\blink180.bin  
L0063: File format selection: automatic  
L0064: File format: Binary  
L0065: File loading successful. Bytes loaded 72 (00000048h).  
L0066:  
L0067: Buffer operation "Checksum", time elapsed: 15 ms  
L0068: Buffer checksum in range of [0h..3FFFFh]: 03FB01E6h - Byte sum (x8), Straight  
L0069:  
L0070: >> 05.08.2021, 14:08:55  
L0071: Opened window "Program?" (parent window "PG4UW v3.15h/06.2015 - universal control...").  
L0072:  
L0073: >> 05.08.2021, 14:09:03  
L0074: Closed window "Program?" (by pressing "Yes").  
L0075:  
L0076: >> 05.08.2021, 14:09:03  
L0077: Programming device: STMicroelectronics M27C2001.  
L0078: Buffer checksum in range of [0h..3FFFFh]: 03FB01E6h - 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: 0:02:21.0
L0085: Statistics info: Success:1 Failure:0 Other failure:0 Total:1

```

## Simple test program with MMU enabled

2121-08-13

This works, i.e. MMU regs set to reset values:

```

# File blinkm180.z80
0000 ; Test program that blinks the LED
0000 ; using, no RAM, no interrupt
0000
0000 ; Internal ports
0000 CBR: equ 0x0038 ;MMU Common Base Register
0000 BBR: equ 0x0039 ;MMU Bank Base Register
0000 CBAR: equ 0x003a ;MMU Common/Bank Area Register
0000
0000 boot:
0000 ; Set up the MMU
0000 ;
0000 ; Common Bank 0
0000 ;   logical: 0x0000 - 0x3fff
0000 ;   physical: 0x000000 - 0x03ffff
0000 ; Bank Area
0000 ;   logical: 0x4000 - 0xbffff
0000 ;   physical: 0x74000 - 0x7bffff
0000 ; Common Bank 1
0000 ;   logical: 0xc000 - 0xfffff
0000 ;   physical: 0xfc000 - 0xfffffff
0000
0000 ; test by setting MMU regs to reset values
0000 3e 00
0002 01 38 00
0005 ed 79
0007 3e 00
0009 01 39 00
000c ed 79
000e 3e ff
0010 01 3a 00
0013 ed 79
0015
0015 mloop:
0015 21 ff ff
0018 3e 01
001a d3 43
001c
001c 2b
001d 7c
001e b5
001f c2 1c 00
0022 21 ff ff
0025 3e 00
0027 d3 42
0029
0029 2b
002a 7c
002b b5
002c c2 29 00
002f
002f c3 15 00
0032
0032 built:
0032     include "built180.z80"
0032     db ", Built 2021-08-13 17:03"
# End of file built180.z80
004a 00
004b
# End of file blinkm180.z80
004b

```

But this MMU reg setup does not work:

```

...
0000 ; Internal ports
0000 CBR: equ 0x0038 ;MMU Common Base Register
0000 BBR: equ 0x0039 ;MMU Bank Base Register
0000 CBAR: equ 0x003a ;MMU Common/Bank Area Register
0000
0000 ; External ports

```

```

0000      LEDOFF: equ 0x42    ;Write turns LED off
0000      LEDON: equ 0x43    ;Write turns LED on
0000
0000      boot:
0000
0000      ; Set up the MMU
0000      ;
0000      ; Common Bank 0
0000      ;   logical: 0x0000 - 0x3fff
0000      ;   physical: 0x00000 - 0x03ffff
0000      ; Bank Area
0000      ;   logical: 0x4000 - 0xbfff
0000      ;   physical: 0x74000 - 0x7bfff
0000      ; Common Bank 1
0000      ;   logical: 0xc000 - 0xfffff
0000      ;   physical: 0xfc000 - 0xfffffff
0000
0000 3e f0      ld a,0xf0
0002 01 38 00    ld bc,CBR
0005 ed 79      out (c),a
0007 3e 70      ld a,0x70
0009 01 39 00    ld bc,BBR
000c ed 79      out (c),a
000e 3e c4      ld a,0xc4
0010 01 3a 00    ld bc,CBAR
0013 ed 79      out (c),a
...

```

2021-08-15

When testing this setup:

```

...
; test by setting MMU regs to reset values, but RAM
; starting at physical address 0x8f000
ld a,0x80
ld bc,CBR
out (c),a
ld a,0x80
ld bc,BBR
out (c),a
ld a,0xff
ld bc,CBAR
out (c),a
...

```

The logic analyser shows this:



After this code is executed.

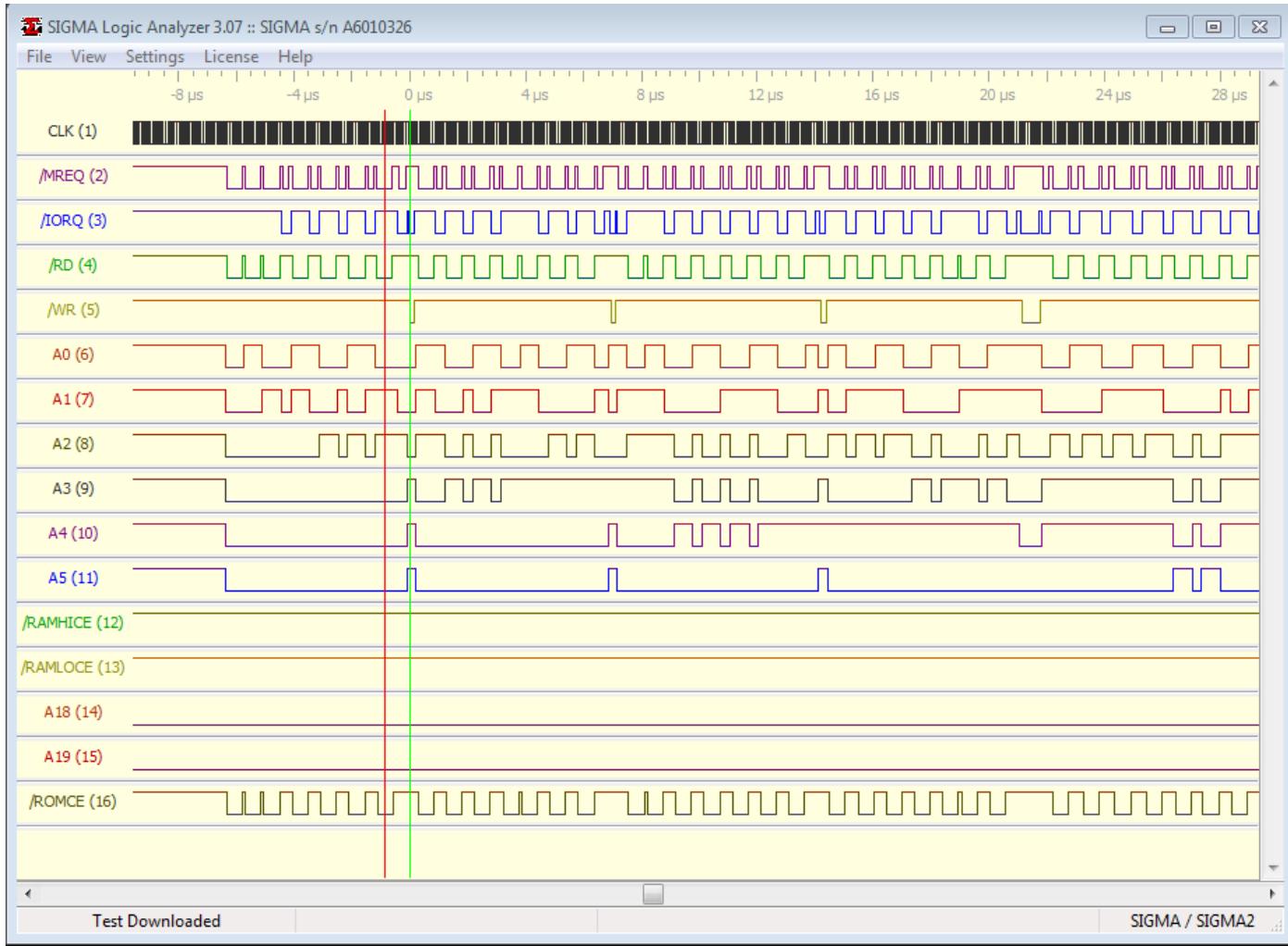
```
ld a,0x80
ld bc,BBR
out (c),a
```

the address logic goes wrong. What is the problem?

When modifying the MMU setup;

```
; test by setting MMU regs to reset values, but RAM
; starting at physical address 0x8f000
ld a,0x80
ld bc,CBR
out (c),a
ld a,0x00      <- this was changed from 0x80 to 0x00
ld bc,BBR
out (c),a
ld a,0xff
ld bc,CBAR
out (c),a
```

The LED blinks and the program works. It looks like this on the logic analyser:



This program works that in addition reads and writes to the RAM:

```

0000 ; Test program that blinks the LED
0000 ; using, setting up the MMU and touching RAM
0000
0000 ; Internal ports
0000 CBR:    equ 0x0038 ;MMU Common Base Register
0000 BBR:    equ 0x0039 ;MMU Bank Base Register
0000 CBAR:   equ 0x003a ;MMU Common/Bank Area Register
0000
0000 boot:
0000 ; Set up the MMU
0000 ;
0000 ; Common Bank 0
0000 ;    logical: 0x0000 - 0xffff
0000 ;    physical: 0x00000 - 0x0ffff
0000 ;    Bank Area, not used in this test (probably?)
0000 ; Common Bank 1
0000 ;    logical: 0xf000 - 0xffff
0000 ;    physical: 0x8f000 - 0x8ffff
0000
0000 RAMADR: equ 0xf800
0000
0000 ; test by setting MMU regs to reset values, but RAM
0000 ; starting at physical address 0x8f000
0000 3e 80
0002 01 38 00
0005 ed 79
0007 3e 00
0009 01 39 00
000c ed 79
000e 3e ff
0010 01 3a 00
0013 ed 79
0015
0015 11 00 f8
0018
0018 21 ff ff
001b 3e 01
001d d3 43
001f
001f 1a
0020 3c
0021 12

        ld a,0x80
        ld bc,CBR
        out (c),a
        ld a,0x00
        ld bc,BBR
        out (c),a
        ld a,0xff
        ld bc,CBAR
        out (c),a

        ld de,RAMADR
mloop:
        ld hl,0xffff
        ld a,1
        out(0x43),a ; LED on
onloop:
        ld a,(de)
        inc a
        ld (de),a

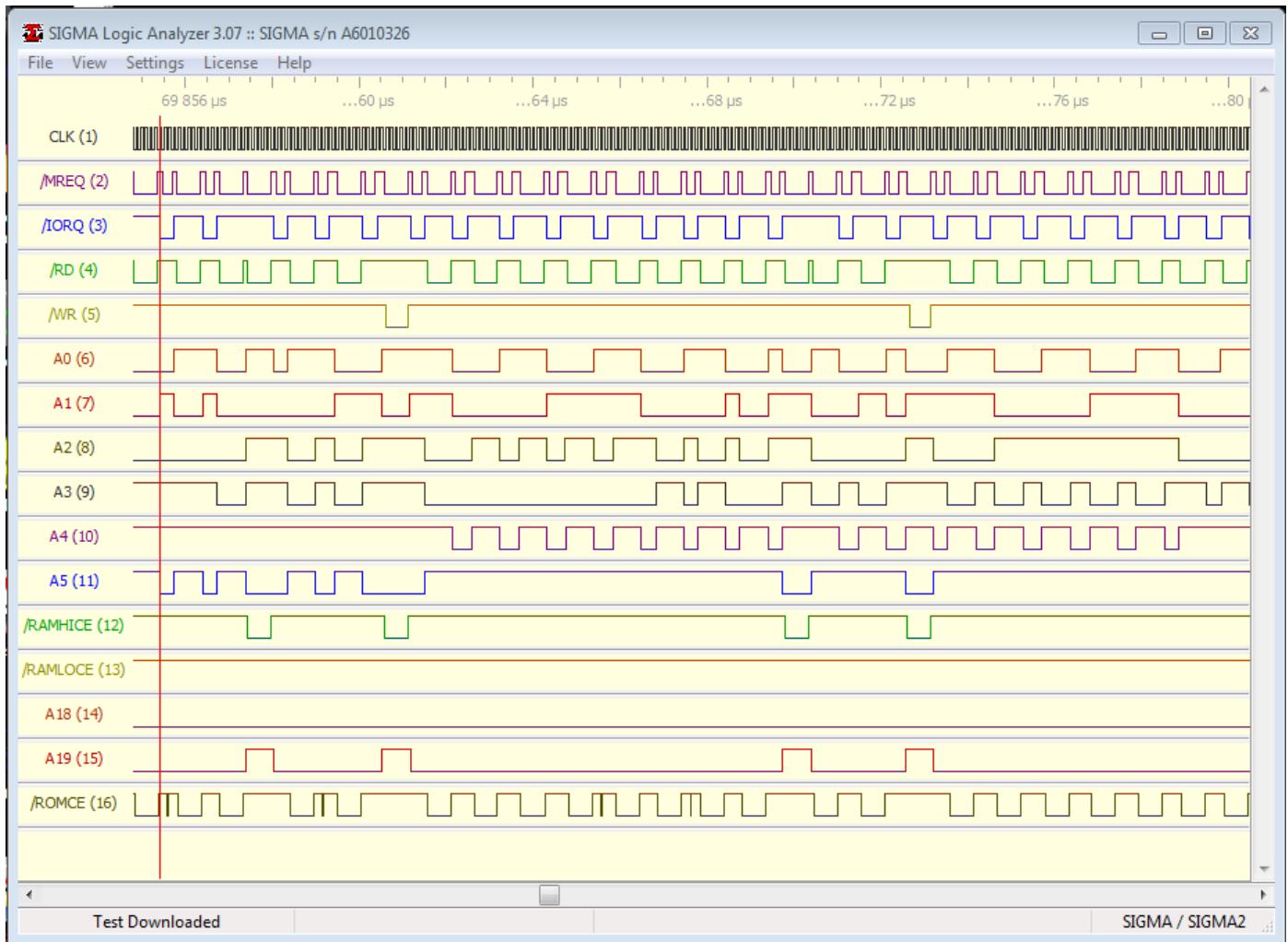
```

```

0022 1c           inc e
0023 2b           dec hl
0024 7c           ld a,h
0025 b5           or l
0026 c2 1f 00     jp nz, onloop
0029 21 ff ff     ld hl, 0xffff
002c 3e 00         ld a, 0
002e d3 42         out(0x42), a ; LED off
0030               offloop:
0030 1a           ld a, (de)
0031 3c           inc a
0032 12           ld (de), a
0033 1c           inc e
0034 2b           dec hl
0035 7c           ld a,h
0036 b5           or l
0037 c2 30 00     jp nz, offloop
003a               mloop
003d               built:
003d             include "built180.z80"
003d             db ", Built 2021-08-15 17:14"
# End of file built180.z80
0055 00           db 0
0056
# End of file blinkm180.z80
0056

```

Logic analyser:



## More thorough test program

2021-08-16

Makefile:

```
test180.bin : test180.z80
```

```

date +'    db ", Built %F %R"' > tbuilt180.z80
z80asm --list=test180.lst --output=test180.bin test180.z80

blinkm180.bin : blinkm180.z80
date +'    db ", Built %F %R"' > built180.z80
z80asm --list=blinkm180.lst --output=blinkm180.bin blinkm180.z80

blink180.bin : blink180.z80
date +'    db ", Built %F %R"' > built180.z80
z80asm --list=blink180.lst --output=blink180.bin blink180.z80

```

Program saved as test180\_2.z80

```

# File test180.z80
0000          ; Test program for the Z180 computer
0000          ; testing:
0000          ;   - MMU setup
0000          ;   - RAM as stack
0000          ; todo
0000          ;   - Serial
0000          ;   - RAM test
0000          ;   - copy test program to RAM and execute
0000          ;   - test all RAM using MEMSEL
0000
0000          ; Internal ports
0000 CBR:    equ 0x0038 ;MMU Common Base Register
0000 BBR:    equ 0x0039 ;MMU Bank Base Register
0000 CBAR:   equ 0x003a ;MMU Common/Bank Area Register
0000
0000          ; External ports
0000 LEDOFF: equ 0x42  ;Write turns LED off
0000 LEDON:  equ 0x43  ;Write turns LED on
0000
0000 boot:
0000
0000          ; Set up the MMU
0000
0000          ; Common Bank 0
0000          ;   logical: 0x0000 - 0xefff
0000          ;   physical: 0x000000 - 0x0effff
0000          ; Bank Area, not used in this test (probably?)
0000          ; Common Bank 1
0000          ;   logical: 0xf000 - 0xffff
0000          ;   physical: 0x8f000 - 0x8ffff
0000
0000 3e 80
0002 01 38 00
0005 ed 79
0007 3e 00
0009 01 39 00
000c ed 79
000e 3e ff
0010 01 3a 00
0013 ed 79
0015
0015 31 00 00
0018
0018          ; Initialize other devices
0018          ;   for now just blink LED
0018
0018 06 01
001a cd 4d 00
001d 06 03
001f cd 6a 00
0022
0022 06 02
0024 cd 4d 00
0027 06 03
0029 cd 6a 00
002c
002c 06 03
002e cd 4d 00
0031 06 03
0033 cd 6a 00
0036
0036 06 04
0038 cd 4d 00
003b 06 03
003d cd 6a 00
0040
0040 testloop:
0040 06 0a
0042 cd 4d 00
0045 06 03
0047 cd 6a 00
004a c3 40 00
004d
004d          ; ledblink flashes the LED
004d          ; number of times in B reg

```

```
004d          ; using reg: a, b, hl
004d
004d 21 ff ff    ledblink:
0050 3e 01        ld hl,0xffff
0052 d3 43        ld a,1
0054             out(LEDON),a
0054 2b          ledonLoop:
0055 7c          dec hl
0056 b5          ld a,h
0057 c2 54 00      or l
005a 21 ff ff      jp nz,ledonloop
005d 3e 00        ld hl,0xffff
005f d3 42        ld a,0
0061             out(LEDOFF),a
0061 2b          ledoffloop:
0062 7c          dec hl
0063 b5          ld a,h
0064 c2 61 00      or l
0067 10 e4        jp nz,ledoffloop
0069 c9          djnz ledblink
006a             ret
006a
006a          ; delays makes a delay multiplied
006a          ; by number of times in B reg
006a          ; using reg: a, b, hl
006a 21 ff ff    delays:
006d             ld hl,0xffff
006d 2b          delayloop:
006e 7c          dec hl
006f b5          ld a,h
0070 c2 6d 00      or l
0073 21 ff ff      jp nz,delayloop
0076             delay2loop:
0076 2b          dec hl
0077 7c          ld a,h
0078 b5          or l
0079 c2 76 00      jp nz,delay2loop
007c 10 ec        djnz delays
007e c9          ret
007f
007f          built:
007f          include "tbuilt180.z80"
007f ..          db ", Built 2021-08-16 14:05"
# End of file tbuilt180.z80
0097 00          db 0
# End of file test180.z80
0098
```

Logic analyser showing call when writing return address to RAM.



## Test program with serial output

2021-08-17

Saved as test180\_3.z80

```
# File test180.z80
0000 ; Test program for the Z180 computer
0000 ; test180.z80
0000 ;
0000 ; testing:
0000 ; - MMU setup
0000 ; - RAM as stack
0000 ; - Serial
0000 ; todo
0000 ; - RAM test
0000 ; - copy test program to RAM and execute
0000 ; - test all RAM using MEMSEL
0000 ;
0000 ; Internal ports
0000 ;
0000 ; ASCII Registers port 0 and 1
CNTLA0: equ 0x0000 ;ASCII Channel Control Register A 0
CNTLA1: equ 0x0001 ;ASCII Channel Control Register A 1
CNTLB0: equ 0x0002 ;ASCII Control Register B 0
CNTLB1: equ 0x0003 ;ASCII Control Register B 1
STAT0: equ 0x0004 ;ASCII Status Register 0
STAT1: equ 0x0005 ;ASCII Status Register 1
TDR0: equ 0x0006 ;ASCII Transmit Data Register 0
TDR1: equ 0x0007 ;ASCII Transmit Data Register 1
RDR0: equ 0x0008 ;ASCII Receive Data FIFO 0
RDR1: equ 0x0009 ;ASCII Receive Data FIFO 1
0000 ;
0000 ; MMU Registers
CBR: equ 0x0038 ;MMU Common Base Register
BBR: equ 0x0039 ;MMU Bank Base Register
CBAR: equ 0x003a ;MMU Common/Bank Area Register
0000 ;
0000 ; External ports
0000
```

```
| 0000 ;Select EPROM or RAM on address 0x0000 - 0x3fffff
| 0000 ROMSEL: equ 0x40    ;Write selects EPROM (reset condition)
| 0000 RAMSEL: equ 0x41    ;Write selects RAM
| 0000
| 0000 ;LED on/off
| 0000 LEDOFF: equ 0x42    ;Write turns LED off (reset condition)
| 0000 LEDON:  equ 0x43    ;Write turns LED on
| 0000
| 0000 ;SPI device select and AVR reset
| 0000 CSPORT: equ 0x44    ;Write to bit 0 - 3 (reset condition, all 0)
| 0000 ;Bit 0: select SD_CS0 when set to 1
| 0000 ;Bit 1: select SD_CS1 when set to 1
| 0000 ;Bit 2: select ATSS (AVR) when set to 1
| 0000 ;Bit 1: reset AVR when set to 1
| 0000 ;      if JP8 pin 2-3 connected
| 0000
| 0000
| 0000 boot:
| 0000
| 0000 init:
| 0000
| 0000 ; Set up the MMU
| 0000 ;
| 0000 ; Common Bank 0
| 0000 ;   logical: 0x0000 - 0xefff
| 0000 ;   physical: 0x00000 - 0x0efff
| 0000 ; Bank Area, not used in this test (probably?)
| 0000 ; Common Bank 1
| 0000 ;   logical: 0xf000 - 0xfffff
| 0000 ;   physical: 0x8f000 - 0x8fffff
| 0000
| 0000 3e 80      ld a, 0x80
| 0002 01 38 00   ld bc, CBR
| 0005 ed 79      out (c), a
| 0007 3e 00      ld a, 0x00
| 0009 01 39 00   ld bc, BBR
| 000c ed 79      out (c), a
| 000e 3e ff      ld a, 0xff
| 0010 01 3a 00   ld bc, CBAR
| 0013 ed 79      out (c), a
| 0015
| 0015 31 00 00   ; Set up Stack Pointer (first push/call will wrap to 0xfffff)
| 0015           ld sp, 0x0000
| 0018
| 0018 ; Initialize devices
| 0018 ; and blink LED
| 0018
| 0018 06 01      ld b, 1      ;1 blink, MMU initialized
| 001a cd 6f 00   call ledblink
| 001d 06 03      ld b, 3
| 001f cd 8c 00   call delays
| 0022
| 0022 cd 46 00   call asci0init
| 0025 06 02      ld b, 2      ;2 blinks, ASCII 0 initialized
| 0027 cd 6f 00   call ledblink
| 002a 06 03      ld b, 3
| 002c cd 8c 00   call delays
| 002f
| 002f 06 03      ld b, 3
| 0031 cd 6f 00   call ledblink
| 0034 06 03      ld b, 3
| 0036 cd 8c 00   call delays
| 0039
| 0039 06 04      ld b, 4
| 003b cd 6f 00   call ledblink
| 003e 06 03      ld b, 3
| 0040 cd 8c 00   call delays
| 0043
| 0043 c3 a1 00   jp testloop
| 0046
| 0046 ; ASCII routines
| 0046
| 0046 ; Initialize port 0
| 0046 asci0init:
| 0046 3e 64      ld a, 01100100b
| 0048 ; bit 7 = 0: MPE - disabled
| 0048 ; bit 6 = 1: RE - Rx enabled
| 0048 ; bit 5 = 1: TE - Tx enabled
| 0048 ; bit 4 = 0: RTS0 - set to low, RTS active (?)
| 0048 ; bit 3 = 0: MPBR/EFR - not used
| 0048 ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
| 0048 01 00 00   ld bc, CNTLA0
| 004b ed 79      out (c), a
| 004d
| 004d ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
| 004d ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
| 004d 3e 21      ld a, 00100001b
| 004f ; bit 7 = 0: MPBT - disabled
| 004f ; bit 6 = 0: MP - disabled
| 004f ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
| 004f ; bit 4 = 0: PEO - ignored as no parity configured
| 004f ; bit 3 = 0: DR - Clock factor = 16
| 004f ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
```

```
004f 01 02 00          ld bc, CNTLB0
0052 ed 79          out (c), a
0054
0054 c9          ret
0055
0055 ; Output a character on port 0
0055 ; reg E contains character to output
0055 asci0putc:
0055 01 04 00          ld bc, STAT0
0058 ed 78          in a, (c)
005a e6 02          and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
005c 28 f7          jr z, asci0putc ;not empty yet
005e 7b          ld a, e
005f 01 06 00          ld bc, TDR0
0062 ed 79          out (c), a      ;output character
0064 c9          ret
0065
0065 ; Output a character string on port 0
0065 ; reg HL points to string to output
0065 ; the string is ended by 0
0065 asci0pstr:
0065 7e          ld a, (hl)
0066 b7          or a
0067 c8          ret z
0068 5e          ld e, (hl)
0069 23          inc hl
006a cd 55 00          call asci0putc
006d 18 f6          jr asci0pstr
006f
006f ; ledblink flashes the LED
006f ; number of times in B reg
006f ; using reg: a, b, hl
006f ledblink:
006f 21 ff ff          ld hl, 0xffff
0072 3e 01          ld a, 1
0074 d3 43          out(LEDON), a
0076
0076 2b          dec hl
0077 7c          ld a, h
0078 b5          or l
0079 c2 76 00          jp nz, ledonloop
007c 21 ff ff          ld hl, 0xffff
007f 3e 00          ld a, 0
0081 d3 42          out(LEDOFF), a
0083
0083 2b          dec hl
0084 7c          ld a, h
0085 b5          or l
0086 c2 83 00          jp nz, ledoffloop
0089 10 e4          djnz ledblink
008b c9          ret
008c
008c ; delays makes a delay multiplied
008c ; by number of times in B reg
008c ; using reg: a, b, hl
008c delays:
008c 21 ff ff          ld hl, 0xffff
008f
008f 2b          dec hl
0090 7c          ld a, h
0091 b5          or l
0092 c2 8f 00          jp nz, delaylloop
0095 21 ff ff          ld hl, 0xffff
0098
0098 2b          dec hl
0099 7c          ld a, h
009a b5          or l
009b c2 98 00          jp nz, delay2loop
009e 10 ec          djnz delays
00a0 c9          ret
00a1
00a1 ; Main test loop
00a1 testloop:
00a1 06 08          ld b, 8
00a3 cd 6f 00          call ledblink
00a6 06 03          ld b, 3
00a8 cd 8c 00          call delays
00ab
00ab 21 b4 00          ld hl, built
00ae cd 65 00          call asci0pstr
00b1
00b1 c3 a1 00          jp testloop
00b4
00b4 built:
00b4 ..          db "Test program for Z180 computer "
00d3           include "tbuilt180.z80"
00d3 ..          db ", Built 2021-08-17 16:46"
# End of file tbuilt180.z80
00eb .. 0a          db '\r', '\n'
00ed 00          db 0
00ee
# End of file test180.z80
00ee
```

For some mysterious reason it was not possible to have a jmp init after the label boot: and put the built: text before init.

TODO: More experimenting with MMU setup is needed.

## Test program with serial output and input on ports 0 & 1

2021-08-18

Saved as test180\_5.z80

```
; Test program for the Z180 computer
; test180.z80
;
; testing:
;   - MMU setup
;   - RAM as stack
;   - Serial output port 0 & 1
;   - Serial input port 0 & 1
; todo
;   - RAM test
;   - copy test program to RAM and execute
;   - test all RAM using MEMSEL
;   - interrupt test

; Internal ports

; ASCII Registers port 0 and 1
CNTLA0: equ 0x0000 ;ASCII Channel Control Register A 0
CNTLA1: equ 0x0001 ;ASCII Channel Control Register A 1
CNTLB0: equ 0x0002 ;ASCII Control Register B 0
CNTLB1: equ 0x0003 ;ASCII Control Register B 1
STAT0: equ 0x0004 ;ASCII Status Register 0
STAT1: equ 0x0005 ;ASCII Status Register 1
TDR0: equ 0x0006 ;ASCII Transmit Data Register 0
TDR1: equ 0x0007 ;ASCII Transmit Data Register 1
RDR0: equ 0x0008 ;ASCII Receive Data FIFO 0
RDR1: equ 0x0009 ;ASCII Receive Data FIFO 1

; MMU Registers
CBR: equ 0x0038 ;MMU Common Base Register
BBR: equ 0x0039 ;MMU Bank Base Register
CBAR: equ 0x003a ;MMU Common/Bank Area Register

; External ports

;Select EPROM or RAM on address 0x0000 - 0x3fffff
ROMSEL: equ 0x40 ;Write selects EPROM (reset condition)
RAMSEL: equ 0x41 ;Write selects RAM

;LED on/off
LEDOFF: equ 0x42 ;Write turns LED off (reset condition)
LEDON: equ 0x43 ;Write turns LED on

;SPI device select and AVR reset
CSPORT: equ 0x44 ;Write to bit 0 - 3 (reset condition, all 0)
;Bit 0: select SD_CS0 when set to 1
;Bit 1: select SD_CS1 when set to 1
;Bit 2: select ATSS (AVR) when set to 1
;Bit 1: reset AVR when set to 1
;           if JP8 pin 2-3 connected

boot:
init:

; Set up the MMU
;
; Common Bank 0
;   logical: 0x0000 - 0xffff
;   physical: 0x00000 - 0x0ffff, EPROM (or low RAM if enabled)
; Bank Area, not used in this test (probably?)
; Common Bank 1
;   logical: 0xf000 - 0xffff
;   physical: 0x8f000 - 0x8ffff, high RAM
;
; the inner workings of the MMU is a bit mysterious
; but this configuration works. TODO investigate further

    ld a, 0x80
    ld bc, CBR
    out (c), a
    ld a, 0x00
    ld bc, BBR
    out (c), a
    ld a, 0xff
```

```

ld bc, CBAR
out (c), a

; Set up Stack Pointer (first push/call will wrap to 0xffff)
ld sp, 0x0000

; Initialize devices
; and blink LED

ld b, 1      ;1 blink, MMU initialized
call ledblink
ld b, 3
call delays

call asci0init
ld b, 2      ;2 blinks, ASCII 0 initialized
call ledblink
ld b, 3
call delays

call asci1init
ld b, 3      ;3 blinks, ASCII 1 initialized
call ledblink
ld b, 3
call delays

jp testloop

; ASCII routines

; Initialize port 0
asci0init:
ld a, 01100100b
    ; bit 7 = 0: MPE - disabled
    ; bit 6 = 1: RE - Rx enabled
    ; bit 5 = 1: TE - Tx enabled
    ; bit 4 = 0: RTS0 - set to low, RTS active (?)
    ; bit 3 = 0: MPBR/EFR - not used
    ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
ld bc, CNTLA0
out (c), a

; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
ld a, 00100001b
    ; bit 7 = 0: MPBT - disabled
    ; bit 6 = 0: MP - disabled
    ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
    ; bit 4 = 0: PEO - ignored as no parity configured
    ; bit 3 = 0: DR - Clock factor = 16
    ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
ld bc, CNTLB0
out (c), a

ret

; Output a character on port 0
; reg E contains character to output
asci0putc:
ld bc, STAT0
in a, (c)
and 00000010b  ;test bit 1 = TDRE: Transmit Data Register Empty
jr z, asci0putc ;not empty yet
ld a, e
ld bc, TDR0
out (c), a      ;output character
ret

; Input a character from port 0
; reg E contains the character
; if E == 0 no character is available
asci0getc:
ld e, 0
ld bc, STAT0
in a, (c)
and 10000000b  ;test bit 7 = RDRF: Recieve data in FIFO
ret z           ;empty
ld a, e
ld bc, RDR0
in a, (c)       ;input character
ld e, a
ret

; Output a character string on port 0
; reg HL points to string to output
; the string is ended by 0
asci0pstr:
ld a, (hl)
or a
ret z
ld e, (hl)
inc hl
call asci0putc

```

```

jr asci0pstr

; Initialize port 1
ascilinit:
    ld a, 01100100b
        ; bit 7 = 0: MPE - disabled
        ; bit 6 = 1: RE - Rx enabled
        ; bit 5 = 1: TE - Tx enabled
        ; bit 4 = 0: RTS0 - set to low, RTS active (?)
        ; bit 3 = 0: MPBR/EFR - not used
        ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
    ld bc, CNTLA1
    out (c), a

; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
    ld a, 00100001b
        ; bit 7 = 0: MPBT - disabled
        ; bit 6 = 0: MP - disabled
        ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
        ; bit 4 = 0: PEO - ignored as no parity configured
        ; bit 3 = 0: DR - Clock factor = 16
        ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
    ld bc, CNTLB1
    out (c), a

    ret

; Output a character on port 1
; reg E contains character to output
ascilputc:
    ld bc, STAT1
    in a, (c)
    and 00000010b    ;test bit 1 = TDRE: Transmit Data Register Empty
    jr z, ascilputc  ;not empty yet
    ld a, e
    ld bc, TDR1
    out (c), a       ;output character
    ret

; Input a character from port 1
; reg E contains the character
; if E == 0 no character is available
ascilgetc:
    ld e, 0
    ld bc, STAT1
    in a, (c)
    and 10000000b    ;test bit 7 = RDRF: Recieve data in FIFO
    ret z            ;empty
    ld a, e
    ld bc, RDR1
    in a, (c)        ;input character
    ld e, a
    ret

; Output a character string on port 1
; reg HL points to string to output
; the string is ended by 0
ascilpstr:
    ld a, (hl)
    or a
    ret z
    ld e, (hl)
    inc hl
    call ascilputc
    jr ascilpstr

; ledblink flashes the LED
; number of times in B reg
; using reg: a, b, hl
ledblink:
    ld hl, 0xffff
    ld a, 1
    out(LEDON), a
ledonloop:
    dec hl
    ld a, h
    or l
    jp nz, ledonloop
    ld hl, 0xffff
    ld a, 0
    out(LEDOFF), a
ledoffloop:
    dec hl
    ld a, h
    or l
    jp nz, ledoffloop
    djnz ledblink
    ret

; delays makes a delay multiplied
; by number of times in B reg
; using reg: a, b, hl

```

```

delays:
    ld hl, 0xffff
delay1loop:
    dec hl
    ld a, h
    or l
    jp nz, delay1loop
    ld hl, 0xffff
delay2loop:
    dec hl
    ld a, h
    or l
    jp nz, delay2loop
    djnz delays
    ret

; Main test loop
testloop:
    ld b, 5
    call ledblink
    ld b, 2
    call delays

    ld hl, asci0txt
    call asci0pstr
    ld hl, built
    call asci0pstr
    call asci0getc
    ld a, e
    or a
    jp z, asci0noin
    call asci0putc
    ld hl, inptxt
    call asci0pstr
asci0noin:
    ld hl, asci0txt
    call asci0pstr
    ld hl, built
    call asci0pstr
    call asci0getc
    ld a, e
    or a
    jp z, asci0noin
    call asci0putc
    ld hl, inptxt
    call asci0pstr
asci0noin:
    jp testloop

asci0txt:
    db "ASCII port 0 - "
    db 0
asci0txt:
    db "ASCII port 1 - "
    db 0

built:
    db "Test program for Z180 computer"
    include "tbuilt180.z80"
    db '\r', '\n'
    db 0

inptxt:
    db "<- was received"
    db '\r', '\n'
    db 0

```

## Test program with MMU setup and serial output and input on ports 0 & 1

2021-08-19

Saved as test180\_6.z80

MMU setup

```

...
0000          ; Set up the MMU
0000          ;
0000          ; Common Bank 0
0000          ;   logical: 0x0000 - 0x1fff
0000          ;   physical: 0x00000 - 0x01ffff, EPROM (or low RAM if enabled)
0000          ; Bank Area
0000          ;   logical: 0x2000 - 0xefff
0000          ;   physical: 0x42000 - 0x4effff, low RAM above EPROM

```

```

0000 ; Common Bank 1
0000 ;   logical: 0xf000 - 0xfffff
0000 ;   physical: 0x8f000 - 0x8ffff, high RAM
0000 ;
0000 ; the inner workings of the MMU is a bit mysterious
0000 ;
0000
0000 3e f2      ld a, 0xf2    ;<CA><BA>
0002 01 3a 00    ld bc, CBAR
0005 ed 79      out (c), a
0007 3e 80      ld a, 0x80
0009 01 38 00    ld bc, CBR
000c ed 79      out (c), a
000e 3e 40      ld a, 0x40
0010 01 39 00    ld bc, BBR
0013 ed 79      out (c), a
...

```

Note the order in which the MMU registers are configured. If CBAR was configured last the addressing is wrong after configuring CBR.

## Test program with MMU setup, RAM access and serial output and input on ports 0 & 1

2021-08-19

Saved as test180\_7.z80

```

; Test program for the Z180 computer
; test180.z80
;
; testing:
;   - simple MMU setup
;   - RAM as stack
;   - Serial output port 0 & 1
;   - Serial input port 0 & 1
;   - MMU setup with Common Bank 0, Bank Area, Common Bank 1
; todo
;   - RAM test
;   - copy test program to RAM and execute
;   - test all RAM using MEMSEL
;   - interrupt test
;
; Internal ports
;
; ASCII Registers port 0 and 1
CNTLA0: equ 0x0000 ;ASCII Channel Control Register A 0
CNTLA1: equ 0x0001 ;ASCII Channel Control Register A 1
CNTLB0: equ 0x0002 ;ASCII Control Register B 0
CNTLB1: equ 0x0003 ;ASCII Control Register B 1
STAT0: equ 0x0004 ;ASCII Status Register 0
STAT1: equ 0x0005 ;ASCII Status Register 1
TDR0: equ 0x0006 ;ASCII Transmit Data Register 0
TDR1: equ 0x0007 ;ASCII Transmit Data Register 1
RDR0: equ 0x0008 ;ASCII Receive Data FIFO 0
RDR1: equ 0x0009 ;ASCII Receive Data FIFO 1
;
; MMU Registers
CBR:   equ 0x0038 ;MMU Common Base Register
BBR:   equ 0x0039 ;MMU Bank Base Register
CBAR:  equ 0x003a ;MMU Common/Bank Area Register
;
; External ports
;
;Select EPROM or RAM on address 0x0000 - 0x3ffff
ROMSEL: equ 0x40    ;Write selects EPROM (reset condition)
RAMSEL: equ 0x41    ;Write selects RAM
;
;LED on/off
LEDOFF: equ 0x42    ;Write turns LED off (reset condition)
LEDON:  equ 0x43    ;Write turns LED on
;
;SPI device select and AVR reset
CSPORT: equ 0x44    ;Write to bit 0 - 3 (reset condition, all 0)
;Bit 0: select SD_CS0 when set to 1
;Bit 1: select SD_CS1 when set to 1
;Bit 2: select ATSS (AVR) when set to 1
;Bit 3: reset AVR when set to 1
;           if JP8 pin 2-3 connected
;
boot:
init:

```

```

; Set up the MMU
;
; Common Bank 0
;   logical: 0x0000 - 0x1fff
;   physical: 0x00000 - 0x01ffff, EPROM (or low RAM if enabled)
; Bank Area
;   logical: 0x2000 - 0xefff
;   physical: 0x42000 - 0x4efff, low RAM above EPROM
; Common Bank 1
;   logical: 0xf000 - 0xffff
;   physical: 0x8f000 - 0x8ffff, high RAM
;
; the inner workings of the MMU is a bit mysterious
; but CBAR must be configured before CBR and BBR

    ld a, 0xf2    ;<CA><BA>
    ld bc, CBAR
    out (c), a
    ld a, 0x80
    ld bc, CBR
    out (c), a
    ld a, 0x40
    ld bc, BBR
    out (c), a

; Set up Stack Pointer (first push/call will wrap to 0xffff)
    ld sp, 0x0000

; Initialize devices
; and blink LED

    ld b, 1      ;1 blink, MMU initialized
    call ledblink
    ld b, 3
    call delays

    call asci0init
    ld b, 2      ;2 blinks, ASCII 0 initialized
    call ledblink
    ld b, 3
    call delays

    call ascilinit
    ld b, 3      ;3 blinks, ASCII 1 initialized
    call ledblink
    ld b, 3
    call delays

    ld a, 0
    ld (0x2021), a

    jp testloop

; ASCII routines

; Initialize port 0
asci0init:
    ld a, 01100100b
        ; bit 7 = 0: MPE - disabled
        ; bit 6 = 1: RE - Rx enabled
        ; bit 5 = 1: TE - Tx enabled
        ; bit 4 = 0: RTS0 - set to low, RTS active (?)
        ; bit 3 = 0: MPBR/EFR - not used
        ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
    ld bc, CNTLA0
    out (c), a

    ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
    ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
    ld a, 00100001b
        ; bit 7 = 0: MPBT - disabled
        ; bit 6 = 0: MP - disabled
        ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
        ; bit 4 = 0: PEO - ignored as no parity configured
        ; bit 3 = 0: DR - Clock factor = 16
        ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
    ld bc, CNTLB0
    out (c), a

    ret

; Output a character on port 0
; reg E contains character to output
asci0putc:
    ld bc, STAT0
    in a, (c)
    and 00000010b    ;test bit 1 = TDRE: Transmit Data Register Empty
    jr z, asci0putc ;not empty yet
    ld a, e
    ld bc, TDR0
    out (c), a       ;output character
    ret

```

```
; Input a character from port 0
; reg E contains the character
; if E == 0 no character is available
asci0getc:
    ld e, 0
    ld bc, STAT0
    in a, (c)
    and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
    ret z           ;empty
    ld a, e
    ld bc, RDR0
    in a, (c)       ;input character
    ld e, a
    ret

; Output a character string on port 0
; reg HL points to string to output
; the string is ended by 0
asci0pstr:
    ld a, (hl)
    or a
    ret z
    ld e, (hl)
    inc hl
    call asci0putc
    jr asci0pstr

; Initialize port 1
ascilinit:
    ld a, 01100100b
        ; bit 7 = 0: MPE - disabled
        ; bit 6 = 1: RE - Rx enabled
        ; bit 5 = 1: TE - Tx enabled
        ; bit 4 = 0: RTS0 - set to low, RTS active (?)
        ; bit 3 = 0: MPBR/EFR - not used
        ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
    ld bc, CNTLA1
    out (c), a

    ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
    ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
    ld a, 00100001b
        ; bit 7 = 0: MPBT - disabled
        ; bit 6 = 0: MP - disabled
        ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
        ; bit 4 = 0: PEO - ignored as no parity configured
        ; bit 3 = 0: DR - Clock factor = 16
        ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
    ld bc, CNTLB1
    out (c), a

    ret

; Output a character on port 1
; reg E contains character to output
ascilputc:
    ld bc, STAT1
    in a, (c)
    and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
    jr z, ascilputc ;not empty yet
    ld a, e
    ld bc, TDR1
    out (c), a      ;output character
    ret

; Input a character from port 1
; reg E contains the character
; if E == 0 no character is available
ascilgetc:
    ld e, 0
    ld bc, STAT1
    in a, (c)
    and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
    ret z           ;empty
    ld a, e
    ld bc, RDR1
    in a, (c)       ;input character
    ld e, a
    ret

; Output a character string on port 1
; reg HL points to string to output
; the string is ended by 0
ascilpstr:
    ld a, (hl)
    or a
    ret z
    ld e, (hl)
    inc hl
    call ascilputc
    jr ascilpstr
```

```
; ledblink flashes the LED
; number of times in B reg
; using reg: a, b, hl
ledblink:
    ld hl, 0xffff
    ld a, 1
    out(LEDON), a
ledonloop:
    dec hl
    ld a, h
    or l
    jp nz, ledonloop
    ld hl, 0xffff
    ld a, 0
    out(LEDOFF), a
ledoffloop:
    dec hl
    ld a, h
    or l
    jp nz, ledoffloop
    djnz ledblink
    ret

; delays makes a delay multiplied
; by number of times in B reg
; using reg: a, b, hl
delays:
    ld hl, 0xffff
delay1loop:
    dec hl
    ld a, h
    or l
    jp nz, delay1loop
    ld hl, 0xffff
delay2loop:
    dec hl
    ld a, h
    or l
    jp nz, delay2loop
    djnz delays
    ret

; Main test loop
testloop:
    ld b, 5
    call ledblink
    ld b, 2
    call delays

    ld hl, asci0txt
    call asci0pstr
    ld hl, indicator
    ld b, 0
    ld a, (0x2021)
    ld c, a
    add hl, bc
    ld e, (hl)
    call asci0putc
    ld hl, built
    call asci0pstr
    call asci0getc
    ld a, e
    or a
    jp z, asci0noin
    call asci0putc
    ld hl, inptxt
    call asci0pstr
asci0noin:
    ld hl, asciltxt
    call ascilpstr
    ld hl, indicator
    ld b, 0
    ld a, (0x2021)
    ld c, a
    add hl, bc
    ld e, (hl)
    call ascilputc
    ld hl, built
    call ascilpstr
    call ascilgetc
    ld a, e
    or a
    jp z, ascilnoin
    call ascilputc
    ld hl, inptxt
    call ascilpstr
ascilnoin:
;Indicator index for messages

    ld a, (0x2021)
    inc a
```

```

        and 00000011b
        ld (0x2021), a

        jp testloop

ascii0txt:
        db "ASCII port 0 "
        db 0
asci1txt:
        db "ASCII port 1 "
        db 0

indicator:
        db '|', '/', '-', '\'

built:
        db " Test program for Z180 computer"
        include "tbuilt180.z80"
        db '\r', '\n'
        db 0

inptxt:
        db "<- was received"
        db '\r', '\n'
        db 0

```

## Test program of MMU setup

---

2021-08-20

Only the MMU setup is somewhat changed in this setup.

Saved as test180\_8.z80

```

...
; Set up the MMU
;
; Common Bank 0, 4KB
;   logical: 0x0000 - 0xffff
;   physical: 0x00000 - 0x00ffff, EPROM or start of low RAM if enabled
; Bank Area, 56KB
;   logical: 0x1000 - 0xe000
;   physical: 0x41000 - 0x4efff, low RAM chip above EPROM
; Common Bank 1, 4KB
;   logical: 0xf000 - 0xffff
;   physical: 0xff000 - 0xfffff, end of high RAM chip
;
; The MMU function is a bit mysterious but I learned that CBAR
; must be configured before CBR and BBR otherwise strange
; things will happen.

ld a, 0xf1    ;<CA><BA>
ld bc, CBAR
out (c), a
ld a, 0xf0
ld bc, CBR
out (c), a
ld a, 0x40
ld bc, BBR
out (c), a
...

```

## Test program copied to RAM and running from there

---

2021-08-21

Saved as test180\_11.z80

```

# File test180.z80
0000          ; Test program for the Z180 computer
0000          ; test180.z80
0000
0000          ; testing:
0000          ;   - simple MMU setup
0000          ;   - RAM as stack
0000          ;   - Serial output port 0 & 1
0000          ;   - Serial input port 0 & 1
0000          ;   - MMU setup with Common Bank 0, Bank Area, Common Bank 1

```

```

0000      ; - simple RAM test
0000      ; - copy test program to RAM and execute
0000      ; - switch to low RAM using MEMSEL
0000      ; - test 74LS74 select outputs
0000      ; todo
0000      ; - test all RAM using MEMSEL
0000      ; - interrupt test
0000
0000      ; Internal ports
0000
0000      ; ASCII Registers port 0 and 1
0000      CNTLA0: equ 0x0000 ;ASCII Channel Control Register A 0
0000      CNTLA1: equ 0x0001 ;ASCII Channel Control Register A 1
0000      CNTLB0: equ 0x0002 ;ASCII Control Register B 0
0000      CNTLB1: equ 0x0003 ;ASCII Control Register B 1
0000      STAT0: equ 0x0004 ;ASCII Status Register 0
0000      STAT1: equ 0x0005 ;ASCII Status Register 1
0000      TDR0:  equ 0x0006 ;ASCII Transmit Data Register 0
0000      TDR1:  equ 0x0007 ;ASCII Transmit Data Register 1
0000      RDR0:  equ 0x0008 ;ASCII Receive Data FIFO 0
0000      RDR1:  equ 0x0009 ;ASCII Receive Data FIFO 1
0000
0000      ; MMU Registers
0000      CBR:   equ 0x0038 ;MMU Common Base Register
0000      BBR:   equ 0x0039 ;MMU Bank Base Register
0000      CBAR:  equ 0x003a ;MMU Common/Bank Area Register
0000
0000      ; External ports
0000
0000      ;Select EPROM or RAM on address 0x0000 - 0x3ffff
0000      ROMSEL: equ 0x40    ;Write selects EPROM (reset condition)
0000      RAMSEL: equ 0x41    ;Write selects RAM
0000
0000      ;LED on/off
0000      LEDOFF: equ 0x42   ;Write turns LED off (reset condition)
0000      LEDON:  equ 0x43   ;Write turns LED on
0000
0000      ;SPI device select and AVR reset
0000      CSPORT: equ 0x44   ;Write to bit 0 - 3 (reset condition, all 0)
0000          ;Bit 0: select SD_CS0 when set to 1
0000          ;Bit 1: select SD_CS1 when set to 1
0000          ;Bit 2: select ATSS (AVR) when set to 1
0000          ;Bit 1: reset AVR when set to 1
0000          ;           if JP8 pin 2-3 connected
0000
0000
0000      boot:
0000
0000      init:
0000
0000      ; Set up the MMU
0000      ;
0000      ; Common Bank 0, 4KB
0000      ;     logical: 0x0000 - 0x0fff
0000      ;     physical: 0x000000 - 0x000fff, EPROM or start of low RAM if enabled
0000      ; Bank Area, 56KB
0000      ;     logical: 0x1000 - 0xffff
0000      ;     physical: 0x41000 - 0x4efff, low RAM chip above EPROM
0000      ; Common Bank 1, 4KB
0000      ;     logical: 0xf000 - 0xffff
0000      ;     physical: 0xff000 - 0xfffff, end of high RAM chip
0000
0000      ; The MMU function is a bit mysterious but I learned that CBAR
0000      ; must be configured before CBR and BBR otherwise strange
0000      ; things will happen.
0000
0000      3e f1      ;<CA><BA>
0002 01 3a 00      ld a, 0xf1
0005 ed 79      ld bc, CBAR
0007 3e f0      out (c), a
0009 01 38 00      ld a, 0xf0
000c ed 79      ld bc, CBR
000e 3e 40      out (c), a
0010 01 39 00      ld a, 0x40
0013 ed 79      ld bc, BBR
0015                  out (c), a
0015
0015      HIRAM: equ 0xf000
0015
0015      ; copy the program to high RAM
0015 01 c9 01      ld bc, prgend - prgstart
0018 21 2e 00      ld hl, prgineeprom
001b 11 00 f0      ld de, HIRAM
001e
001e 78      cloop:
001f b1      ld a,b
0020 ca 2b 00      or c
0023 7e      jp z,cpend
0024 23      ld a,(hl)
0025 12      inc hl
0026 13      ld (de),a
0027 0b      inc de
0028 c3 1e 00      dec bc
0029 00      jp cloop
002b      cpPEND:

```

```

002b c3 00 f0      jp HIRAM    ; jump to the copied code
002e
002e
002e
002e
f000
f000
f000      prgineprom:
f000      org HIRAM
f000      prgstart:
f000      ; Set up Stack Pointer (first push/call will wrap to 0xffff)
f000      ld sp, 0x0000
f003
f003      ; Initialize devices
f003      ; and blink LED
f003
f003      f003 06 01      ld b, 1      ;1 blink, MMU initialized
f005      cd aa f0      call ledblink
f008 06 02      ld b, 2      call delays
f00a      cd c7 f0
f00d
f00d      f00d cd 34 f0      call asci0init
f010 06 02      ld b, 2      ;2 blinks, ASCII 0 initialized
f012      cd aa f0      call ledblink
f015 06 02      ld b, 2      call delays
f017      cd c7 f0
f01a
f01a      f01a cd 6f f0      call ascilinit
f01d 06 03      ld b, 3      ;3 blinks, ASCII 1 initialized
f01f      cd aa f0      call ledblink
f022 06 02      ld b, 2      call delays
f024      cd c7 f0
f027
f027      f027 3e 00      ld a, 0
f029 32 c9 f1      ld (indindex), a
f02c
f02c      f02c 3e 11      ld a, 00010001b
f02e 32 ca f1      ld (cspattern), a
f031
f031      f031 c3 dc f0      jp testloop
f034
f034      f034 routines
f034
f034      f034 Initialize port 0
f034      f034 asci0init:
f034 3e 64      ld a, 01100100b
f036
f036      f036      ; bit 7 = 0: MPE - disabled
f036      f036      ; bit 6 = 1: RE - Rx enabled
f036
f036      f036      ; bit 5 = 1: TE - Tx enabled
f036
f036      f036      ; bit 4 = 0: RTS0 - set to low, RTS active (?)
f036
f036      f036      ; bit 3 = 0: MPBR/EFR - not used
f036
f036      f036      ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
f036 01 00 00
f039 ed 79      ld bc, CNTLA0
f03b
f03b      f03b      out (c), a
f03b
f03b      f03b      ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
f03b
f03b      f03b      ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
f03b 3e 21      ld a, 00100001b
f03d
f03d      f03d      ; bit 7 = 0: MPBT - disabled
f03d      f03d      ; bit 6 = 0: MP - disabled
f03d
f03d      f03d      ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
f03d
f03d      f03d      ; bit 4 = 0: PEO - ignored as no parity configured
f03d
f03d      f03d      ; bit 3 = 0: DR - Clock factor = 16
f03d
f03d      f03d      ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
f03d 01 02 00
f040 ed 79      ld bc, CNTLB0
f042
f042      f042 c9      out (c), a
f042
f042      f042      ret
f043
f043      f043      ; Output a character on port 0
f043      f043      ; reg E contains character to output
f043      f043 asci0putc:
f043 01 04 00
f046 ed 78      ld bc, STAT0
f048 e6 02      in a, (c)
f04a 28 f7      and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
f04c 7b
f04d 01 06 00
f050 ed 79      jr z, asci0putc ;not empty yet
f052 c9      ld a, e
f053
f053      f053      ld bc, TDR0
f053      f053      out (c), a ;output character
f053
f053      f053      ret
f053
f053      f053      ; Input a character from port 0
f053      f053      ; reg E contains the character
f053      f053      ; if E == 0 no character is available
f053      f053 asci0getc:
f053 1e 00
f055 01 04 00
f058 ed 78      ld e, 0
f05a e6 80      ld bc, STAT0
f05c c8      in a, (c)
f05d 7b      and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
f05e 01 08 00
f061 ed 78      ret z ;empty
f063 5f      ld a, e
f064 c9      ld bc, RDR0
f064 c9      in a, (c) ;input character
f064 c9      ld e, a
f064 c9      ret

```

```

f065          ; Output a character string on port 0
f065          ; reg HL points to string to output
f065          ; the string is ended by 0
f065          asci0pstr:
f065          ld a, (hl)
f066          or a
f067          ret z
f068          ld e, (hl)
f069          inc hl
f06a          call asci0putc
f06d          jr asci0pstr
f06f
f06f          ; Initialize port 1
f06f          asciilinit:
f06f          3e 64
f06f          ld a, 01100100b
f071          ; bit 7 = 0: MPE - disabled
f071          ; bit 6 = 1: RE - Rx enabled
f071          ; bit 5 = 1: TE - Tx enabled
f071          ; bit 4 = 0: RTS0 - set to low, RTS active (?)
f071          ; bit 3 = 0: MPBR/EFR - not used
f071          ; bit 2 = 0 = 100 - Start + 8 bit data + 1 stop
f071          ld bc, CNTLAI
f074          out (c), a
f076
f076          ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
f076          ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
f076          3e 21
f076          ld a, 00100001b
f078          ; bit 7 = 0: MPBT - disabled
f078          ; bit 6 = 0: MP - disabled
f078          ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
f078          ; bit 4 = 0: PEO - ignored as no parity configured
f078          ; bit 3 = 0: DR - Clock factor = 16
f078          ; bit 2 = 0 = 001: SS2-SS0 - Divide Ratio: 2
f078          ld bc, CNTLBI
f07b          out (c), a
f07d
f07d          c9
f07e          ret
f07e
f07e          ; Output a character on port 1
f07e          ; reg E contains character to output
f07e          asciiputc:
f07e          01 05 00
f081          ed 78
f083          e6 02
f085          28 f7
f087          7b
f088          01 07 00
f08b          ed 79
f08d          c9
f08e
f08e          ; Input a character from port 1
f08e          ; reg E contains the character
f08e          ; if E == 0 no character is available
f08e          asciigetc:
f08e          1e 00
f090          01 05 00
f093          ed 78
f095          e6 80
f097          c8
f098          7b
f099          01 09 00
f09c          ed 78
f09e          5f
f09f          c9
f0a0
f0a0          ; Output a character string on port 1
f0a0          ; reg HL points to string to output
f0a0          ; the string is ended by 0
f0a0          ascilpstr:
f0a0          7e
f0a1          b7
f0a2          c8
f0a3          5e
f0a4          23
f0a5          cd 7e f0
f0a8          18 f6
f0aa
f0aa          ; ledblink flashes the LED
f0aa          ; number of times in B reg
f0aa          ; using reg: a, b, hl
f0aa          ledblink:
f0aa          ld hl, 0xffff
f0ad          3e 01
f0af          d3 43
f0b1
f0b1          2b
f0b2          7c
f0b3          b5
f0b4          c2 b1 f0
f0b7          21 ff ff
f0ba          3e 00
f0bc          d3 42
out(LEDON), a
ledonloop:
dec hl
ld a, h
or l
jp nz, ledonloop
ld hl, 0xffff
ld a, 0
out(LEDOFF), a

```

```

f0be          ledoffloop:
f0be 2b        dec hl
f0bf 7c        ld a, h
f0c0 b5        or l
f0c1 c2 be f0  jp nz, ledoffloop
f0c4 10 e4    djnz ledblink
f0c6 c9        ret

f0c7          ; delays makes a delay multiplied
f0c7          ; by number of times in B reg
f0c7          ; using reg: a, b, hl
f0c7          delays:
f0c7 21 ff ff  ld hl, 0xffff
f0ca          delay1loop:
f0ca 2b        dec hl
f0cb 7c        ld a, h
f0cc b5        or l
f0cd c2 ca f0  jp nz, delay1loop
f0d0 21 ff ff  ld hl, 0xffff
f0d3          delay2loop:
f0d3 2b        dec hl
f0d4 7c        ld a, h
f0d5 b5        or l
f0d6 c2 d3 f0  jp nz, delay2loop
f0d9 10 ec    djnz delays
f0db c9        ret

f0dc          ; Main test loop
f0dc          testloop:
f0dc 06 01    ld b, 1
f0de cd aa f0  call ledblink
f0e1 06 01    ld b, 1
f0e3 cd c7 f0  call delays

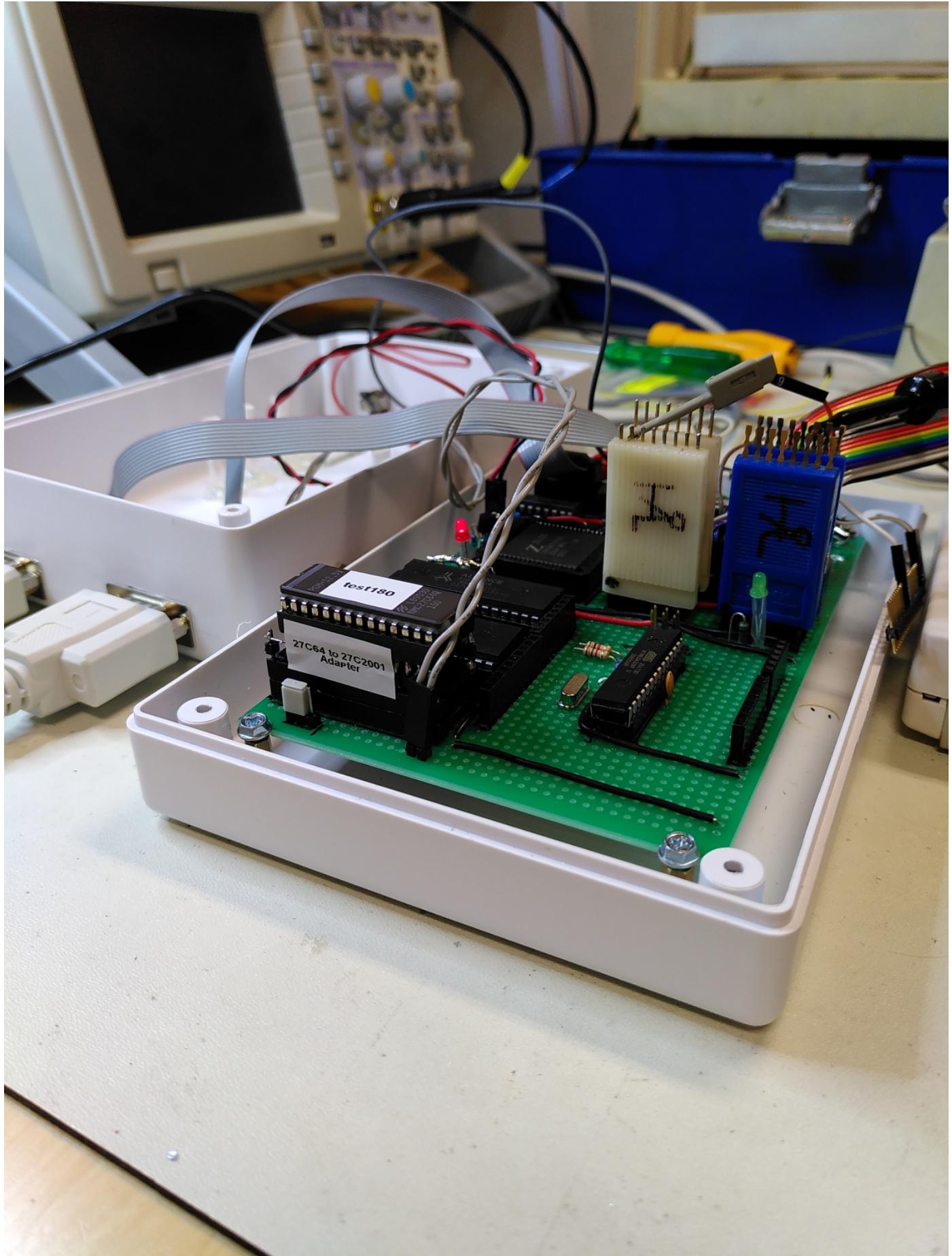
f0e6          ld a, 1
f0e8 d3 41    out (RAMSEL), a
f0ea 06 02    ld b, 2
f0ec cd c7 f0  call delays
f0ef 3e 01    ld a, 1
f0f1 d3 40    out (ROMSEL), a
f0f3          ld hl, asci0txt
f0f6 cd 65 f0  call asci0pstr
f0f9 21 78 f1  ld hl, indicator
f0fc 06 00    ld b, 0
f0fe 3a c9 f1  ld a, (indindex)
f101 4f        ld c, a
f102 09        add hl, bc
f103 5e        ld e, (hl)
f104 cd 43 f0  call asci0putc
f107 21 7c f1  ld hl, built
f10a cd 65 f0  call asci0pstr
f10d cd 53 f0  call asci0getc
f110 7b        ld a, e
f111 b7        or a
f112 ca 1e f1  jp z, asci0noin
f115 cd 43 f0  call asci0putc
f118 21 b6 f1  ld hl, inptxt
f11b cd 65 f0  call asci0pstr
f11e          asci0noin:
f11e 21 6b f1  ld hl, asci1txt
f121 cd a0 f0  call asci1pstr
f124 21 78 f1  ld hl, indicator
f127 06 00    ld b, 0
f129 3a c9 f1  ld a, (indindex)
f12c 4f        ld c, a
f12d 09        add hl, bc
f12e 5e        ld e, (hl)
f12f cd 7e f0  call asci1putc
f132 21 7c f1  ld hl, built
f135 cd a0 f0  call asci1pstr
f138 cd 8e f0  call asci1getc
f13b 7b        ld a, e
f13c b7        or a
f13d ca 49 f1  jp z, asci1noin
f140 cd 7e f0  call asci1putc
f143 21 b6 f1  ld hl, inptxt
f146 cd a0 f0  call asci1pstr
f149          asci1noin:
f149          ;Indicator index for messages
f149          ld a, (indindex)
f149          inc a
f14d e6 03    and 00000011b
f14f 32 c9 f1  ld (indindex), a

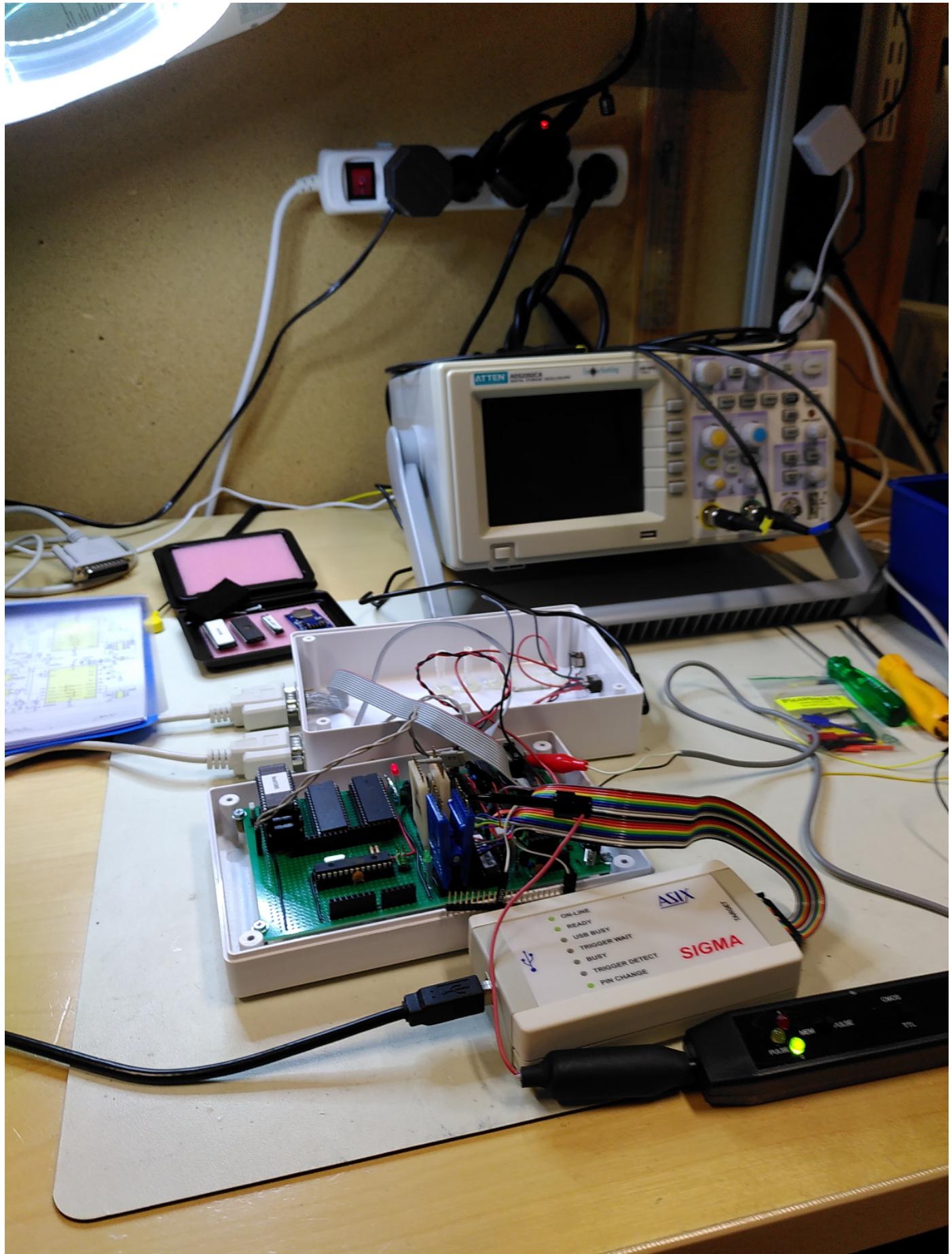
f152          ;Chip Select and reset output
f152 3a ca f1  ld a, (cspattern)
f155 d3 44    out (CSPORT), a
f157 07        rlca
f158 32 ca f1  ld (cspattern), a

```

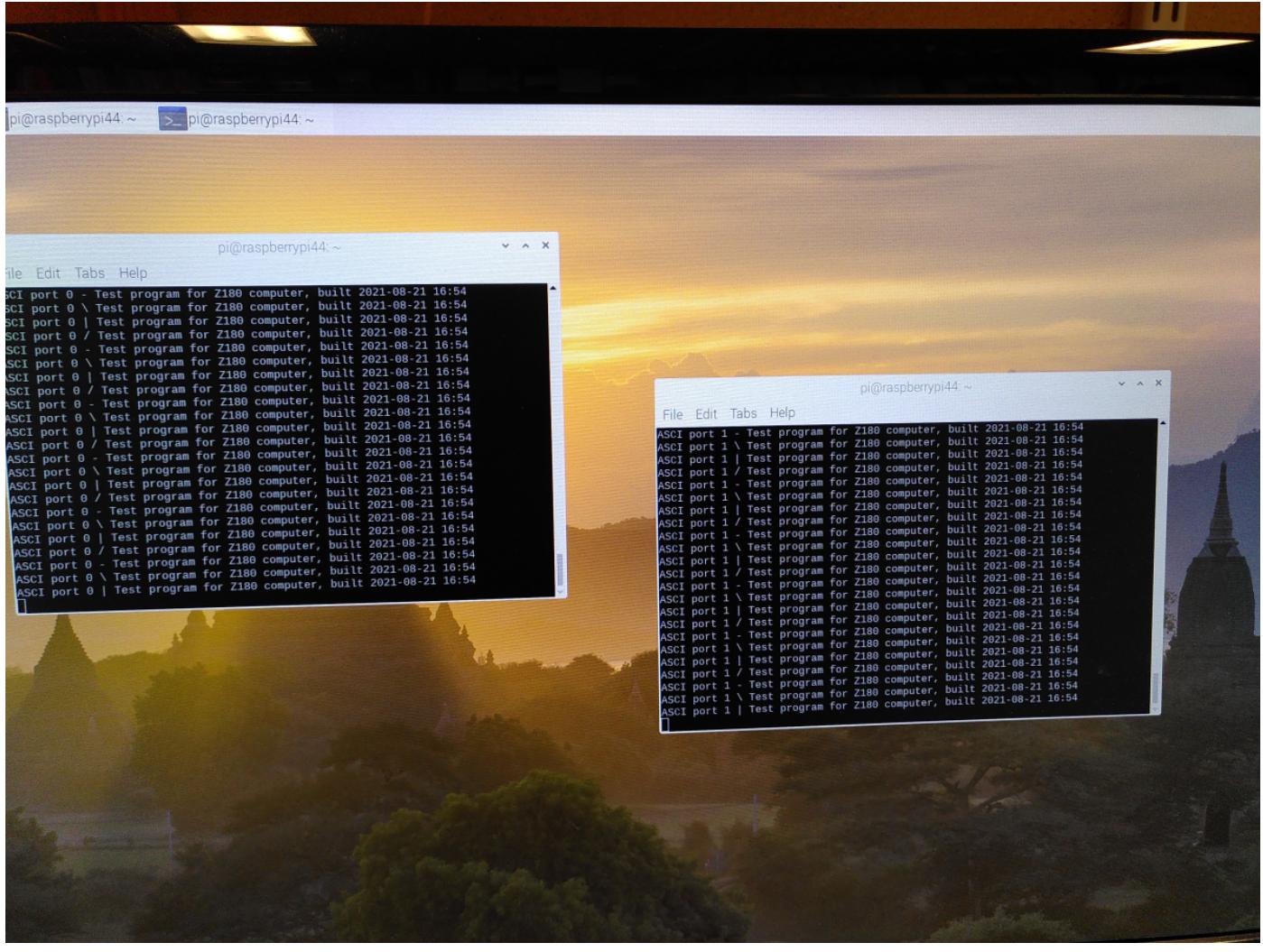
```
f15b          jp testloop
f15b c3 dc f0
f15e
f15e          asci0txt:
f15e ..        db "ASCII port 0 "
f16a 00        db 0
f16b          asc1txt:
f16b ..        db "ASCII port 1 "
f177 00        db 0
f178
f178          indicator:
f178 .. 2f 2d 5c      db '|', '/', '-', '\\'
f17c
f17c          built:
f17c ..        db " Test program for Z180 computer"
f19b          include "tbuilt180.z80"
f19b ..        db ", built 2021-08-21 16:54"
# End of file tbuilt180.z80
f1b3 .. 0a        db '\r', '\n'
f1b5 00        db 0
f1b6
f1b6          inptxt:
f1b6 ..        db "<- was received"
f1c6 .. 0a        db '\r', '\n'
f1c8 00        db 0
f1c9
f1c9          prgend:
f1c9
f1c9          indindex:
f1c9 00        db 0
f1ca
f1ca          cspattern:
f1ca 00        db 0
f1cb
# End of file test180.z80
f1cb
```

Lab setup when running this test program:





Test output:



When running in RAM the 3 wait states when accessing memory may be modified as the AS6C4008-55PCN 512KB x 8 SRAM has an access time of 55ns while the M27C2001-10F1 256KB x 8 EPROM has an access time of 100ns (M27C2001-15 has 150ns access time).

## Test of RAM

2021-08-22

Saved as test180\_12.z80

```
# File test180.z80
0000 ; Test program for the Z180 computer
0000 ; test180.z80
0000 ;
0000 ; testing:
0000 ; - simple MMU setup
0000 ; - RAM as stack
0000 ; - Serial output port 0 & 1
0000 ; - Serial input port 0 & 1
0000 ; - MMU setup with Common Bank 0, Bank Area, Common Bank 1
0000 ; - simple RAM test
0000 ; - copy test program to RAM and execute
0000 ; - switch to low RAM using MEMSEL
0000 ; - test 74LS74 select outputs
0000 ; - test all RAM using MMU and MEMSEL
0000 ; todo
0000 ; - interrupt test
0000 ;
0000 ; Internal ports
0000 ;
0000 ; ASCII Registers port 0 and 1
CNTLA0: equ 0x0000 ;ASCII Channel Control Register A 0
0000 CNTLA1: equ 0x0001 ;ASCII Channel Control Register A 1
0000 CNTLB0: equ 0x0002 ;ASCII Control Register B 0
0000 CNTLB1: equ 0x0003 ;ASCII Control Register B 1
0000 STAT0: equ 0x0004 ;ASCII Status Register 0
0000 STAT1: equ 0x0005 ;ASCII Status Register 1
0000 TDR0: equ 0x0006 ;ASCII Transmit Data Register 0
```

```

0000      TDR1:  equ 0x0007 ;ASCII Transmit Data Register 1
0000      RDR0:  equ 0x0008 ;ASCII Receive Data FIFO 0
0000      RDR1:  equ 0x0009 ;ASCII Receive Data FIFO 1
0000
0000      ; MMU Registers
0000      CBR:   equ 0x0038 ;MMU Common Base Register
0000      BBR:   equ 0x0039 ;MMU Bank Base Register
0000      CBAR:  equ 0x003a ;MMU Common/Bank Area Register
0000
0000      ; External ports
0000
0000      ;Select EPROM or RAM on address 0x0000 - 0x3fff
0000      ROMSEL: equ 0x40    ;Write selects EPROM (reset condition)
0000      RAMSEL: equ 0x41    ;Write selects RAM
0000
0000      ;LED on/off
0000      LEDOFF: equ 0x42   ;Write turns LED off (reset condition)
0000      LEDON:  equ 0x43   ;Write turns LED on
0000
0000      ;SPI device select and AVR reset
0000      CSPORT: equ 0x44   ;Write to bit 0 - 3 (reset condition, all 0)
0000          ;Bit 0: select SD_CS0 when set to 1
0000          ;Bit 1: select SD_CS1 when set to 1
0000          ;Bit 2: select ATSS (AVR) when set to 1
0000          ;Bit 1: reset AVR when set to 1
0000          ;           if JP8 pin 2-3 connected
0000
0000
0000      boot:
0000
0000      init:
0000
0000      ; Set up the MMU initially
0000
0000      ; Common Bank 0, 4KB
0000          logical: 0x0000 - 0x0fff
0000          physical: 0x00000 - 0x00fff, EPROM or start of low RAM if enabled
0000      ; Bank Area, 56KB
0000          logical: 0x1000 - 0xffff
0000          physical: 0x41000 - 0x4efff, low RAM chip above EPROM
0000      ; Common Bank 1, 4KB
0000          logical: 0xf000 - 0xffff
0000          physical: 0xff000 - 0xfffff, end of high RAM chip
0000
0000      ; The MMU function is a bit mysterious but I learned that CBAR
0000      ; must be configured before CBR and BBR otherwise strange
0000      ; things will happen.
0000
0000 3e f1      ld a, 0xf1    ;<CA><BA>
0002 01 3a 00    ld bc, CBAR
0005 ed 79      out (c), a
0007 3e f0      ld a, 0xf0
0009 01 38 00    ld bc, CBR
000c ed 79      out (c), a
000e 3e 40      ld a, 0x40
0010 01 39 00    ld bc, BBR
0013 ed 79      out (c), a
0015
0015      HIRAM: equ 0xf000
0015
0015      ; copy the program to high RAM
0015 01 5e 05    ld bc,prgend - prgstart
0018 21 2e 00    ld hl,prgineprom
001b 11 00 f0    ld de,HIRAM
001e
001e 78      cloop:
001f b1      ld a,b
0020 ca 2b 00    or c
0023 7e      jp z,cpend
0024 23      ld a,(hl)
0025 12      inc hl
0026 13      ld (de),a
0027 0b      inc de
0028 c3 1e 00    dec bc
002b      jp cloop
002b c3 00 f0    cpend:
002b      jp HIRAM    ; jump to the copied code
002e
002e      prgineprom:
002e
002e      org HIRAM
002e
002e      f000
0000      f000
0000      f000
0000 31 00 00    f003
0003
0003      f003
0003      f003
0003 06 01      f005 cd aa f0
0008 06 02      f008 cd c7 f0
0000      ; Initialize devices
0000      ; and blink LED
0000      ld b, 1      ;1 blink, MMU initialized
0000      call ledblink
0000      ld b, 2
0000      call delays

```

```

f00d
f00d cd 34 f0          call asci0init
f010 06 02              ld b, 2           ;2 blinks, ASCII 0 initialized
f012 cd aa f0          call ledblink
f015 06 02              ld b, 2           ;3 blinks, ASCII 1 initialized
f017 cd c7 f0          call delays
f01a
f01a cd 6f f0          call asci0init
f01d 06 03              ld b, 3           ;3 blinks, ASCII 1 initialized
f01f cd aa f0          call ledblink
f022 06 02              ld b, 2           ;3 blinks, ASCII 1 initialized
f024 cd c7 f0          call delays
f027
f027 3e 00              ld a, 0
f029 32 5e f5          ld (indindex), a
f02c
f02c 3e 11              ld a, 00010001b
f02e 32 5f f5          ld (cspattern), a
f031
f031 c3 05 f1          jp testloop
f034
f034 ; ASCII routines
f034
f034 ; Initialize port 0
f034 asci0init:
f034 3e 64              ld a, 01100100b
f036 ; bit 7 = 0: MPE - disabled
f036 ; bit 6 = 1: RE - Rx enabled
f036 ; bit 5 = 1: TE - Tx enabled
f036 ; bit 4 = 0: RTS0 - set to low, RTS active (?)
f036 ; bit 3 = 0: MPBR/EFR - not used
f036 ; bit 2 = 0 = 100 - Start + 8 bit data + 1 stop
f036 01 00 00
f039 ed 79              ld bc, CNTLАО
f03b
f03b out (c), a
f03b
f03b ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
f03b ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
f03b 3e 21              ld a, 00100001b
f03d ; bit 7 = 0: MPBT - disabled
f03d ; bit 6 = 0: MP - disabled
f03d ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
f03d ; bit 4 = 0: PEO - ignored as no parity configured
f03d ; bit 3 = 0: DR - Clock factor = 16
f03d ; bit 2 = 0 = 001: SS2-SS0 - Divide Ratio: 2
f03d 01 02 00
f040 ed 79              ld bc, CNTLB0
f042
f042 c9                ret
f043
f043 ; Output a character on port 0
f043 ; reg E contains character to output
f043 asci0putc:
f043 01 04 00
f046 ed 78              ld bc, STAT0
f048 e6 02              in a, (c)
f04a 28 f7              and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
f04c 7b
f04d 01 06 00
f050 ed 79              jr z, asci0putc ;not empty yet
f052 c9                ld a, e
f052
f053
f053 ; Input a character from port 0
f053 ; reg E contains the character
f053 ; if E == 0 no character is available
f053 asci0getc:
f053 1e 00
f055 01 04 00
f058 ed 78              ld bc, STAT0
f05a e6 80              in a, (c)
f05c c8                and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
f05d 7b
f05e 01 08 00
f061 ed 78              ret z           ;empty
f063 5f
f064 c9                ld a, e
f064
f065
f065 ; Output a character string on port 0
f065 ; reg HL points to string to output
f065 ; the string is ended by 0
f065 asci0pstr:
f065 7e
f066 b7                ld a, (hl)
f067 c8                or a
f068 5e
f069 23
f06a cd 43 f0          ret z
f06d 18 f6              ld e, (hl)
f06f
f06f ; Initialize port 1
f06f asci0init:
f06f 3e 64              inc hl
f06f 01100100b
f071 ; bit 7 = 0: MPE - disabled
f071 ; bit 6 = 1: RE - Rx enabled

```

```
f071 ; bit 5 = 1: TE - Tx enabled
f071 ; bit 4 = 0: RTS0 - set to low, RTS active (?)
f071 ; bit 3 = 0: MPBR/EFR - not used
f071 ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
f071 01 01 00 ld bc, CNTLAI
f074 ed 79 out (c), a
f076 ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
f076 ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
f076 3e 21 ld a, 00100001b
f078 ; bit 7 = 0: MPBT - disabled
f078 ; bit 6 = 0: MP - disabled
f078 ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
f078 ; bit 4 = 0: PEO - ignored as no parity configured
f078 ; bit 3 = 0: DR - Clock factor = 16
f078 ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
f078 01 03 00 ld bc, CNTLBI
f07b ed 79 out (c), a
f07d
f07d c9 ret
f07e
f07e ; Output a character on port 1
f07e ; reg E contains character to output
f07e asciilputc:
f07e 01 05 00 ld bc, STAT1
f081 ed 78 in a, (c)
f083 e6 02 and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
f085 28 f7 jr z, asciilputc ;not empty yet
f087 7b ld a, e
f088 01 07 00 ld bc, TDR1
f08b ed 79 out (c), a ;output character
f08d c9 ret
f08e
f08e ; Input a character from port 1
f08e ; reg E contains the character
f08e ; if E == 0 no character is available
f08e asciilgetc:
f08e 1e 00 ld e, 0
f090 01 05 00 ld bc, STAT1
f093 ed 78 in a, (c)
f095 e6 80 and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
f097 c8 ret z ;empty
f098 7b ld a, e
f099 01 09 00 ld bc, RDR1
f09c ed 78 in a, (c) ;input character
f09e 5f ld e, a
f09f c9 ret
f0a0
f0a0 ; Output a character string on port 1
f0a0 ; reg HL points to string to output
f0a0 ; the string is ended by 0
f0a0 asciipstr:
f0a0 7e ld a, (hl)
f0a1 b7 or a
f0a2 c8 ret z
f0a3 5e ld e, (hl)
f0a4 23 inc hl
f0a5 cd 7e f0 call asciilputc
f0a8 18 f6 jr asciipstr
f0aa
f0aa ; ledblink flashes the LED
f0aa ; number of times in B reg
f0aa ; using reg: a, b, hl
f0aa ledblink:
f0aa 21 ff ff ld hl, 0xffff
f0ad 3e 01 ld a, 1
f0af d3 43 out(LEDON), a
f0b1
f0b1 2b dec hl
f0b2 7c ld a, h
f0b3 b5 or l
f0b4 c2 b1 f0 jp nz, ledonloop
f0b7 21 ff ff ld hl, 0xffff
f0ba 3e 00 ld a, 0
f0bc d3 42 out(LEDOFF), a
f0be
f0be 2b dec hl
f0bf 7c ld a, h
f0c0 b5 or l
f0c1 c2 be f0 jp nz, ledoffloop
f0c4 10 e4 djnz ledblink
f0c6 c9 ret
f0c7
f0c7 ; delays makes a delay multiplied
f0c7 ; by number of times in B reg
f0c7 ; using reg: a, b, hl
f0c7 delays:
f0c7 21 ff ff ld hl, 0xffff
f0ca delayloop:
f0ca 2b dec hl
f0cb 7c ld a, h
f0cc b5 or l
f0cd c2 ca f0 jp nz, delayloop
```

```
f0d0 21 ff ff          ld hl, 0xffff
f0d3      delay2loop:
f0d3 2b          dec hl
f0d4 7c          ld a, h
f0d5 b5          or l
f0d6 c2 d3 f0      jp nz, delay2loop
f0d9 10 ec          djnz delays
f0db c9          ret
f0dc
f0dc          ; Test RAM
f0dc          ; reg bc: number of bytes to test
f0dc          ; reg hl: start address of test
f0dc          ; reg a: returns 0 if RAM ok, 1 if error
f0dc          ramtest:
f0dc 3e 00          ld a,0           ; reset error flag
f0de 32 60 f5          ld (ramerr),a
f0e1          ramtstop:
f0e1 1e 00          ld e,0x00
f0e3 73          ld (hl),e
f0e4 7e          ld a,(hl)
f0e5 bb          cp e
f0e6 ca ee f0      jp z,ramtstff
f0e9 3e 01          ld a,1
f0eb 32 60 f5          ld (ramerr),a
f0ee          ramtstff:
f0ee 1e ff          ld e,0xff
f0f0 73          ld (hl),e
f0f1 7e          ld a,(hl)
f0f2 bb          cp e
f0f3 ca fb f0      jp z,ramtstnxt
f0f6 3e 01          ld a,1
f0f8 32 60 f5          ld (ramerr),a
f0fb          ramtstnxt:
f0fb 23          inc hl
f0fc 0b          dec bc
f0fd 78          ld a,b
f0fe b1          or c
f0ff 20 e0      jr nz, ramtstop
f101 3a 60 f5          ld a, (ramerr)
f104 c9          ret
f105
f105          ; Main test loop
f105          testloop:
f105 06 01          ld b, 1
f107 cd aa f0          call ledblink
f10a 06 01          ld b, 1
f10c cd c7 f0          call delays
f10f
f10f 21 34 f2          ld hl, asci0txt
f112 cd 65 f0          call asci0pstr
f115 21 4e f2          ld hl, indicator
f118 06 00          ld b, 0
f11a 3a 5e f5          ld a, (indindex)
f11d 4f          ld c, a
f11e 09          add hl, bc
f11f 5e          ld e, (hl)
f120 cd 43 f0          call asci0putc
f123 21 52 f2          ld hl, built
f126 cd 65 f0          call asci0pstr
f129 cd 53 f0          call asci0getc
f12c 7b          ld a, e
f12d b7          or a
f12e ca 3a f1          jp z, asci0noin
f131 cd 43 f0          call asci0putc
f134 21 8c f2          ld hl, inptxt
f137 cd 65 f0          call asci0pstr
f13a
f13a          asci0noin:
f13a 21 41 f2          ld hl, asci1txt
f13d cd a0 f0          call asci1pstr
f140 21 4e f2          ld hl, indicator
f143 06 00          ld b, 0
f145 3a 5e f5          ld a, (indindex)
f148 4f          ld c, a
f149 09          add hl, bc
f14a 5e          ld e, (hl)
f14b cd 7e f0          call asci1putc
f14e 21 52 f2          ld hl, built
f151 cd a0 f0          call asci1pstr
f154 cd 8e f0          call asci1getc
f157 7b          ld a, e
f158 b7          or a
f159 ca 65 f1          jp z, asci1noin
f15c cd 7e f0          call asci1putc
f15f 21 8c f2          ld hl, inptxt
f162 cd a0 f0          call asci1pstr
f165
f165          asci1noin:
f165          ;Indicator index for messages
f165 3a 5e f5          ld a, (indindex)
f168 3c          inc a
f169 e6 03          and 00000011b
f16b 32 5e f5          ld (indindex), a
```

```
f16e
f16e          ;Chip Select and reset outputs
f16e 3a 5f f5    ld a, (cspattern)
f171 d3 44    out (CSPORT), a
f173 07    rlca
f174 32 5f f5    ld (cspattern), a
f177
f177          ; Test RAM
f177
f177          ; Enable RAM in low memory
f177 3e 01    ld a, 1
f179 d3 41    out (RAMSEL), a ; select RAM instead of EPROM from address 0
f17b
f17b          ; Test physical RAM 0x00000 - 0x0efff
f17b 3e 00    ld a, 0x00
f17d 01 39 00    ld bc, BBR
f180 ed 79    out (c), a
f182
f182 21 b2 f2    ld hl, ramtxt0
f185 cd 65 f0    call asci0pstr
f188 21 b2 f2    ld hl, ramtxt0
f18b cd a0 f0    call asci0pstr
f18e 21 00 00    ld hl, 0x0000
f191 01 00 f0    ld bc, 0xf000
f194 cd dc f0    call ramtest
f197 b7    or a
f198 28 0e    jr z, ramt0ok
f19a 21 a7 f2    ld hl, ramtxtterr
f19d cd 65 f0    call asci0pstr
f1a0 21 a7 f2    ld hl, ramtxtterr
f1a3 cd a0 f0    call asci0pstr
f1a6 18 0c    jr ramt0end
f1a8
f1a8 21 9f f2    ramt0ok:
f1ab cd 65 f0    ld hl, ramtxtok
f1ae 21 9f f2    call asci0pstr
f1b1 cd a0 f0    ld hl, ramtxtok
f1b4
f1b4          ; Test physical RAM 0x0f000 - 0x1cff
f1b4 3e 0e    ld a, 0x0e
f1b6 01 39 00    ld bc, BBR
f1b9 ed 79    out (c), a
f1bb
f1bb 21 d6 f2    ld hl, ramtxt1
f1be cd 65 f0    call asci0pstr
f1c1 21 d6 f2    ld hl, ramtxt1
f1c4 cd a0 f0    call asci0pstr
f1c7 21 00 10    ld hl, 0x1000
f1ca 01 00 e0    ld bc, 0xe000
f1cd cd dc f0    call ramtest
f1d0 b7    or a
f1d1 28 0e    jr z, ramt1ok
f1d3 21 a7 f2    ld hl, ramtxtterr
f1d6 cd 65 f0    call asci0pstr
f1d9 21 a7 f2    ld hl, ramtxtterr
f1dc cd a0 f0    call asci0pstr
f1df 18 0c    jr ramt1end
f1e1
f1e1 21 9f f2    ramt1ok:
f1e4 cd 65 f0    ld hl, ramtxtok
f1e7 21 9f f2    call asci0pstr
f1ea cd a0 f0    ld hl, ramtxtok
f1ed
f1ed          ; Test physical RAM 0x7a000 - 0x87fff
f1ed 3e 79    ld a, 0x79
f1ef 01 39 00    ld bc, BBR
f1f2 ed 79    out (c), a
f1f4
f1f4 21 f6 f3    ld hl, ramtxt9
f1f7 cd 65 f0    call asci0pstr
f1fa 21 f6 f3    ld hl, ramtxt9
f1fd cd a0 f0    call asci0pstr
f200 21 00 10    ld hl, 0x1000
f203 01 00 e0    ld bc, 0xe000
f206 cd dc f0    call ramtest
f209 b7    or a
f20a 28 0e    jr z, ramt9ok
f20c 21 a7 f2    ld hl, ramtxtterr
f20f cd 65 f0    call asci0pstr
f212 21 a7 f2    ld hl, ramtxtterr
f215 cd a0 f0    call asci0pstr
f218 18 0c    jr ramt9end
f21a
f21a 21 9f f2    ramt9ok:
f21d cd 65 f0    ld hl, ramtxtok
f220 21 9f f2    call asci0pstr
f223 cd a0 f0    ld hl, ramtxtok
f226
f226          ; Enable EPROM in low memory
f226 3e 00    ld a, 0
```

```
f228 d3 40          out (ROMSEL), a
f22a
f22a      ; Restore Bank Area
f22a 3e 40          ld a, 0x40
f22c 01 39 00        ld bc, BBR
f22f ed 79          out (c), a
f231
f231 c3 05 f1        jp testloop
f234
f234      asci0txt:
f234 ..          db "ASCI port 0 "
f240 00          db 0
f241
f241      asc1txt:
f241 ..          db "ASCI port 1 "
f24d 00          db 0
f24e
f24e      indicator:
f24e .. 2f 2d 5c    db '|', '/', '-', '\\'
f252
f252      built:
f252 ..          db " Test program for Z180 computer"
f271           include "tbuilt180.z80"
f271 ..          db ", built 2021-08-22 16:31"
# End of file tbuilt180.z80
f289 .. 0a          db '\r', '\n'
f28b 00          db 0
f28c
f28c      inptxt:
f28c ..          db "<- was received"
f29c .. 0a          db '\r', '\n'
f29e 00          db 0
f29f
f29f      ramtxtok:
f29f ..          db "- OK"
f2a4 .. 0a          db '\r', '\n'
f2a6 00          db 0
f2a7
f2a7      ramtxterr:
f2a7 ..          db "- ERROR"
f2af .. 0a          db '\r', '\n'
f2b1 00          db 0
f2b2
f2b2      ramtxt0:
f2b2 ..          db "Test physical RAM 0x00000 - 0x0efff"
f2d5 00          db 0
f2d6
f2d6      ramtxt1:
f2d6 ..          db "Test physical RAM 0x0f000 - 0x1cff"
f2f9 00          db 0
f2fa
f2fa      ramtxt2:
f2fa ..          db "Test physical RAM 0x1d000 - 0x25fff"
f31d 00          db 0
f31e
f31e      ramtxt3:
f31e ..          db "Test physical RAM 0x26000 - 0x33fff"
f341 00          db 0
f342
f342      ramtxt4:
f342 ..          db "Test physical RAM 0x34000 - 0x41fff"
f365 00          db 0
f366
f366      ramtxt5:
f366 ..          db "Test physical RAM 0x42000 - 0x4ffff"
f389 00          db 0
f38a
f38a      ramtxt6:
f38a ..          db "Test physical RAM 0x50000 - 0x5dfff"
f3ad 00          db 0
f3ae
f3ae      ramtxt7:
f3ae ..          db "Test physical RAM 0x5e000 - 0x6bfff"
f3d1 00          db 0
f3d2
f3d2      ramtxt8:
f3d2 ..          db "Test physical RAM 0x6c000 - 0x79fff"
f3f5 00          db 0
f3f6
f3f6      ramtxt9:
f3f6 ..          db "Test physical RAM 0x7a000 - 0x87fff"
f419 00          db 0
f41a
f41a      ramtxt10:
f41a ..          db "Test physical RAM 0x88000 - 0x95fff"
f43d 00          db 0
f43e
f43e      ramtxt11:
f43e ..          db "Test physical RAM 0x96000 - 0xa3fff"
f461 00          db 0
f462
f462      ramtxt12:
f462 ..          db "Test physical RAM 0xa4000 - 0xb1fff"
f485 00          db 0
```

```

f486
f486          ramtxt13:
f486 ..        db "Test physical RAM 0xb2000 - 0xbffff"
f4a9 00        db 0
f4aa
f4aa          ramtxt14:
f4aa ..        db "Test physical RAM 0xc0000 - 0xcdffff"
f4cd 00        db 0
f4ce
f4ce          ramtxt15:
f4ce ..        db "Test physical RAM 0xce000 - 0xdbffff"
f4f1 00        db 0
f4f2
f4f2          ramtxt16:
f4f2 ..        db "Test physical RAM 0xdc000 - 0xe9ffff"
f515 00        db 0
f516
f516          ramtxt17:
f516 ..        db "Test physical RAM 0xea000 - 0xf7ffff"
f539 00        db 0
f53a
f53a          ramtxt18:
f53a ..        db "Test physical RAM 0xf8000 - 0xfeffff"
f55d 00        db 0
f55e
f55e          prgend:
f55e
f55e          indindex:
f55e 00        db 0
f55f
f55f          cspattern:
f55f 00        db 0
f560
f560          ramerr:
f560 00        db 0
f561
# End of file test180.z80
f561

```

## Test of RAM with no wait states

2021-08-23

The test program runs in 25s from reset to second printout of the line saying when it was built. With the default 3 waite states it runs in 45s.

Saved as test180\_12.z80

```

# File test180.z80
0000          ; Test program for the Z180 computer
0000          ; test180.z80
0000
0000          ;
0000          ; testing:
0000          ; - simple MMU setup
0000          ; - RAM as stack
0000          ; - Serial output port 0 & 1
0000          ; - Serial input port 0 & 1
0000          ; - MMU setup with Common Bank 0, Bank Area, Common Bank 1
0000          ; - simple RAM test
0000          ; - copy test program to RAM and execute
0000          ; - switch to low RAM using MEMSEL
0000          ; - test 74LS74 select outputs
0000          ; - test all RAM using MMU and MEMSEL
0000          ; - set no wait states for RAM access
0000          ; todo
0000          ; - interrupt test
0000
0000          ; Internal ports
0000
0000          ; ASCII Registers port 0 and 1
CNTLA0: equ 0x0000 ;ASCII Channel Control Register A 0
CNTLA1: equ 0x0001 ;ASCII Channel Control Register A 1
CNTLB0: equ 0x0002 ;ASCII Control Register B 0
CNTLB1: equ 0x0003 ;ASCII Control Register B 1
STAT0:  equ 0x0004 ;ASCII Status Register 0
STAT1:  equ 0x0005 ;ASCII Status Register 1
TDR0:   equ 0x0006 ;ASCII Transmit Data Register 0
TDR1:   equ 0x0007 ;ASCII Transmit Data Register 1
RDR0:   equ 0x0008 ;ASCII Receive Data FIFO 0
RDR1:   equ 0x0009 ;ASCII Receive Data FIFO 1
0000
0000          ; DMA/WAIT Control Register
DCNTL:  equ 0x0032 ;DMA/WAIT Control Register
0000
0000          ; MMU Registers

```

```

0000      CBR:    equ 0x0038 ;MMU Common Base Register
0000      BBR:    equ 0x0039 ;MMU Bank Base Register
0000      CBAR:   equ 0x003a ;MMU Common/Bank Area Register
0000
0000      ; External ports
0000
0000      ;Select EPROM or RAM on address 0x0000 - 0x3fffff
0000      ROMSEL: equ 0x40    ;Write selects EPROM (reset condition)
0000      RAMSEL: equ 0x41    ;Write selects RAM
0000
0000      ;LED on/off
0000      LEDOFF: equ 0x42   ;Write turns LED off (reset condition)
0000      LEDON:  equ 0x43   ;Write turns LED on
0000
0000      ;SPI device select and AVR reset
0000      CSPORT: equ 0x44   ;Write to bit 0 - 3 (reset condition, all 0)
0000          ;Bit 0: select SD_CS0 when set to 1
0000          ;Bit 1: select SD_CS1 when set to 1
0000          ;Bit 2: select ATSS (AVR) when set to 1
0000          ;Bit 1: reset AVR when set to 1
0000          ;           if JP8 pin 2-3 connected
0000
0000
0000      boot:
0000
0000      init:
0000
0000      ; Set up the MMU initially
0000
0000      ; Common Bank 0, 4KB
0000          logical: 0x0000 - 0x0fff
0000          physical: 0x00000 - 0x00ffff, EPROM or start of low RAM if enabled
0000      ; Bank Area, 56KB
0000          logical: 0x1000 - 0xffff
0000          physical: 0x41000 - 0x4efff, low RAM chip above EPROM
0000      ; Common Bank 1, 4KB
0000          logical: 0xf000 - 0xffff
0000          physical: 0xff000 - 0xfffff, end of high RAM chip
0000
0000      ; The MMU function is a bit mysterious but I learned that CBAR
0000      ; must be configured before CBR and BBR otherwise strange
0000      ; things will happen.
0000
0000      3e f1
0002 01 3a 00
0005 ed 79
0007 3e f0
0009 01 38 00
000c ed 79
000e 3e 40
0010 01 39 00
0013 ed 79
0015
0015
0015
0015 01 98 05
0018 21 2e 00
001b 11 00 f0
001e
001e 78
001f b1
0020 ca 2b 00
0023 7e
0024 23
0025 12
0026 13
0027 0b
0028 c3 1e 00
002b
002b c3 00 f0
002e
002e
002e
002e
f000
f000
f000
f000 31 00 00
f003
f003
f003 01 32 00
f006 ed 78
f008 e6 3f
f00a ed 79
f00c
f00c
f00c
f00c 06 01
f00e cd b3 f0
f011 06 02
0000      ld a, 0xf1    ;<CA><BA>
0000      ld bc, CBAR
0000      out (c), a
0000      ld a, 0xf0
0000      ld bc, CBR
0000      out (c), a
0000      ld a, 0x40
0000      ld bc, BBR
0000      out (c), a
0000
0000      HIRAM: equ 0xf000
0000
0000      ; copy the program to high RAM
0000          ld bc,prgend - prgstart
0000          ld hl,prgineprom
0000          ld de,HIRAM
0000
0000      cloop:
0000          ld a,b
0000          or c
0000          jp z,cpend
0000          ld a,(hl)
0000          inc hl
0000          ld (de),a
0000          inc de
0000          dec bc
0000          jp cloop
0000
0000      cpend:
0000          jp HIRAM    ; jump to the copied code
0000
0000      prgineprom:
0000
0000          org HIRAM
0000
0000      prgstart:
0000
0000      ; Set up Stack Pointer (first push/call will wrap to 0xfffff)
0000          ld sp, 0x0000
0000
0000      ; When running from RAM configure no memory wait states
0000          ld bc, DCNTL
0000          in a, (c)
0000          and 00111111b ; reset MVI0 and MVI1
0000          out (c), a
0000
0000      ; Initialize devices
0000      ; and blink LED
0000
0000          ld b, 1        ;1 blink, MMU initialized
0000          call ledblink
0000          ld b, 2

```

```

f013 cd d0 f0          call delays
f016
f016 cd 3d f0          call asci0init
f019 06 02              ld b, 2      ;2 blinks, ASCII 0 initialized
f01b cd b3 f0          call ledblink
f01e 06 02              ld b, 2
f020 cd d0 f0          call delays
f023
f023 cd 78 f0          call ascilinit
f026 06 03              ld b, 3      ;3 blinks, ASCII 1 initialized
f028 cd b3 f0          call ledblink
f02b 06 02              ld b, 2
f02d cd d0 f0          call delays
f030
f030 3e 00              ld a, 0
f032 32 98 f5          ld (indindex), a
f035
f035 3e 11              ld a, 00010001b
f037 32 99 f5          ld (cspattern), a
f03a
f03a c3 0e f1          jp testloop
f03d
f03d
f03d ; ASCII routines
f03d
f03d ; Initialize port 0
f03d asci0init:
f03d 3e 64              ld a, 01100100b
f03f                  ; bit 7 = 0: MPE - disabled
f03f                  ; bit 6 = 1: RE - Rx enabled
f03f                  ; bit 5 = 1: TE - Tx enabled
f03f                  ; bit 4 = 0: RTS0 - set to low, RTS active (?)
f03f                  ; bit 3 = 0: MPBR/EFR - not used
f03f                  ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
f03f 01 00 00              ld bc, CNTLA0
f042 ed 79              out (c), a
f044
f044 ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
f044 ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
f044 3e 21              ld a, 00100001b
f046                  ; bit 7 = 0: MPBT - disabled
f046                  ; bit 6 = 0: MP - disabled
f046                  ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
f046                  ; bit 4 = 0: PEO - ignored as no parity configured
f046                  ; bit 3 = 0: DR - Clock factor = 16
f046                  ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
f046 01 02 00              ld bc, CNTLB0
f049 ed 79              out (c), a
f04b
f04b c9
f04c
f04c ; Output a character on port 0
f04c ; reg E contains character to output
f04c asci0putc:
f04c 01 04 00              ld bc, STAT0
f04f ed 78              in a, (c)
f051 e6 02              and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
f053 28 f7              jr z, asci0putc ;not empty yet
f055 7b
f056 01 06 00              ld a, e
f059 ed 79              ld bc, TDR0
f05b c9                  out (c), a ;output character
f05c
f05c ; Input a character from port 0
f05c ; reg E contains the character
f05c ; if E == 0 no character is available
f05c asci0getc:
f05c 1e 00              ld e, 0
f05e 01 04 00              ld bc, STAT0
f061 ed 78              in a, (c)
f063 e6 80              and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
f065 c8
f066 7b
f067 01 08 00              ret z ;empty
f06a ed 78              ld a, e
f06a , (c) ;input character
f06c 5f
f06d c9                  ld e, a
f06e
f06e ; Output a character string on port 0
f06e ; reg HL points to string to output
f06e ; the string is ended by 0
f06e asci0pstr:
f06e 7e                  ld a, (hl)
f06f b7                  or a
f070 c8                  ret z
f071 5e
f072 23
f073 cd 4c f0              ld e, (hl)
f073 , (c) ;call asci0putc
f076 18 f6                  inc hl
f078
f078 ; Initialize port 1
f078 asci0init:
f078 3e 64              ld a, 01100100b
f07a ; bit 7 = 0: MPE - disabled

```

```

f07a ; bit 6 = 1: RE - Rx enabled
f07a ; bit 5 = 1: TE - Tx enabled
f07a ; bit 4 = 0: RTS0 - set to low, RTS active (?)
f07a ; bit 3 = 0: MPBR/EFR - not used
f07a ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
f07a 01 01 00 ld bc, CNTLAI1
f07d ed 79 out (c), a
f07f ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
f07f ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
f07f 3e 21 ld a, 00100001b
f081 ; bit 7 = 0: MPBT - disabled
f081 ; bit 6 = 0: MP - disabled
f081 ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
f081 ; bit 4 = 0: PEO - ignored as no parity configured
f081 ; bit 3 = 0: DR - Clock factor = 16
f081 ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
f081 01 03 00 ld bc, CNTLB1
f084 ed 79 out (c), a
f086
f086 c9 ret
f087
f087 ; Output a character on port 1
f087 ; reg E contains character to output
f087 asciputc:
f087 01 05 00 ld bc, STAT1
f08a ed 78 in a, (c)
f08c e6 02 and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
f08e 28 f7 jr z, asciputc ;not empty yet
f090 7b ld a, e
f091 01 07 00 ld bc, TDR1
f094 ed 79 out (c), a ;output character
f096 c9 ret
f097
f097 ; Input a character from port 1
f097 ; reg E contains the character
f097 ; if E == 0 no character is available
f097 ascigetc:
f097 1e 00 ld e, 0
f099 01 05 00 ld bc, STAT1
f09c ed 78 in a, (c)
f09e e6 80 and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
f0a0 c8 ret z ;empty
f0a1 7b ld a, e
f0a2 01 09 00 ld bc, RDR1
f0a5 ed 78 in a, (c) ;input character
f0a7 5f ld e, a
f0a8 c9 ret
f0a9
f0a9 ; Output a character string on port 1
f0a9 ; reg HL points to string to output
f0a9 ; the string is ended by 0
f0a9 asciplstr:
f0a9 7e ld a, (hl)
f0aa b7 or a
f0ab c8 ret z
f0ac 5e ld e, (hl)
f0ad 23 inc hl
f0ae cd 87 f0 call asciputc
f0b1 18 f6 jr asciplstr
f0b3
f0b3 ; ledblink flashes the LED
f0b3 ; number of times in B reg
f0b3 ; using reg: a, b, hl
f0b3 ledblink:
f0b3 21 ff ff ld hl, 0xffff
f0b6 3e 01 ld a, 1
f0b8 d3 43 out(LEDON), a
f0ba ledonloop:
f0ba 2b dec hl
f0bb 7c ld a, h
f0bc b5 or l
f0bd c2 ba f0 jp nz, ledonloop
f0c0 21 ff ff ld hl, 0xffff
f0c3 3e 00 ld a, 0
f0c5 d3 42 out(LEDOFF), a
f0c7 ledoffloop:
f0c7 2b dec hl
f0c8 7c ld a, h
f0c9 b5 or l
f0ca c2 c7 f0 jp nz, ledoffloop
f0cd 10 e4 djnz ledblink
f0cf c9 ret
f0d0
f0d0 ; delays makes a delay multiplied
f0d0 ; by number of times in B reg
f0d0 ; using reg: a, b, hl
f0d0 delays:
f0d0 21 ff ff ld hl, 0xffff
f0d3 delayloop:
f0d3 2b dec hl
f0d4 7c ld a, h
f0d5 b5 or l

```

```
| f0d6 c2 d3 f0          jp nz, delayloop
| f0d9 21 ff ff          ld hl, 0xffff
| f0dc                  delay2loop:
| f0dc 2b                dec hl
| f0dd 7c                ld a, h
| f0de b5                or l
| f0df c2 dc f0          jp nz, delay2loop
| f0e2 10 ec              djnz delays
| f0e4 c9                ret
| f0e5
| f0e5                  ; Test RAM
| f0e5                  ; reg bc: number of bytes to test
| f0e5                  ; reg hl: start address of test
| f0e5                  ; reg a; returns 0 if RAM ok, 1 if error
| f0e5 ramtest:
| f0e5 3e 00              ld a,0           ; reset error flag
| f0e7 32 9a f5          ld (ramerr),a
| f0ea
| f0ea 1e 00              ld e,0x00
| f0ec 73                ld (hl),e
| f0ed 7e                ld a,(hl)
| f0ee bb                cp e
| f0ef ca f7 f0          jp z,ramtstff
| f0f2 3e 01              ld a,1
| f0f4 32 9a f5          ld (ramerr),a
| f0f7
| f0f7 1e ff              ld e,0xff
| f0f9 73                ld (hl),e
| f0fa 7e                ld a,(hl)
| f0fb bb                cp e
| f0fc ca 04 f1          jp z,ramtstnxt
| f0ff 3e 01              ld a,1
| f101 32 9a f5          ld (ramerr),a
| f104
| f104 23                ramtstnxt:
| f105 0b                inc hl
| f106 78                dec bc
| f107 b1                ld a,b
| f108 20 e0              or c
| f10a 3a 9a f5          jr nz, ramtstlop
| f10d c9                ld a, (ramerr)
| f10e
| f10e                  ret
| f10e
| f10e                  ; Main test loop
| f10e 06 01              testloop:
| f110 cd b3 f0          ld b, 1
| f113 06 01              call ledblink
| f115 cd d0 f0          ld b, 1
| f118                  call delays
| f118 21 e7 f1          ld hl, asci0txt
| f11b cd 6e f0          call asci0pstr
| f11e 21 01 f2          ld hl, indicator
| f121 06 00              ld b, 0
| f123 3a 98 f5          ld a, (indindex)
| f126 4f                ld c, a
| f127 09                add hl, bc
| f128 5e                ld e, (hl)
| f129 cd 4c f0          call asci0putc
| f12c 21 05 f2          ld hl, built
| f12f cd 6e f0          call asci0pstr
| f132 cd 5c f0          call asci0getc
| f135 7b                ld a, e
| f136 b7                or a
| f137 ca 43 f1          jp z, asci0noin
| f13a cd 4c f0          call asci0putc
| f13d 21 3f f2          ld hl, inptxt
| f140 cd 6e f0          call asci0pstr
| f143
| f143                  asci0noin:
| f143 21 f4 f1          ld hl, asci0txt
| f146 cd a9 f0          call asci0pstr
| f149 21 01 f2          ld hl, indicator
| f14c 06 00              ld b, 0
| f14e 3a 98 f5          ld a, (indindex)
| f151 4f                ld c, a
| f152 09                add hl, bc
| f153 5e                ld e, (hl)
| f154 cd 87 f0          call asci0putc
| f157 21 05 f2          ld hl, built
| f15a cd a9 f0          call asci0pstr
| f15d cd 97 f0          call asci0getc
| f160 7b                ld a, e
| f161 b7                or a
| f162 ca 6e f1          jp z, asci0noin
| f165 cd 87 f0          call asci0putc
| f168 21 3f f2          ld hl, inptxt
| f16b cd a9 f0          call asci0pstr
| f16e
| f16e                  asci0noin:
| f16e 3a 98 f5          ;Indicator index for messages
| f16e                  ld a, (indindex)
| f171 3c                inc a
| f172 e6 03              and 00000011b
```

```
f174 32 98 f5           ld (indindex), a
f177
f177
f177 3a 99 f5           ;Chip Select and reset outputs
f17a d3 44               ld a, (cspattern)
f17c 07                 out (CSPORT), a
f17d 32 99 f5           rlc a
f180
f180
f180
f180 3e 01               ld (cspattern), a
f182 d3 41               out (RAMSEL), a ; select RAM instead of EPROM from address 0
f184
f184 dd 21 52 f2
f188
f188 dd 6e 00
f18b dd 66 01
f18e 7c
f18f b5
f190 28 47
f192 dd 7e 02
f195 01 39 00
f198 ed 79
f19a
f19a cd 6e f0
f19d dd 6e 00
f1a0 dd 66 01
f1a3 cd a9 f0
f1a6 dd 6e 03
f1a9 dd 66 04
f1ac dd 4e 05
f1af dd 46 06
f1b2 cd e5 f0
f1b5 b7
f1b6 28 0e
f1b8 21 e1 f2
f1bb cd 6e f0
f1be 21 e1 f2
f1c1 cd a9 f0
f1c4 18 0c
f1c6
f1c6 21 d9 f2
f1c9 cd 6e f0
f1cc 21 d9 f2
f1cf cd a9 f0
f1d2
f1d2 01 07 00
f1d5 dd 09
f1d7 18 af
f1d9
f1d9
f1d9 3e 00
f1db d3 40
f1dd
f1dd
f1dd 3e 40
f1df 01 39 00
f1e2 ed 79
f1e4
f1e4 c3 0e f1
f1e7
f1e7 ..             asci0txt:
f1f3 00               db "ASCII port 0 "
f1f4 ..             asciltxt:
f1f4 ..             db "ASCII port 1 "
f200 00
f201
f201 .. 2f 2d 5c
f205
f205 ..             indicator:
f205 ..             db '|', '/', '-', '\\'
f205 ..             built:
f205 ..             db " Test program for Z180 computer"
f224
f224 ..             include "tbuilt180.z80"
f224 ..             db ", built 2021-08-23 11:21"
# End of file tbuilt180.z80
f23c .. 0a             db '\r', '\n'
f23e 00               db 0
f23f
f23f
f23f ..             inptxt:
f23f ..             db "<- was received"
f24f .. 0a             db '\r', '\n'
f251 00               db 0
f252
f252 ..             ramtsttab:
f252 ec f2             dw ramtxt0 ;test message, end of table if NUL
f254 00               db 0x00 ;BBR
f255 00 00             dw 0x0000 ;logical test start address
f257 00 f0             dw 0xf000 ;size of RAM block to test
```

```
f259          dw ramtxt1 ;test message
f259 10 f3    db 0x0e   ;BBR
f25b 0e       dw 0x1000 ;logical test start address
f25c 00 10    dw 0xe000 ;size of RAM block to test
f260
f260 34 f3    dw ramtxt2 ;test message
f262 1c       db 0x1c   ;BBR
f263 00 10    dw 0x1000 ;logical test start address
f265 00 e0    dw 0xe000 ;size of RAM block to test
f267
f267 58 f3    dw ramtxt3 ;test message
f269 2a       db 0x2a   ;BBR
f26a 00 10    dw 0x1000 ;logical test start address
f26c 00 e0    dw 0xe000 ;size of RAM block to test
f26e
f26e 7c f3    dw ramtxt4 ;test message
f270 38       db 0x38   ;BBR
f271 00 10    dw 0x1000 ;logical test start address
f273 00 e0    dw 0xe000 ;size of RAM block to test
f275
f275 a0 f3    dw ramtxt5 ;test message
f277 46       db 0x46   ;BBR
f278 00 10    dw 0x1000 ;logical test start address
f27a 00 e0    dw 0xe000 ;size of RAM block to test
f27c
f27c c4 f3    dw ramtxt6 ;test message
f27e 54       db 0x54   ;BBR
f27f 00 10    dw 0x1000 ;logical test start address
f281 00 e0    dw 0xe000 ;size of RAM block to test
f283
f283 e8 f3    dw ramtxt7 ;test message
f285 62       db 0x62   ;BBR
f286 00 10    dw 0x1000 ;logical test start address
f288 00 e0    dw 0xe000 ;size of RAM block to test
f28a
f28a 0c f4    dw ramtxt8 ;test message
f28c 70       db 0x70   ;BBR
f28d 00 10    dw 0x1000 ;logical test start address
f28f 00 e0    dw 0xe000 ;size of RAM block to test
f291
f291 30 f4    dw ramtxt9 ;test message
f293 7e       db 0x7e   ;BBR
f294 00 10    dw 0x1000 ;logical test start address
f296 00 e0    dw 0xe000 ;size of RAM block to test
f298
f298 54 f4    dw ramtxt10 ;test message
f29a 8c      db 0x8c   ;BBR
f29b 00 10    dw 0x1000 ;logical test start address
f29d 00 e0    dw 0xe000 ;size of RAM block to test
f29f
f29f 78 f4    dw ramtxt11 ;test message
f2a1 9a       db 0x9a   ;BBR
f2a2 00 10    dw 0x1000 ;logical test start address
f2a4 00 e0    dw 0xe000 ;size of RAM block to test
f2a6
f2a6 9c f4    dw ramtxt12 ;test message
f2a8 a8       db 0xa8   ;BBR
f2a9 00 10    dw 0x1000 ;logical test start address
f2ab 00 e0    dw 0xe000 ;size of RAM block to test
f2ad
f2ad c0 f4    dw ramtxt13 ;test message
f2af b6       db 0xb6   ;BBR
f2b0 00 10    dw 0x1000 ;logical test start address
f2b2 00 e0    dw 0xe000 ;size of RAM block to test
f2b4
f2b4 e4 f4    dw ramtxt14 ;test message
f2b6 c4       db 0xc4   ;BBR
f2b7 00 10    dw 0x1000 ;logical test start address
f2b9 00 e0    dw 0xe000 ;size of RAM block to test
f2bb
f2bb 08 f5    dw ramtxt15 ;test message
f2bd d2       db 0xd2   ;BBR
f2be 00 10    dw 0x1000 ;logical test start address
f2c0 00 e0    dw 0xe000 ;size of RAM block to test
f2c2
f2c2 2c f5    dw ramtxt16 ;test message
f2c4 e0       db 0xe0   ;BBR
f2c5 00 10    dw 0x1000 ;logical test start address
f2c7 00 e0    dw 0xe000 ;size of RAM block to test
f2c9
f2c9 50 f5    dw ramtxt17 ;test message
f2cb ee       db 0xee   ;BBR
f2cc 00 10    dw 0x1000 ;logical test start address
f2ce 00 e0    dw 0xe000 ;size of RAM block to test
f2d0
f2d0 74 f5    dw ramtxt18 ;test message
f2d2 fc       db 0xfc   ;BBR
f2d3 00 10    dw 0x1000 ;logical test start address
f2d5 00 20    dw 0x2000 ;size of RAM block to test
f2d7
f2d7 00 00    dw 0       ;end of table
f2d9
```

```
f2d9          ramtxtok:
f2d9 ..        db " - OK"
f2de .. 0a      db '\r', '\n'
f2e0 00        db 0
f2e1          ramtxtterr:
f2e1 ..        db " - ERROR"
f2e9 .. 0a      db '\r', '\n'
f2eb 00        db 0
f2ec          ramtxt0:
f2ec ..        db "Test physical RAM 0x00000 - 0x0efff"
f30f 00        db 0
f310          ramtxt1:
f310 ..        db "Test physical RAM 0x0f000 - 0x1cffff"
f333 00        db 0
f334          ramtxt2:
f334 ..        db "Test physical RAM 0x1d000 - 0x2afff"
f357 00        db 0
f358          ramtxt3:
f358 ..        db "Test physical RAM 0x2b000 - 0x38ffff"
f37b 00        db 0
f37c          ramtxt4:
f37c ..        db "Test physical RAM 0x39000 - 0x46ffff"
f39f 00        db 0
f3a0          ramtxt5:
f3a0 ..        db "Test physical RAM 0x47000 - 0x54ffff"
f3c3 00        db 0
f3c4          ramtxt6:
f3c4 ..        db "Test physical RAM 0x55000 - 0x62ffff"
f3e7 00        db 0
f3e8          ramtxt7:
f3e8 ..        db "Test physical RAM 0x63000 - 0x70ffff"
f40b 00        db 0
f40c          ramtxt8:
f40c ..        db "Test physical RAM 0x71000 - 0x7efff"
f42f 00        db 0
f430          ramtxt9:
f430 ..        db "Test physical RAM 0x7f000 - 0x8cffff"
f453 00        db 0
f454          ramtxt10:
f454 ..       db "Test physical RAM 0x8d000 - 0x9afff"
f477 00        db 0
f478          ramtxt11:
f478 ..       db "Test physical RAM 0x9b000 - 0xa8ffff"
f49b 00        db 0
f49c          ramtxt12:
f49c ..       db "Test physical RAM 0xa9000 - 0xb6ffff"
f4bf 00        db 0
f4c0          ramtxt13:
f4c0 ..       db "Test physical RAM 0xb7000 - 0xc4ffff"
f4e3 00        db 0
f4e4          ramtxt14:
f4e4 ..       db "Test physical RAM 0xc5000 - 0xd2ffff"
f507 00        db 0
f508          ramtxt15:
f508 ..       db "Test physical RAM 0xd3000 - 0xe0ffff"
f52b 00        db 0
f52c          ramtxt16:
f52c ..       db "Test physical RAM 0xe1000 - 0xeeffff"
f54f 00        db 0
f550          ramtxt17:
f550 ..       db "Test physical RAM 0xef000 - 0xfcffff"
f573 00        db 0
f574          ramtxt18:
f574 ..       db "Test physical RAM 0xfd000 - 0xfeffff"
f597 00        db 0
f598          ; The rest of RAM is for variables
f598          prgend:
f598          indindex:
f598 ..       db 0
f599          f599          cspattern:
```

```
| f599 00          db 0
| f59a
| f59a          ramerr:
| f59a 00          db 0
| f59b
# End of file test180.z80
f59b
```

2021-08-29

Added copyright and saved as test180\_14.z80

## Test of interrupt

---

2021-08-29

It is not (yet) fully clear for me how the interrupt vectors for internal devices work.

- [Z180 Q & A - z180faq.pdf](https://www-users.cs.york.ac.uk/~pcc/Circuits/64180/docs/z180faq.pdf) (<https://www-users.cs.york.ac.uk/~pcc/Circuits/64180/docs/z180faq.pdf>)
- [Help Z180 Interrupt Vector INT0 Mode 2](https://www.embeddedrelated.com/showthread/comp.arch.embedded/38109-1.php) (<https://www.embeddedrelated.com/showthread/comp.arch.embedded/38109-1.php>)
- [z88dk: create embedded z180 target · Issue #50 · z88dk/z88dk](https://github.com/z88dk/z88dk/issues/50) (<https://github.com/z88dk/z88dk/issues/50>)
- [NASCOM\\_BASIC\\_4.7/z180intr.asm at master · feilipu/NASCOM\\_BASIC\\_4.7](https://github.com/feilipu/NASCOM_BASIC_4.7/blob/master/includes/z180intr.asm) ([https://github.com/feilipu/NASCOM\\_BASIC\\_4.7/blob/master/includes/z180intr.asm](https://github.com/feilipu/NASCOM_BASIC_4.7/blob/master/includes/z180intr.asm))
- [NASCOM\\_BASIC\\_4.7/yaz180.inc at master · feilipu/NASCOM\\_BASIC\\_4.7](https://github.com/feilipu/NASCOM_BASIC_4.7/blob/master/includes/yaz180.inc) ([https://github.com/feilipu/NASCOM\\_BASIC\\_4.7/blob/master/includes/yaz180.inc](https://github.com/feilipu/NASCOM_BASIC_4.7/blob/master/includes/yaz180.inc))
- [UZI180](http://p112.sourceforge.net/uzidoc.html) (<http://p112.sourceforge.net/uzidoc.html>)

Z180 interrupt vectors

## Z8018x Family MPU User Manual

82



individual I/O (PRT, DMAC, CSI/O, ASCI) control register. The lower vector of INT1 INT2 and internal interrupt are summarized in Table 9.

**Table 9. Vector Table**

Interrupt Source	Priority	IL			Fixed Code				
		b7	b6	b5	b4	b3	b2	b1	b0
INT1	Highest	—	—	—	0	0	0	0	0
INT2		—	—	—	0	0	0	1	0
PRT channel 0		—	—	—	0	0	1	0	0
PRT channel 1		—	—	—	0	0	1	1	0
DMA channel 0		—	—	—	0	1	0	0	0
DMA channel 1		—	—	—	0	1	0	1	0
CSI/O		—	—	—	0	1	1	0	0
ASCI channel 0		—	—	—	0	1	1	1	0
ASCI channel 1		—	—	—	1	0	0	0	0

The time base for the counters is derived from the system clock (divided by 20) before reaching the counter.

2021-08-30

First test hang the program after "ei". Worked when commenting out initial "ei".

First tried with disabling INT0, but that did not help.

Maybe TCR and the counters must be read in the interrupt routine?

TIF0: Timer Interrupt Flag 0 (bit 6)—When TMDR0 decrements to 0 , TIF0 is set to 1 , and, when enabled by TIE0 = 1 , an interrupt request is generated TIF0 is reset to 0 when TCR is read and the higher or lower byte of TMDR0 is read. During RESET , TIF0 is cleared to 0 .

This made it! Now interrupt from timer 0 is working.

Saved as test180\_16.z80

```
# File test180.z80
0000      ; test180.z80
0000
0000      ; Test program for the Z180 computer
0000
0000      ; testing:
0000      ;   - simple MMU setup
0000      ;   - RAM as stack
0000      ;   - Serial output port 0 & 1
0000      ;   - Serial input port 0 & 1
```

```

0000      ; - MMU setup with Common Bank 0, Bank Area, Common Bank 1
0000      ; - simple RAM test
0000      ; - copy test program to RAM and execute
0000      ; - switch to low RAM using MEMSEL
0000      ; - test 74LS74 select outputs
0000      ; - test all RAM using MMU and MEMSEL
0000      ; - set no wait states for RAM access
0000      ; - interrupt test
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      ; Internal ports
0000
0000      ; ASCII Registers port 0 and 1
0000      CNTLA0: equ 0x0000 ;ASCII Channel Control Register A 0
0000      CNTLA1: equ 0x0001 ;ASCII Channel Control Register A 1
0000      CNTLB0: equ 0x0002 ;ASCII Control Register B 0
0000      CNTLB1: equ 0x0003 ;ASCII Control Register B 1
0000      STAT0: equ 0x0004 ;ASCII Status Register 0
0000      STAT1: equ 0x0005 ;ASCII Status Register 1
0000      TDR0:   equ 0x0006 ;ASCII Transmit Data Register 0
0000      TDR1:   equ 0x0007 ;ASCII Transmit Data Register 1
0000      RDR0:   equ 0x0008 ;ASCII Receive Data FIFO 0
0000      RDR1:   equ 0x0009 ;ASCII Receive Data FIFO 1
0000
0000      ; Timer Registers
0000      TMDR0L: equ 0x000c ;Timer Data Register Channel 0L
0000      TMDR0H: equ 0x000d ;Timer Data Register Channel 0H
0000      RLDR0L: equ 0x000e ;Timer Reload Register 0L
0000      RLDR0H: equ 0x000f ;Timer Reload Register 0H
0000      TCR:    equ 0x0010 ;Timer Control Register
0000      TMDR1L: equ 0x0014 ;Timer Data Register Channel 1L
0000      TMDR1H: equ 0x0015 ;Timer Data Register Channel 1H
0000      RLDR1L: equ 0x0016 ;Timer Reload Register 1L
0000      RLDR1H: equ 0x0017 ;Timer Reload Register 1H
0000
0000      ; DMA/WAIT Control Register
0000      DCNTL:  equ 0x0032 ;DMA/WAIT Control Register
0000
0000      ; Interrupt Control
0000      IL:     equ 0x0033 ;Interrupt Vector Low Register
0000      ITC:    equ 0x0034 ;Int/TRAP Control Register
0000
0000      ; MMU Registers
0000      CBR:    equ 0x0038 ;MMU Common Base Register
0000      BBR:    equ 0x0039 ;MMU Bank Base Register
0000      CBAR:   equ 0x003a ;MMU Common/Bank Area Register
0000
0000      ; External ports
0000
0000      ;Select EPROM or RAM on address 0x0000 - 0x3ffff
0000      ROMSEL: equ 0x40    ;Write selects EPROM (reset condition)
0000      RAMSEL: equ 0x41    ;Write selects RAM
0000
0000      ;LED on/off
0000      LEDOFF: equ 0x42    ;Write turns LED off (reset condition)
0000      LEDON:  equ 0x43    ;Write turns LED on
0000
0000      ;SPI device select and AVR reset
0000      CSPORT: equ 0x44    ;Write to bit 0 - 3 (reset condition, all 0)
0000          ;Bit 0: select SD_CS0 when set to 1
0000          ;Bit 1: select SD_CS1 when set to 1
0000          ;Bit 2: select ATSS (AVR) when set to 1
0000          ;Bit 1: reset AVR when set to 1
0000          ;        if JP8 pin 2-3 connected
0000
0000
0000      boot:
0000
0000      init:
0000
0000      ; Set up the MMU initially
0000      ;
0000      ; Common Bank 0, 4KB
0000      ;   logical: 0x0000 - 0x0fff
0000      ;   physical: 0x00000 - 0x0ffff, EPROM or start of low RAM if enabled
0000      ; Bank Area, 56KB
0000      ;   logical: 0x1000 - 0xefff
0000      ;   physical: 0x41000 - 0x4efff, low RAM chip above EPROM
0000      ; Common Bank 1, 4KB
0000      ;   logical: 0xf000 - 0xfffff
0000      ;   physical: 0xff000 - 0xfffff, end of high RAM chip
0000
0000      ; The MMU function is a bit mysterious but I learned that CBAR
0000      ; must be configured before CBR and BBR otherwise strange
0000      ; things will happen.
0000
0000      3e f1  ;<CA><BA>
0002 01 3a 00

```

```
0005 ed 79          out (c), a
0007 3e f0          ld a, 0xf0
0009 01 38 00        ld bc, CBR
000c ed 79          out (c), a
000e 3e 40          ld a, 0x40
0010 01 39 00        ld bc, BBR
0013 ed 79          out (c), a
0015
0015             HIRAM: equ 0xf000
0015
0015 ; copy the program to high RAM
0015 01 52 06        ld bc,prgend - prgstart
0018 21 2e 00        ld hl,prgineprom
001b 11 00 f0        ld de,HIRAM
001e
001e 78            cloop:
001f b1            ld a,b
0020 ca 2b 00        or c
0023 7e            jp z,cpend
0024 23            ld a,(hl)
0025 12            inc hl
0026 13            ld (de),a
0027 0b            inc de
0028 c3 1e 00        dec bc
002b
002b c3 00 f0        jp cploop
002e
002e cpend:
002e             jp prgstart ; jump to the copied code
002e
002e prgineprom:
002e             org HIRAM
002e
002e prgstart:
002e
002e ; Set up Stack Pointer (first push/call will wrap to 0xffff)
002e f000 31 00 00    ld sp, 0x0000
003
003 f003
003 f003 01 32 00    ; When running from RAM configure no memory wait states
003 f006 ed 78        ld bc, DCNTL
003 f008 e6 3f        in a, (c)
003 f00a ed 79        and 00111111b ; reset MVI0 and MVI1
003 f00c
003 f00c             out (c), a
003 f00c
003 f00c ; Initialize devices
003 f00c ; and blink LED
003 f00c
003 f00c 06 01        ld b, 1      ;1 blink, MMU initialized
003 f00e cd e0 f0        call ledblink
003 f011 06 02        ld b, 2
003 f013 cd fd f0        call delays
003 f016
003 f016 cd 6a f0        call asci0init
003 f019 06 02        ld b, 2      ;2 blinks, ASCII 0 initialized
003 f01b cd e0 f0        call ledblink
003 f01e 06 02        ld b, 2
003 f020 cd fd f0        call delays
003 f023
003 f023 cd a5 f0        call ascilinit
003 f026 06 03        ld b, 3      ;3 blinks, ASCII 1 initialized
003 f028 cd e0 f0        call ledblink
003 f02b 06 02        ld b, 2
003 f02d cd fd f0        call delays
003 f030
003 f030 3e 00        ld a, 0      ;set print indicator index
003 f032 32 52 f6        ld (indindex), a
003 f035
003 f035 3e 11        ld a, 00010001b ;set rotating bits for 74LS74 CS output
003 f037 32 53 f6        ld (cspattern), a
003 f03a
003 f03a 3e 00        ld a, 0      ;reset interrupt indicator
003 f03c 32 55 f6        ld (gotint), a
003 f03f
003 f03f ; Disable all external interrupts
003 f03f 01 34 00        ld bc, ITC
003 f042 3e 00        ld a, 00000000b
003 f044 ed 79        out (c), a
003 f046
003 f046 01 0c 00        ld bc, TMDROL
003 f049 e6 ff        and 0xff      ; set low timer 0
003 f04b ed 79        out (c), a
003 f04d 01 0d 00        ld bc, TMDR0H
003 f050 e6 ef        and 0xef      ; set high timer 0
003 f052 ed 79        out (c), a
003 f054
003 f054 01 10 00        ld bc, TCR
003 f057 3e 11        ld a, 00010001b ; bit 4 TIE0 (Timer Interrupt Enable 0)
003 f059
003 f059 ed 79        ld a, 00010001b ; bit 0 TDE0 (Timer Down Count Enable 0)
003 f059 out (c), a
003 f05b
003 f05b ; Initialize internal interrupts and enable
003 f05b initinit:
003 f05b 3e f6        ld a, ivblock / 256
003 f05d ed 47        ld i, a
```

```

| f05f 3e 40          ld a, ivblock & 0x00ff
| f061 01 33 00        ld bc, IL
| f064 ed 79          out (c), a
| f066 fb             ei
| f067
| f067 c3 3b f1        jp testloop
| f06a
| f06a
| f06a ; ASCII routines
| f06a
| f06a ; Initialize port 0
| f06a asci0init:
| f06a 3e 64          ld a, 01100100b
| f06c           ; bit 7 = 0: MPE - disabled
| f06c           ; bit 6 = 1: RE - Rx enabled
| f06c           ; bit 5 = 1: TE - Tx enabled
| f06c           ; bit 4 = 0: RTS0 - set to low, RTS active (?)
| f06c           ; bit 3 = 0: MPBR/EFR - not used
| f06c           ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
| f06c 01 00 00        ld bc, CNTLАО
| f06f ed 79          out (c), a
| f071
| f071 ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
| f071 ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
| f071 3e 21          ld a, 00100001b
| f073           ; bit 7 = 0: MPBT - disabled
| f073           ; bit 6 = 0: MP - disabled
| f073           ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
| f073           ; bit 4 = 0: PEO - ignored as no parity configured
| f073           ; bit 3 = 0: DR - Clock factor = 16
| f073           ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
| f073 01 02 00        ld bc, CNTLB0
| f076 ed 79          out (c), a
| f078
| f078 c9            ret
| f079
| f079 ; Output a character on port 0
| f079 ; reg E contains character to output
| f079 asci0putc:
| f079 01 04 00        ld bc, STAT0
| f07c ed 78          in a, (c)
| f07e e6 02          and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
| f080 28 f7          jr z, asci0putc ;not empty yet
| f082 7b             ld a, e
| f083 01 06 00        ld bc, TDR0
| f086 ed 79          out (c), a ;output character
| f088 c9            ret
| f089
| f089 ; Input a character from port 0
| f089 ; reg E contains the character
| f089 ; if E == 0 no character is available
| f089 asci0getc:
| f089 1e 00          ld e, 0
| f08b 01 04 00        ld bc, STAT0
| f08e ed 78          in a, (c)
| f090 e6 80          and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
| f092 c8            ret z ;empty
| f093 7b             ld a, e
| f094 01 08 00        ld bc, RDR0
| f097 ed 78          in a, (c) ;input character
| f099 5f             ld e, a
| f09a c9            ret
| f09b
| f09b ; Output a character string on port 0
| f09b ; reg HL points to string to output
| f09b ; the string is ended by 0
| f09b asci0pstr:
| f09b 7e              ld a, (hl)
| f09c b7              or a
| f09d c8              ret z
| f09e 5e              ld e, (hl)
| f09f 23              inc hl
| f0a0 cd 79 f0          call asci0putc
| f0a3 18 f6          jr asci0pstr
| f0a5
| f0a5 ; Initialize port 1
| f0a5 asci1init:
| f0a5 3e 64          ld a, 01100100b
| f0a7           ; bit 7 = 0: MPE - disabled
| f0a7           ; bit 6 = 1: RE - Rx enabled
| f0a7           ; bit 5 = 1: TE - Tx enabled
| f0a7           ; bit 4 = 0: RTS0 - set to low, RTS active (?)
| f0a7           ; bit 3 = 0: MPBR/EFR - not used
| f0a7           ; bit 2 - 0 = 100 - Start + 8 bit data + 1 stop
| f0a7 01 01 00        ld bc, CNTLAI
| f0aa ed 79          out (c), a
| f0ac
| f0ac ; set Baudrate: PHI / (PS * DR * SS) = Baud Rate
| f0ac ; 9216000 Hz / (30 * 16 * 2) = 9600 Baud
| f0ac 3e 21          ld a, 00100001b
| f0ae           ; bit 7 = 0: MPBT - disabled
| f0ae           ; bit 6 = 0: MP - disabled
| f0ae           ; bit 5 = 1: CTS/PS - prescale = 30 (PS as SS2-0 are not 111)
| f0ae           ; bit 4 = 0: PEO - ignored as no parity configured

```

```
f0ae ; bit 3 = 0: DR - Clock factor = 16
f0ae ; bit 2 - 0 = 001: SS2-SS0 - Divide Ratio: 2
f0ae 01 03 00 ld bc, CNTLBl
f0b1 ed 79 out (c), a
f0b3
f0b3 c9 ret
f0b4
f0b4 ; Output a character on port 1
f0b4 ; reg E contains character to output
f0b4 asciilputc:
f0b4 01 05 00 ld bc, STAT1
f0b7 ed 78 in a, (c)
f0b9 e6 02 and 00000010b ;test bit 1 = TDRE: Transmit Data Register Empty
f0bb 28 f7 jr z, asciilputc ;not empty yet
f0bd 7b ld a, e
f0be 01 07 00 ld bc, TDR1
f0c1 ed 79 out (c), a ;output character
f0c3 c9 ret
f0c4
f0c4 ; Input a character from port 1
f0c4 ; reg E contains the character
f0c4 ; if E == 0 no character is available
f0c4 asciilgetc:
f0c4 1e 00 ld e, 0
f0c6 01 05 00 ld bc, STAT1
f0c9 ed 78 in a, (c)
f0cb e6 80 and 10000000b ;test bit 7 = RDRF: Recieve data in FIFO
f0cd c8 ret z ;empty
f0ce 7b ld a, e
f0cf 01 09 00 ld bc, RDR1
f0d2 ed 78 in a, (c) ;input character
f0d4 5f ld e, a
f0d5 c9 ret
f0d6
f0d6 ; Output a character string on port 1
f0d6 ; reg HL points to string to output
f0d6 ; the string is ended by 0
f0d6 asciipstr:
f0d6 7e ld a, (hl)
f0d7 b7 or a
f0d8 c8 ret z
f0d9 5e ld e, (hl)
f0da 23 inc hl
f0db cd b4 f0 call asciilputc
f0de 18 f6 jr asciipstr
f0e0
f0e0 ; ledblink flashes the LED
f0e0 ; number of times in B reg
f0e0 ; using reg: a, b, hl
f0e0 ledblink:
f0e0 21 ff ff ld hl, 0xffff
f0e3 3e 01 ld a, 1
f0e5 d3 43 out(LEDON), a
f0e7 ledonloop:
f0e7 2b dec hl
f0e8 7c ld a, h
f0e9 b5 or l
f0ea c2 e7 f0 jp nz, ledonloop
f0ed 21 ff ff ld hl, 0xffff
f0f0 3e 00 ld a, 0
f0f2 d3 42 out(LEDOFF), a
f0f4 ledoffloop:
f0f4 2b dec hl
f0f5 7c ld a, h
f0f6 b5 or l
f0f7 c2 f4 f0 jp nz, ledoffloop
f0fa 10 e4 djnz ledblink
f0fc c9 ret
f0fd
f0fd ; delays makes a delay multiplied
f0fd ; by number of times in B reg
f0fd ; using reg: a, b, hl
f0fd delays:
f0fd 21 ff ff ld hl, 0xffff
f100 delaylloop:
f100 2b dec hl
f101 7c ld a, h
f102 b5 or l
f103 c2 00 f1 jp nz, delaylloop
f106 21 ff ff ld hl, 0xffff
f109 delay2loop:
f109 2b dec hl
f10a 7c ld a, h
f10b b5 or l
f10c c2 09 f1 jp nz, delay2loop
f10f 10 ec djnz delays
f111 c9 ret
f112
f112 ; Test RAM
f112 ; reg bc: number of bytes to test
f112 ; reg hl: start address of test
f112 ; reg a; returns 0 if RAM ok, 1 if error
f112 ramtest:
```

```
f112 3e 00          ld a,0           ; reset RAM test error flag
f114 32 54 f6      ld (ramerr),a
f117
f117 1e 00          ramtstop:
f119 73             ld e,0x00
f11a 7e             ld (hl),e
f11b bb             ld a,(hl)
f11c ca 24 f1      cp e
f11f 3e 01          jp z,ramtstff
f121 32 54 f6      ld a,1
f124
f124 1e ff          ld (ramerr),a
f126 73             ramtstff:
f127 7e             ld e,0xff
f128 bb             ld (hl),e
f129 ca 31 f1      ld a,(hl)
f12c 3e 01          cp e
f12e 32 54 f6      jp z,ramtstnxt
f131
f131 23             ld a,1
f132 0b             ld (ramerr),a
f133 78             ramtstnxt:
f134 b1             inc hl
f135 20 e0          dec bc
f137 3a 54 f6      ld a,b
f13a c9             or c
f13b
f13b 06 01          jr nz, ramtstop
f13d cd e0 f0      ld a, (ramerr)
f140 06 01          ret
f142 cd fd f0
f145
f145 21 35 f2      testloop:
f148 cd 9b f0      ld b, 1
f14b 21 4f f2      call ledblink
f14e 06 00          ld b, 1
f150 3a 52 f6      call delays
f153 4f             ld hl, asci0txt
f154 09             call asci0ptr
f155 5e             ld hl, indicator
f156 cd 79 f0      ld b, 0
f159 21 53 f2      ld a, (indindex)
f15c cd 9b f0      ld c, a
f15f cd 89 f0      add hl, bc
f162 7b             ld e, (hl)
f163 b7             ld e, (hl)
f164 ca 70 f1      call asci0putc
f167 cd 79 f0      ld hl, built
f16a 21 92 f2      call asci0ptr
f16d cd 9b f0      call asci0getc
f170
f170 21 42 f2      ascionoin:
f173 cd d6 f0      ld hl, asci0txt
f176 21 4f f2      call asci0ptr
f179 06 00          ld hl, indicator
f17b 3a 52 f6      ld b, 0
f17e 4f             ld a, (indindex)
f17f 09             ld c, a
f180 5e             add hl, bc
f181 cd b4 f0      ld e, (hl)
f184 21 53 f2      call asci0putc
f187 cd d6 f0      ld hl, built
f18a cd c4 f0      call asci0ptr
f18d 7b             ld a, e
f18e b7             or a
f18f ca 9b f1      jp z, ascilnoin
f192 cd b4 f0      call asci0putc
f195 21 92 f2      ld hl, inptxt
f198 cd d6 f0      call ascilptr
f19b
f19b 3a 52 f6      ascilnoin:
f19e 3c             ld a, (indindex)
f19f e6 03          inc a
f1a1 32 52 f6      and 00000011b
f1a4
f1a4 3a 53 f6      ld (indindex), a
f1a7 d3 44          ;Chip Select and reset outputs
f1a9 07             ld a, (cspattern)
f1aa 32 53 f6      out (CSPORT), a
f1ad
f1ad 3e 01          rlca
f1ad 3e 01          ld (cspattern), a
f1af d3 41          ; Test RAM
f1af 3e 01          ; Enable RAM in low memory
f1af d3 41          ld a, 1
f1af d3 41          out (RAMSEL), a ; select RAM instead of EPROM from address 0
f1b1
f1b1 3e 01          ; Test physical RAM
```

```
f1b1 dd 21 d8 f2          ld ix, ramtsttab
f1b5 dd 6e 00          ramloop:
f1b8 dd 66 01          ld l, (ix + 0)
f1bb 7c          ld h, (ix + 1)
f1bc b5          ld a, h
f1bd 28 47          or l
f1bf dd 7e 02          jr z, ramtend ; end of test table
f1c2 01 39 00          ld a, (ix + 2)
f1c5 ed 79          ld bc, BBR
f1c7          out (c), a

f1c7 cd 9b f0          call asci0pstr
f1ca dd 6e 00          ld l, (ix + 0)
f1cd dd 66 01          ld h, (ix + 1)
f1d0 cd d6 f0          call asci0pstr
f1d3 dd 6e 03          ld l, (ix + 3)
f1d6 dd 66 04          ld h, (ix + 4)
f1d9 dd 4e 05          ld c, (ix + 5)
f1dc dd 46 06          ld b, (ix + 6)
f1df cd 12 f1          call ramtest
f1e2 b7          or a
f1e3 28 0e          jr z, ramisok
f1e5 21 67 f3          ld hl, ramtxterr
f1e8 cd 9b f0          call asci0pstr
f1eb 21 67 f3          ld hl, ramtxterr
f1ee cd d6 f0          call asci0pstr
f1f1 18 0c          jr ramtnxt
f1f3          ramisok:
f1f3 21 5f f3          ld hl, ramtxtok
f1f6 cd 9b f0          call asci0pstr
f1f9 21 5f f3          ld hl, ramtxtok
f1fc cd d6 f0          call asci0pstr
f1ff          ramtnxt:
f1ff 01 07 00          ld bc, 7
f202 dd 09          add ix, bc
f204 18 af          jr ramloop
f206          ramtend:
f206          ; Test if interrupt received and print
f206 3a 55 f6          ld a, (gotint) ; Interrupt received?
f209 b7          or a
f20a ca 1b f2          jp z, prtnoint ; no
f20d 21 a5 f2          ld hl, int_msg
f210 cd 9b f0          call asci0pstr
f213 21 a5 f2          ld hl, int_msg
f216 cd d6 f0          call asci0pstr
f219 18 0c          jr prtintend
f21b          prtnoint:
f21b 21 bd f2          ld hl, no_int_msg
f21e cd 9b f0          call asci0pstr
f221 21 bd f2          ld hl, no_int_msg
f224 cd d6 f0          call asci0pstr
f227          prtintend:
f227          ; Enable EPROM in low memory
f227 3e 00          ld a, 0
f229 d3 40          out (ROMSEL), a
f22b          f22b          ; Restore Bank Area
f22b 3e 40          ld a, 0x40
f22d 01 39 00          ld bc, BBR
f230 ed 79          out (c), a
f232          f232 c3 3b f1          jp testloop
f235          f235          asci0txt:
f235 ..          db "ASCI port 0 "
f241 00          db 0
f242          f242          asciltxt:
f242 ..          db "ASCI port 1 "
f24e 00          db 0
f24f          f24f          indicator:
f24f .. 2f 2d 5c          db '|', '/', '-', '\\'
f253          f253          built:
f253 ..          db " Test program for Z180 computer v1.0"
f277          f277          include "tbuilt180.z80"
f277 ..          db ", built 2021-08-30 16:17"
# End of file tbuilt180.z80
f28f .. 0a          db '\r', '\n'
f291 00          db 0
f292          f292          inptxt:
f292 ..          db "<- was received"
f2a2 .. 0a          db '\r', '\n'
f2a4 00          db 0
f2a5          f2a5          int_msg:
f2a5 .. 00          db "Interrupt from timer 0\r\n", 0
f2bd          f2bd          no_int_msg:
f2bd .. 00          db "No interrupt from timer 0\r\n", 0
f2d8
```

```
f2d8 ; RAM test table
f2d8 ramtsttab:
f2d8 72 f3 dw ramtxt0 ;test message, end of table if NUL
f2da 00 db 0x00 ;BBR
f2db 00 00 dw 0x0000 ;logical test start address
f2dd 00 f0 dw 0xf000 ;size of RAM block to test
f2df
f2df 96 f3 dw ramtxt1 ;test message
f2e1 0e db 0xe ;BBR
f2e2 00 10 dw 0x1000 ;logical test start address
f2e4 00 e0 dw 0xe000 ;size of RAM block to test
f2e6
f2e6 ba f3 dw ramtxt2 ;test message
f2e8 1c db 0x1c ;BBR
f2e9 00 10 dw 0x1000 ;logical test start address
f2eb 00 e0 dw 0xe000 ;size of RAM block to test
f2ed
f2ed de f3 dw ramtxt3 ;test message
f2ef 2a db 0x2a ;BBR
f2f0 00 10 dw 0x1000 ;logical test start address
f2f2 00 e0 dw 0xe000 ;size of RAM block to test
f2f4
f2f4 02 f4 dw ramtxt4 ;test message
f2f6 38 db 0x38 ;BBR
f2f7 00 10 dw 0x1000 ;logical test start address
f2f9 00 e0 dw 0xe000 ;size of RAM block to test
f2fb
f2fb 26 f4 dw ramtxt5 ;test message
f2fd 46 db 0x46 ;BBR
f2fe 00 10 dw 0x1000 ;logical test start address
f300 00 e0 dw 0xe000 ;size of RAM block to test
f302
f302 4a f4 dw ramtxt6 ;test message
f304 54 db 0x54 ;BBR
f305 00 10 dw 0x1000 ;logical test start address
f307 00 e0 dw 0xe000 ;size of RAM block to test
f309
f309 6e f4 dw ramtxt7 ;test message
f30b 62 db 0x62 ;BBR
f30c 00 10 dw 0x1000 ;logical test start address
f30e 00 e0 dw 0xe000 ;size of RAM block to test
f310
f310 92 f4 dw ramtxt8 ;test message
f312 70 db 0x70 ;BBR
f313 00 10 dw 0x1000 ;logical test start address
f315 00 e0 dw 0xe000 ;size of RAM block to test
f317
f317 b6 f4 dw ramtxt9 ;test message
f319 7e db 0x7e ;BBR
f31a 00 10 dw 0x1000 ;logical test start address
f31c 00 e0 dw 0xe000 ;size of RAM block to test
f31e
f31e da f4 dw ramtxt10 ;test message
f320 8c db 0x8c ;BBR
f321 00 10 dw 0x1000 ;logical test start address
f323 00 e0 dw 0xe000 ;size of RAM block to test
f325
f325 fe f4 dw ramtxt11 ;test message
f327 9a db 0x9a ;BBR
f328 00 10 dw 0x1000 ;logical test start address
f32a 00 e0 dw 0xe000 ;size of RAM block to test
f32c
f32c 22 f5 dw ramtxt12 ;test message
f32e a8 db 0xa8 ;BBR
f32f 00 10 dw 0x1000 ;logical test start address
f331 00 e0 dw 0xe000 ;size of RAM block to test
f333
f333 46 f5 dw ramtxt13 ;test message
f335 b6 db 0xb6 ;BBR
f336 00 10 dw 0x1000 ;logical test start address
f338 00 e0 dw 0xe000 ;size of RAM block to test
f33a
f33a 6a f5 dw ramtxt14 ;test message
f33c c4 db 0xc4 ;BBR
f33d 00 10 dw 0x1000 ;logical test start address
f33f 00 e0 dw 0xe000 ;size of RAM block to test
f341
f341 8e f5 dw ramtxt15 ;test message
f343 d2 db 0xd2 ;BBR
f344 00 10 dw 0x1000 ;logical test start address
f346 00 e0 dw 0xe000 ;size of RAM block to test
f348
f348 b2 f5 dw ramtxt16 ;test message
f34a e0 db 0xe0 ;BBR
f34b 00 10 dw 0x1000 ;logical test start address
f34d 00 e0 dw 0xe000 ;size of RAM block to test
f34f
f34f d6 f5 dw ramtxt17 ;test message
f351 ee db 0xee ;BBR
f352 00 10 dw 0x1000 ;logical test start address
f354 00 e0 dw 0xe000 ;size of RAM block to test
f356
f356 fa f5 dw ramtxt18 ;test message
```

```
| f358 fc          db 0xfc      ;BBR
f359 00 10        dw 0x1000    ;logical test start address
f35b 00 20        dw 0x2000    ;size of RAM block to test
f35d
f35d 00 00        dw 0          ;end of table
f35f
f35f             ramtxtok:
f35f ..          db " - OK"
f364 .. 0a        db '\r', '\n'
f366 00
f367
f367             ramtxterr:
f367 ..          db " - ERROR"
f36f .. 0a        db '\r', '\n'
f371 00
f372
f372             ramtxt0:
f372 ..          db "Test physical RAM 0x00000 - 0x0effff"
f395 00
f396
f396             ramtxt1:
f396 ..          db "Test physical RAM 0x0f000 - 0x1cffff"
f3b9 00
f3ba
f3ba             ramtxt2:
f3ba ..          db "Test physical RAM 0x1d000 - 0x2affff"
f3dd 00
f3de
f3de             ramtxt3:
f3de ..          db "Test physical RAM 0x2b000 - 0x38ffff"
f401 00
f402
f402             ramtxt4:
f402 ..          db "Test physical RAM 0x39000 - 0x46ffff"
f425 00
f426
f426             ramtxt5:
f426 ..          db "Test physical RAM 0x47000 - 0x54ffff"
f449 00
f44a
f44a             ramtxt6:
f44a ..          db "Test physical RAM 0x55000 - 0x62ffff"
f46d 00
f46e
f46e             ramtxt7:
f46e ..          db "Test physical RAM 0x63000 - 0x70ffff"
f491 00
f492
f492             ramtxt8:
f492 ..          db "Test physical RAM 0x71000 - 0x7effff"
f4b5 00
f4b6
f4b6             ramtxt9:
f4b6 ..          db "Test physical RAM 0x7f000 - 0x8cffff"
f4d9 00
f4da
f4da             ramtxt10:
f4da ..          db "Test physical RAM 0x8d000 - 0x9affff"
f4fe
f4fe             ramtxt11:
f4fe ..          db "Test physical RAM 0x9b000 - 0xa8ffff"
f521 00
f522
f522             ramtxt12:
f522 ..          db "Test physical RAM 0xa9000 - 0xb6ffff"
f545 00
f546
f546             ramtxt13:
f546 ..          db "Test physical RAM 0xb7000 - 0xc4ffff"
f569 00
f56a
f56a             ramtxt14:
f56a ..          db "Test physical RAM 0xc5000 - 0xd2ffff"
f58d 00
f58e
f58e             ramtxt15:
f58e ..          db "Test physical RAM 0xd3000 - 0xe0ffff"
f5b1 00
f5b2
f5b2             ramtxt16:
f5b2 ..          db "Test physical RAM 0xe1000 - 0xeeffff"
f5d5 00
f5d6
f5d6             ramtxt17:
f5d6 ..          db "Test physical RAM 0xef000 - 0xfcffff"
f5f9 00
f5fa
f5fa             ramtxt18:
f5fa ..          db "Test physical RAM 0xfd000 - 0xfeffff"
f61d 00
f61e
f61e             ; Timer interrupt
```

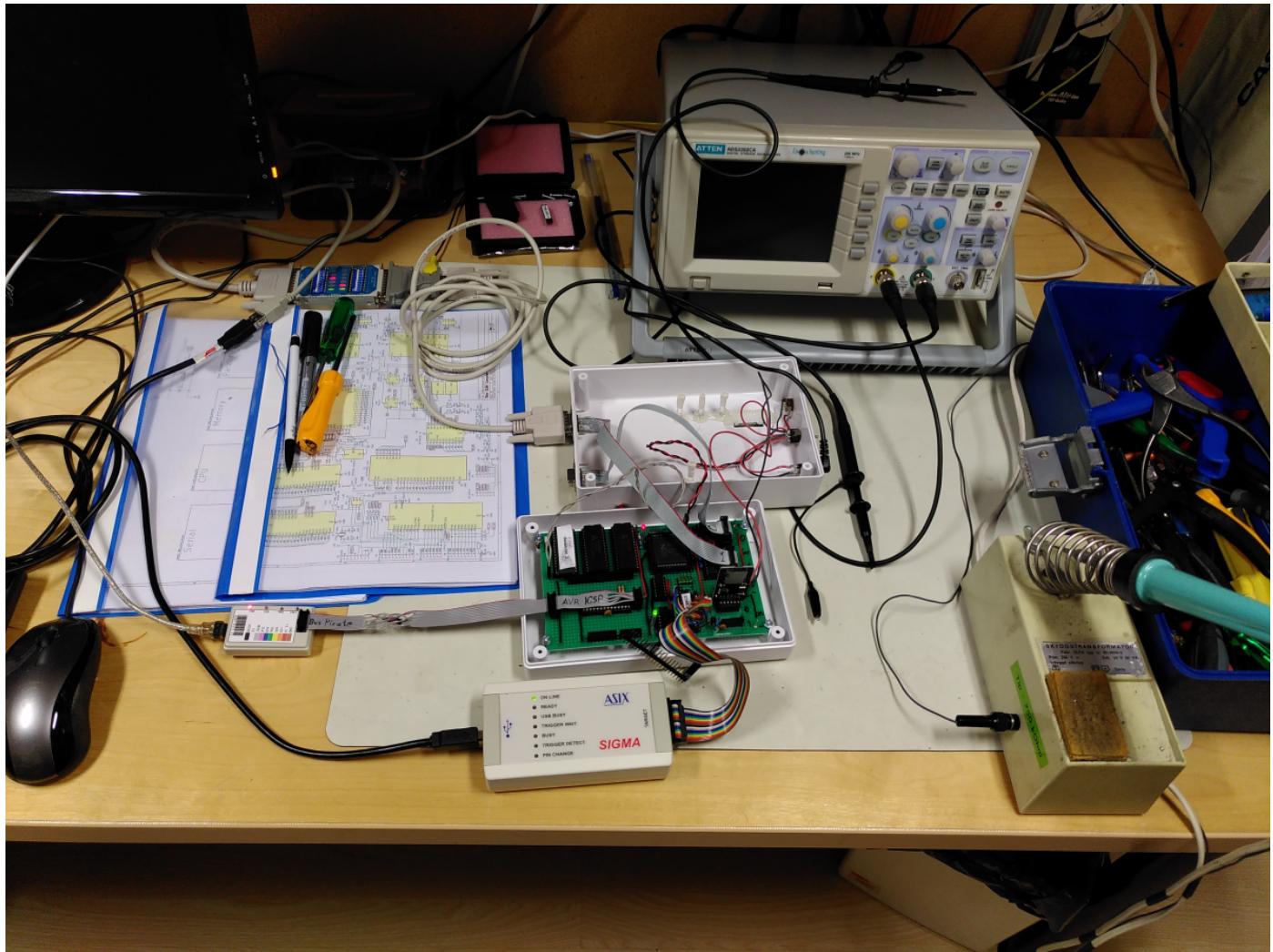
```
f61e          inttimer:
f61e f5          push af
f61f c5          push bc
f620 3e 01          ld a, 1
f622 32 55 f6          ld (gotint), a
f625 01 10 00          ld bc, TCR
f628 ed 78          in a, (c)
f62a 01 0c 00          ld bc, TMDROL
f62d ed 78          in a, (c)
f62f 01 0d 00          ld bc, TMDROH
f632 ed 78          in a, (c)
f634 c1          pop bc
f635 f1          pop af
f636 fb          ei
f637 ed 4d          reti
f639
f639          ; Interrupt routines, dummy for unused devices
f639          intdummy:
f639 fb          ei
f63a ed 4d          reti
f63c
f63c          ; Interrupt vectors for internal devices
f63c          ; make sure that the block is on an even 32 byte address
ivstart:
f63c          if ivstart & 0x001f
f63c 0x00...
f640          ds 32 - (ivstart & 0x001f), 0x00
endif
f640
f640          ivblock:
f640 39 f6          ivint1:    dw intdummy
f642 39 f6          ivint2:    dw intdummy
f644 1e f6          ivprt0:   dw inttimer
f646 39 f6          ivprt1:   dw intdummy
f648 39 f6          ivdma0:   dw intdummy
f64a 39 f6          ivdma1:   dw intdummy
f64c 39 f6          ivcsio:   dw intdummy
f64e 39 f6          ivasci0:  dw intdummy
f650 39 f6          ivasci1:  dw intdummy
f652
f652          prgend:    ;end address for copied code and constants from EPROM to RAM
f652
f652          ; The rest of RAM is for variables and stack
f652          indindex:  ;indicator index for printout
f652 00          db 0
f653
f653          cspattern: ;rotating bits for 74LS74 CS output
f653 00          db 0
f654
f654          ramerr:    ;RAM test error flag
f654 00          db 0
f655
f655          gotint:    ;interrupt indicator
f655 00          db 0
f656
# End of file test180.z80
```

# AVR test programs

## Simple blink and serial test

2021-08-16

Uploaded to the AVR trough the ICSP interface using Bus Pirate.



```

/*
Blink_diy

Blinks a LED and prints a message on the serial port.

Most Arduinos have an on-board LED you can control.
On the UNO, MEGA and ZERO it is attached to
digital pin 13, LED_BUILTIN is set to the correct LED pin
independent of which board is used.
So also on the Ardini Uno look-alike on the Z180 computer board.

This is derived from example code that is in the public domain.

http://www.arduino.cc/en/Tutorial/Blink
*/
int loopcnt;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);

  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // prints title with ending line break
  Serial.println("My DIY Arduino");

  loopcnt = 2;
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(250);                      // wait for a 1/4 second
  digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
  delay(250);                      // wait for a 1/4 second
  digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(250);                      // wait for a 1/4 second
}

```

```
| digitalWrite(LED_BUILTIN, LOW);      // turn the LED off by making the voltage LOW
| delay(1000);                      // wait for a second
| Serial.print("blinked ");
| Serial.print(loopcnt);
| Serial.println(" times");
| loopcnt += 2;
| }
```

---

Retrieved from '[http://192.168.42.21/mediawiki/index.php?title=Z180\\_computer\\_test\\_programs&oldid=8616](http://192.168.42.21/mediawiki/index.php?title=Z180_computer_test_programs&oldid=8616)'

---

**This page was last modified on 30 August 2021, at 16:25.**