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

AVR test programs
- Simple blink and serial test

Z180 test programs

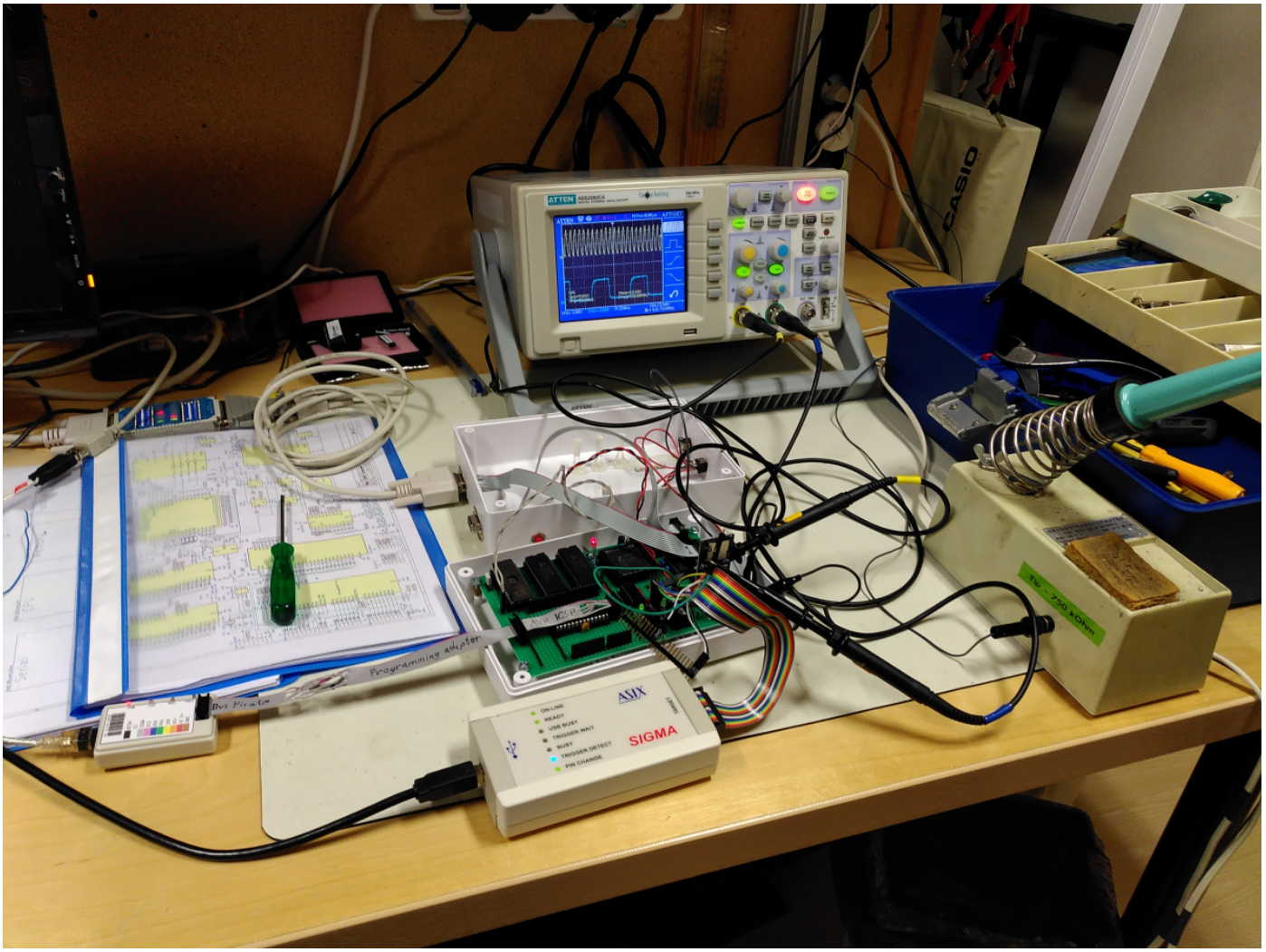
Test program tools

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                    boot:
0000                    mloop:
0000 21 ff ff               ld hl,0xffff
0003 3e 01                  ld a,1
0005 d3 43                  out(0x43),a ; LED on
0007                    onloop:
0007 2b                     dec hl
0008 7c                     ld a,h
0009 b5                     or l
000a c2 07 00               jp nz,onloop
000d 21 ff ff               ld hl,0xffff
0010 3e 00                  ld a,0
0012 d3 42                  out(0x42),a ; LED off
0014                    offloop:
0014 2b                     dec hl
0015 7c                     ld a,h
0016 b5                     or l
0017 c2 14 00               jp nz,offloop
001a
001a c3 00 00               jp mloop
001d
```

```
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]: 03FBD1E6h - 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]: 03FBD1E6h - 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: 0x00000 - 0x03fff
0000                    ; Bank Area
0000                    ;    logical: 0x4000 - 0xbfff
0000                    ;    physical: 0x74000 - 0x7bfff
0000                    ; Common Bank 1
0000                    ;    logical: 0xc000 - 0xffff
0000                    ;    physical: 0xfc000 - 0xfffff
0000
0000                    ; test by setting MMU regs to reset values
0000 3e 00                  ld a,0x00
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                    mloop:
0015 21 ff ff               ld hl,0xffff
0018 3e 01                  ld a,1
001a d3 43                  out(0x43),a ; LED on
001c                    onloop:
001c 2b                     dec hl
001d 7c                     ld a,h
001e b5                     or l
001f c2 1c 00               jp nz,onloop
0022 21 ff ff               ld hl,0xffff
0025 3e 00                  ld a,0
0027 d3 42                  out(0x42),a ; LED off
0029                    offloop:
0029 2b                     dec hl
002a 7c                     ld a,h
002b b5                     or l
002c c2 29 00               jp nz,offloop
002f
002f c3 15 00               jp mloop
0032
0032                    built:
0032                        include "built180.z80"
0032 ..                     db ", Built 2021-08-13 17:03"
# End of file built180.z80
004a 00                     db 0
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
```

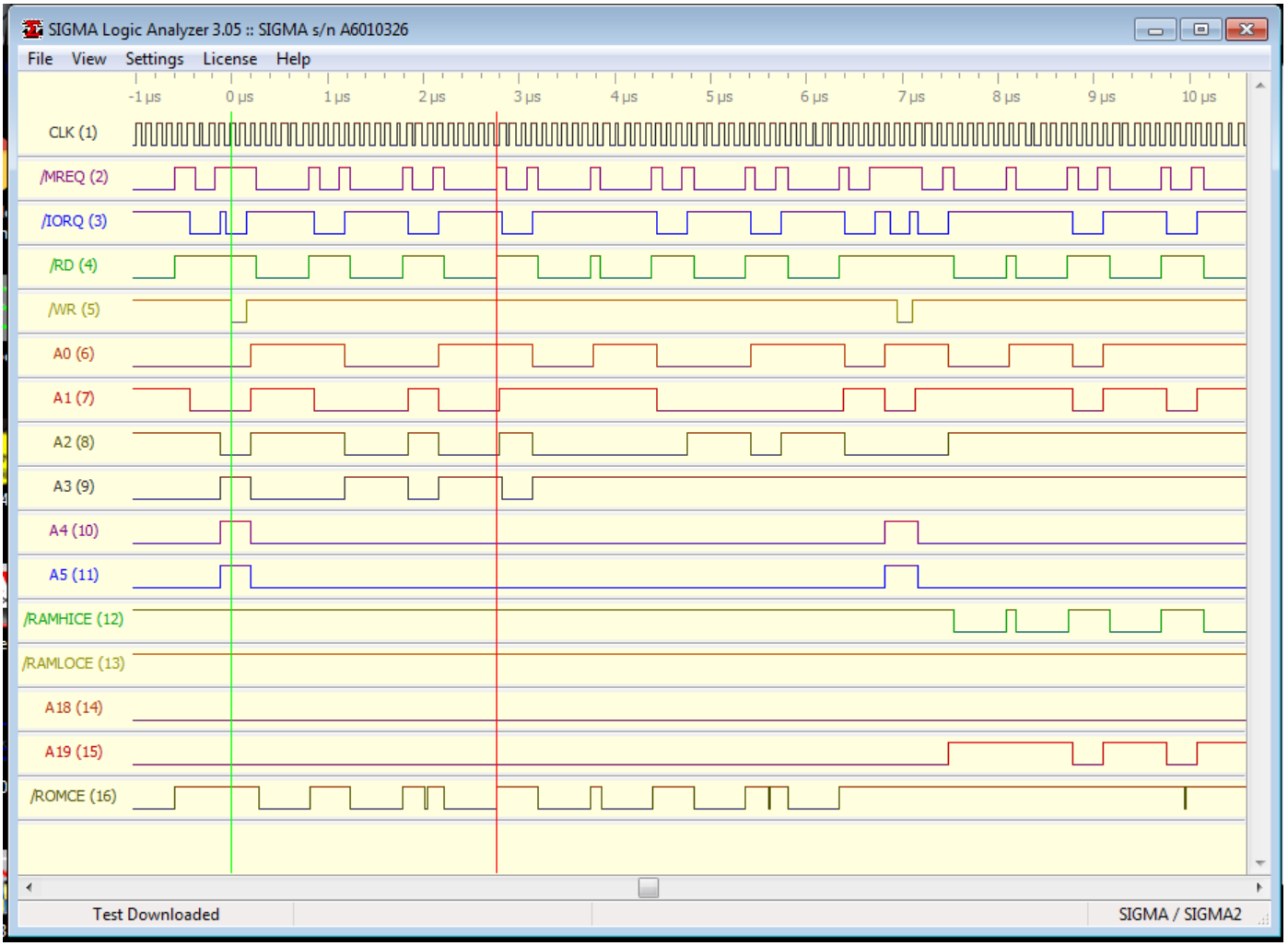
```
0000                    LEDOFF: equ 0x42    ;Write turns LED off
0000                    LEDON:  equ 0x43    ;Write turns LED on
0000
0000                    boot:
0000
0000                    ; Set up the MMU
0000                    ;
0000                    ; Common Bank 0
0000                    ;    logical: 0x0000 - 0x3fff
0000                    ;    physical: 0x00000 - 0x03fff
0000                    ; Bank Area
0000                    ;    logical: 0x4000 - 0xbfff
0000                    ;    physical: 0x74000 - 0x7bfff
0000                    ; Common Bank 1
0000                    ;    logical: 0xc000 - 0xffff
0000                    ;    physical: 0xfc000 - 0xfffff
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.

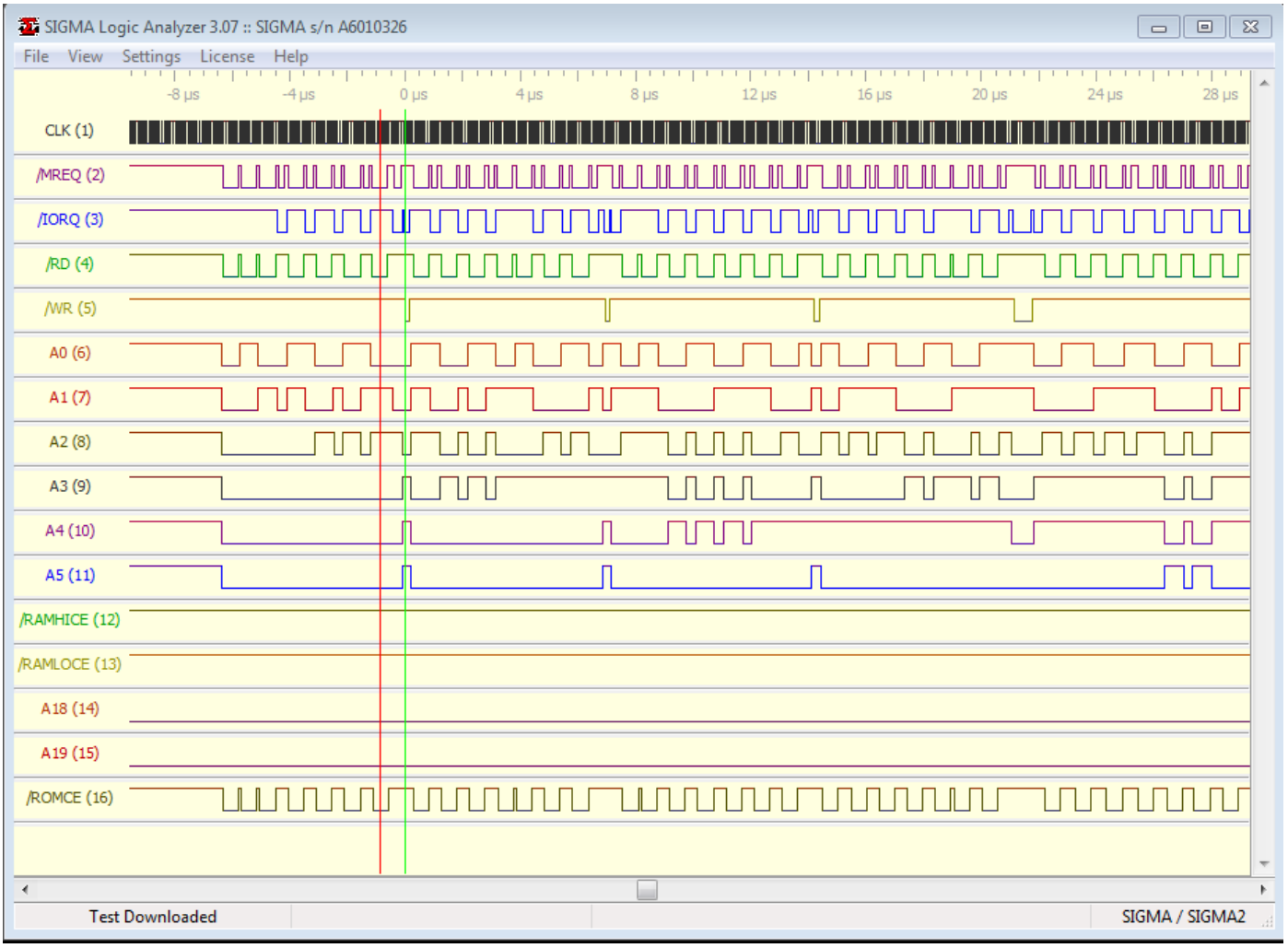
```
    ld a,0x80
    ld bc,BBR
    out (c),a
```

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:

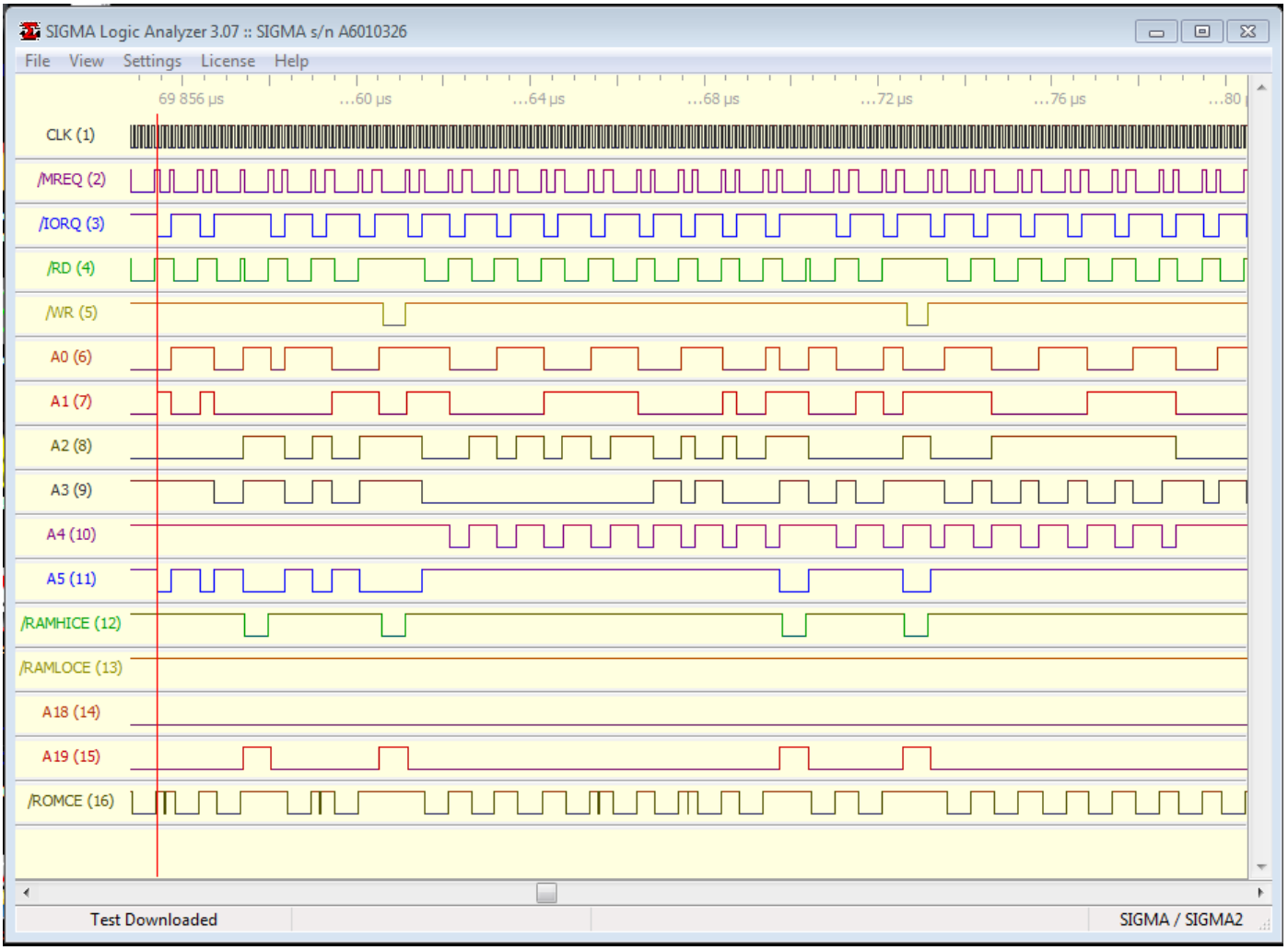


This program works that in addition reads and writes to the RAN:

```
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 - 0xefff
0000                    ;    physical: 0x00000 - 0x0efff
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                  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 11 00 f8               ld de,RAMADR
0018                    mloop:
0018 21 ff ff               ld hl,0xffff
001b 3e 01                  ld a,1
001d d3 43                  out(0x43),a ; LED on
001f                    onloop:
001f 1a                     ld a,(de)
0020 3c                     inc a
0021 12                     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
003a c3 18 00               jp mloop
003d
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
```

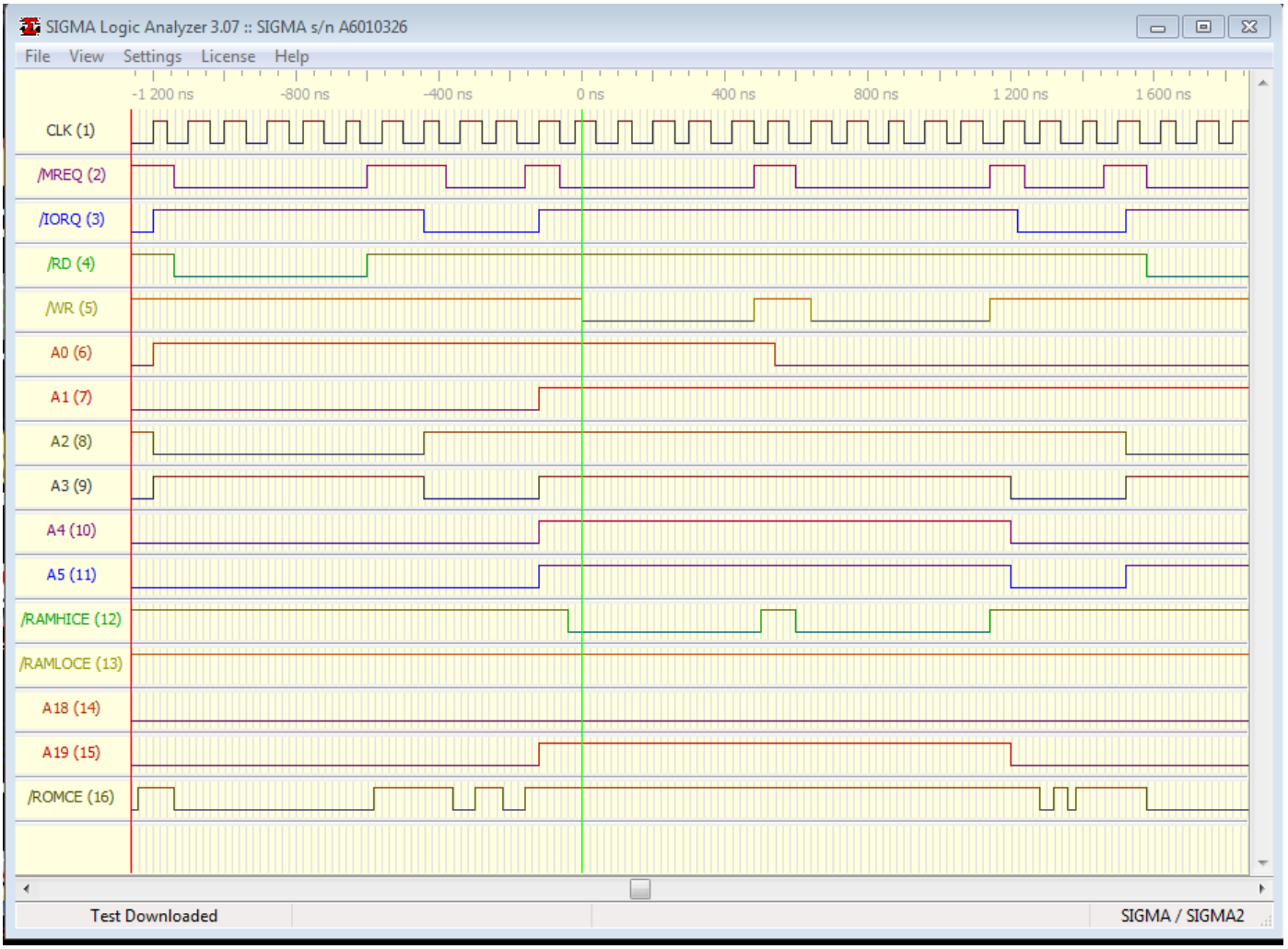
```
	date +'    db ", Built %F %R"' > tbuilt180.z80
	z80asm --list=test180.lst --output=test180.bin test180.z80

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: 0x00000 - 0x0efff
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                  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                    ; Set up Stack Pointer (first push/call will wrap to 0xffff)
0015 31 00 00               ld sp,0x0000
0018
0018                    ; Initialize other devices
0018                    ;   for now just blink LED
0018
0018 06 01                  ld b,1
001a cd 4d 00               call ledblink
001d 06 03                  ld b,3
001f cd 6a 00               call delays
0022
0022 06 02                  ld b,2
0024 cd 4d 00               call ledblink
0027 06 03                  ld b,3
0029 cd 6a 00               call delays
002c
002c 06 03                  ld b,3
002e cd 4d 00               call ledblink
0031 06 03                  ld b,3
0033 cd 6a 00               call delays
0036
0036 06 04                  ld b,4
0038 cd 4d 00               call ledblink
003b 06 03                  ld b,3
003d cd 6a 00               call delays
0040
0040                    testloop:
0040 06 0a                  ld b,10
0042 cd 4d 00               call ledblink
0045 06 03                  ld b,3
0047 cd 6a 00               call delays
004a c3 40 00               jp testloop
004d
004d                    ; ledblink flashes the LED
004d                    ; number of times in B reg
```

```
004d                    ; using reg: a, b, hl
004d                    ledblink:
004d 21 ff ff               ld hl,0xffff
0050 3e 01                  ld a,1
0052 d3 43                  out(LEDON),a
0054                    ledonloop:
0054 2b                     dec hl
0055 7c                     ld a,h
0056 b5                     or l
0057 c2 54 00               jp nz,ledonloop
005a 21 ff ff               ld hl,0xffff
005d 3e 00                  ld a,0
005f d3 42                  out(LEDOFF),a
0061                    ledoffloop:
0061 2b                     dec hl
0062 7c                     ld a,h
0063 b5                     or l
0064 c2 61 00               jp nz,ledoffloop
0067 10 e4                  djnz ledblink
0069 c9                     ret
006a
006a                    ; delays makes a delay multiplied
006a                    ; by number of times in B reg
006a                    ; using reg: a, b, hl
006a                    delays:
006a 21 ff ff               ld hl,0xffff
006d                    delay1loop:
006d 2b                     dec hl
006e 7c                     ld a,h
006f b5                     or l
0070 c2 6d 00               jp nz,delay1loop
0073 21 ff ff               ld hl,0xffff
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                    ; ASCI Registers port 0 and 1
0000                    CNTLA0: equ 0x0000  ;ASCI Channel Control Register A 0
0000                    CNTLA1: equ 0x0001  ;ASCI Channel Control Register A 1
0000                    CNTLB0: equ 0x0002  ;ASCI Control Register B 0
0000                    CNTLB1: equ 0x0003  ;ASCI Control Register B 1
0000                    STAT0:  equ 0x0004  ;ASCI Status Register 0
0000                    STAT1:  equ 0x0005  ;ASCI Status Register 1
0000                    TDR0:   equ 0x0006  ;ASCI Transmit Data Register 0
0000                    TDR1:   equ 0x0007  ;ASCI Transmit Data Register 1
0000                    RDR0:   equ 0x0008  ;ASCI Receive Data FIFO 0
0000                    RDR1:   equ 0x0009  ;ASCI 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
0000                    ;    logical: 0x0000 - 0xefff
0000                    ;    physical: 0x00000 - 0x0efff
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                  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                    ; Set up Stack Pointer (first push/call will wrap to 0xffff)
0015 31 00 00               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, ASCI 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                    ; ASCI 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                    ledonloop:
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                    ledoffloop:
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                    delay1loop:
008f 2b                     dec hl
0090 7c                     ld a, h
0091 b5                     or l
0092 c2 8f 00               jp nz, delay1loop
0095 21 ff ff               ld hl, 0xffff
0098                    delay2loop:
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
;   - interupt test

; Internal ports

; ASCI Registers port 0 and 1
CNTLA0: equ 0x0000  ;ASCI Channel Control Register A 0
CNTLA1: equ 0x0001  ;ASCI Channel Control Register A 1
CNTLB0: equ 0x0002  ;ASCI Control Register B 0
CNTLB1: equ 0x0003  ;ASCI Control Register B 1
STAT0:  equ 0x0004  ;ASCI Status Register 0
STAT1:  equ 0x0005  ;ASCI Status Register 1
TDR0:   equ 0x0006  ;ASCI Transmit Data Register 0
TDR1:   equ 0x0007  ;ASCI Transmit Data Register 1
RDR0:   equ 0x0008  ;ASCI Receive Data FIFO 0
RDR1:   equ 0x0009  ;ASCI 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 1: reset AVR when set to 1
                    ;       if JP8 pin 2-3 connected


boot:

init:

; Set up the MMU
;
; Common Bank 0
;    logical: 0x0000 - 0xefff
;    physical: 0x00000 - 0x0efff, 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, ASCI 0 initialized
    call ledblink
    ld b, 3
    call delays

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

    jp testloop

; ASCI 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
asci1init:
    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
asci1putc:
    ld bc, STAT1
    in a, (c)
    and 00000010b    ;test bit 1 = TDRE: Transmit Data Register Empty
    jr z, asci1putc  ;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
asci1getc:
    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
asci1pstr:
    ld a, (hl)
    or a
    ret z
    ld e, (hl)
    inc hl
    call asci1putc
    jr asci1pstr

; 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, asci1txt
    call asci1pstr
    ld hl, built
    call asci1pstr
    call asci1getc
    ld a, e
    or a
    jp z, asci1noin
    call asci1putc
    ld hl, inptxt
    call asci1pstr
asci1noin:

    jp testloop

asci0txt:
    db "ASCI port 0 - "
    db 0
asci1txt:
    db "ASCI port 1 - "
    db 0

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

inptxt:
    db " <- was recieved"
    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 - 0x01fff, EPROM  (or low RAM if enabled)
0000                    ; Bank Area
0000                    ;    logical: 0x2000 - 0xefff
0000                    ;    physical: 0x42000 - 0x4efff, low RAM above EPROM
```

```
0000                    ; Common Bank 1
0000                    ;    logical: 0xf000 - 0xffff
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
;   - interupt test

; Internal ports

; ASCI Registers port 0 and 1
CNTLA0: equ 0x0000  ;ASCI Channel Control Register A 0
CNTLA1: equ 0x0001  ;ASCI Channel Control Register A 1
CNTLB0: equ 0x0002  ;ASCI Control Register B 0
CNTLB1: equ 0x0003  ;ASCI Control Register B 1
STAT0:  equ 0x0004  ;ASCI Status Register 0
STAT1:  equ 0x0005  ;ASCI Status Register 1
TDR0:   equ 0x0006  ;ASCI Transmit Data Register 0
TDR1:   equ 0x0007  ;ASCI Transmit Data Register 1
RDR0:   equ 0x0008  ;ASCI Receive Data FIFO 0
RDR1:   equ 0x0009  ;ASCI 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 1: 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 - 0x01fff, 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, ASCI 0 initialized
    call ledblink
    ld b, 3
    call delays

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

    ld a, 0
    ld (0x2021), a

    jp testloop

; ASCI 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
asci1init:
    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
asci1putc:
    ld bc, STAT1
    in a, (c)
    and 00000010b    ;test bit 1 = TDRE: Transmit Data Register Empty
    jr z, asci1putc  ;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
asci1getc:
    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
asci1pstr:
    ld a, (hl)
    or a
    ret z
    ld e, (hl)
    inc hl
    call asci1putc
    jr asci1pstr
```

```
; 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, asci1txt
    call asci1pstr
    ld hl, indicator
    ld b, 0
    ld a, (0x2021)
    ld c, a
    add hl, bc
    ld e, (hl)
    call asci1putc
    ld hl, built
    call asci1pstr
    call asci1getc
    ld a, e
    or a
    jp z, asci1noin
    call asci1putc
    ld hl, inptxt
    call asci1pstr
asci1noin:

;Indicator index for messages

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

```
    and 00000011b
    ld (0x2021), a

    jp testloop

asci0txt:
    db "ASCI port 0 "
    db 0
asci1txt:
    db "ASCI port 1 "
    db 0

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

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

inptxt:
    db " <- was recieved"
    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 - 0x0fff
;    physical: 0x00000 - 0x00fff, EPROM or start of low RAM if enabled
; Bank Area, 56KB
;    logical: 0x1000 - 0xefff
;    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                    ;   - interupt test
0000
0000                    ; Internal ports
0000
0000                    ; ASCI Registers port 0 and 1
0000                    CNTLA0: equ 0x0000  ;ASCI Channel Control Register A 0
0000                    CNTLA1: equ 0x0001  ;ASCI Channel Control Register A 1
0000                    CNTLB0: equ 0x0002  ;ASCI Control Register B 0
0000                    CNTLB1: equ 0x0003  ;ASCI Control Register B 1
0000                    STAT0:  equ 0x0004  ;ASCI Status Register 0
0000                    STAT1:  equ 0x0005  ;ASCI Status Register 1
0000                    TDR0:   equ 0x0006  ;ASCI Transmit Data Register 0
0000                    TDR1:   equ 0x0007  ;ASCI Transmit Data Register 1
0000                    RDR0:   equ 0x0008  ;ASCI Receive Data FIFO 0
0000                    RDR1:   equ 0x0009  ;ASCI 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: 0x00000 - 0x00fff, 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 - 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 c9 01               ld bc,prgend - prgstart
0018 21 2e 00               ld hl,prgineprom
001b 11 00 f0               ld de,HIRAM
001e                    cploop:
001e 78                     ld a,b
001f b1                     or c
0020 ca 2b 00               jp z,cpend
0023 7e                     ld a,(hl)
0024 23                     inc hl
0025 12                     ld (de),a
0026 13                     inc de
0027 0b                     dec bc
0028 c3 1e 00               jp cploop
002b                    cpend:
```

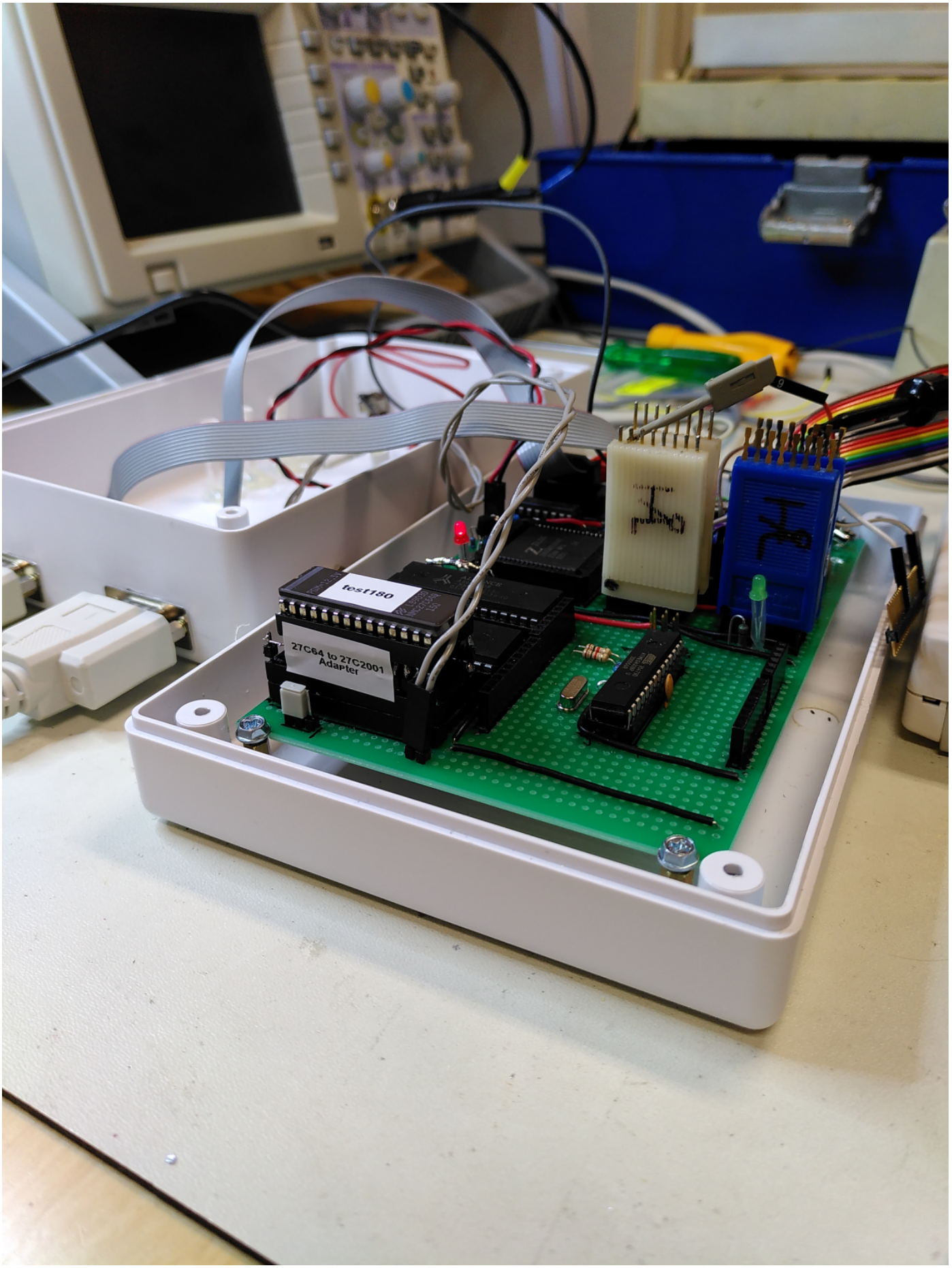
```
002b c3 00 f0               jp HIRAM    ; jump to the copied code
002e
002e                    prgineprom:
002e
002e                        org HIRAM
f000
f000                    prgstart:
f000
f000                    ; Set up Stack Pointer (first push/call will wrap to 0xffff)
f000 31 00 00               ld sp, 0x0000
f003
f003                    ; Initialize devices
f003                    ; and blink LED
f003
f003 06 01                  ld b, 1         ;1 blink, MMU initialized
f005 cd aa f0               call ledblink
f008 06 02                  ld b, 2
f00a cd c7 f0               call delays
f00d
f00d cd 34 f0               call asci0init
f010 06 02                  ld b, 2         ;2 blinks, ASCI 0 initialized
f012 cd aa f0               call ledblink
f015 06 02                  ld b, 2
f017 cd c7 f0               call delays
f01a
f01a cd 6f f0               call asci1init
f01d 06 03                  ld b, 3         ;3 blinks, ASCI 1 initialized
f01f cd aa f0               call ledblink
f022 06 02                  ld b, 2
f024 cd c7 f0               call delays
f027
f027 3e 00                  ld a, 0
f029 32 c9 f1               ld (indindex), a
f02c
f02c 3e 11                  ld a, 00010001b
f02e 32 ca f1               ld (cspattern), a
f031
f031 c3 dc f0               jp testloop
f034
f034                    ; ASCI 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               ld bc, CNTLA0
f039 ed 79                  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               ld bc, CNTLB0
f040 ed 79                  out (c), a
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               ld bc, STAT0
f046 ed 78                  in a, (c)
f048 e6 02                  and 00000010b    ;test bit 1 = TDRE: Transmit Data Register Empty
f04a 28 f7                  jr z, asci0putc  ;not empty yet
f04c 7b                     ld a, e
f04d 01 06 00               ld bc, TDR0
f050 ed 79                  out (c), a       ;output character
f052 c9                     ret
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                  ld e, 0
f055 01 04 00               ld bc, STAT0
f058 ed 78                  in a, (c)
f05a e6 80                  and 10000000b    ;test bit 7 = RDRF: Recieve data in FIFO
f05c c8                     ret z            ;empty
f05d 7b                     ld a, e
f05e 01 08 00               ld bc, RDR0
f061 ed 78                  in a, (c)        ;input character
f063 5f                     ld e, a
f064 c9                     ret
```

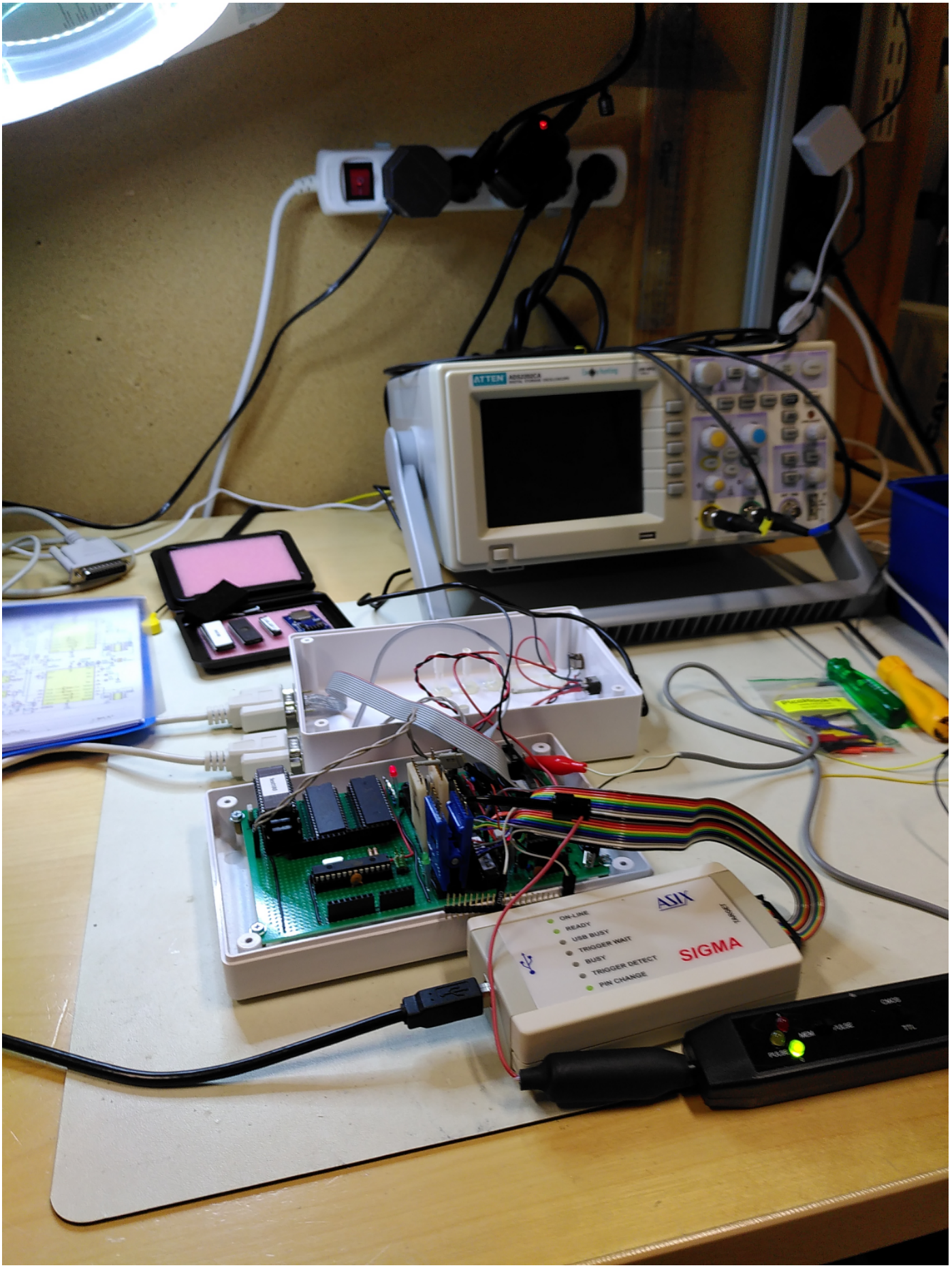
```
f065
f065                    ; Output a character string on port 0
f065                    ; reg HL points to string to output
f065                    ; the string is ended by 0
f065                    asci0pstr:
f065 7e                     ld a, (hl)
f066 b7                     or a
f067 c8                     ret z
f068 5e                     ld e, (hl)
f069 23                     inc hl
f06a cd 43 f0               call asci0putc
f06d 18 f6                  jr asci0pstr
f06f
f06f                    ; Initialize port 1
f06f                    asci1init:
f06f 3e 64                  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 01 01 00               ld bc, CNTLA1
f074 ed 79                  out (c), a
f076
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, CNTLB1
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                    asci1putc:
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, asci1putc  ;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                    asci1getc:
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                    asci1pstr:
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 asci1putc
f0a8 18 f6                  jr asci1pstr
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                    ledonloop:
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                    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
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
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
f0e6 3e 01                  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
f0f3 21 5e f1               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
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
f149                    ;Indicator index for messages
f149
f149 3a c9 f1               ld a, (indindex)
f14c 3c                     inc a
f14d e6 03                  and 00000011b
f14f 32 c9 f1               ld (indindex), a
f152
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
```

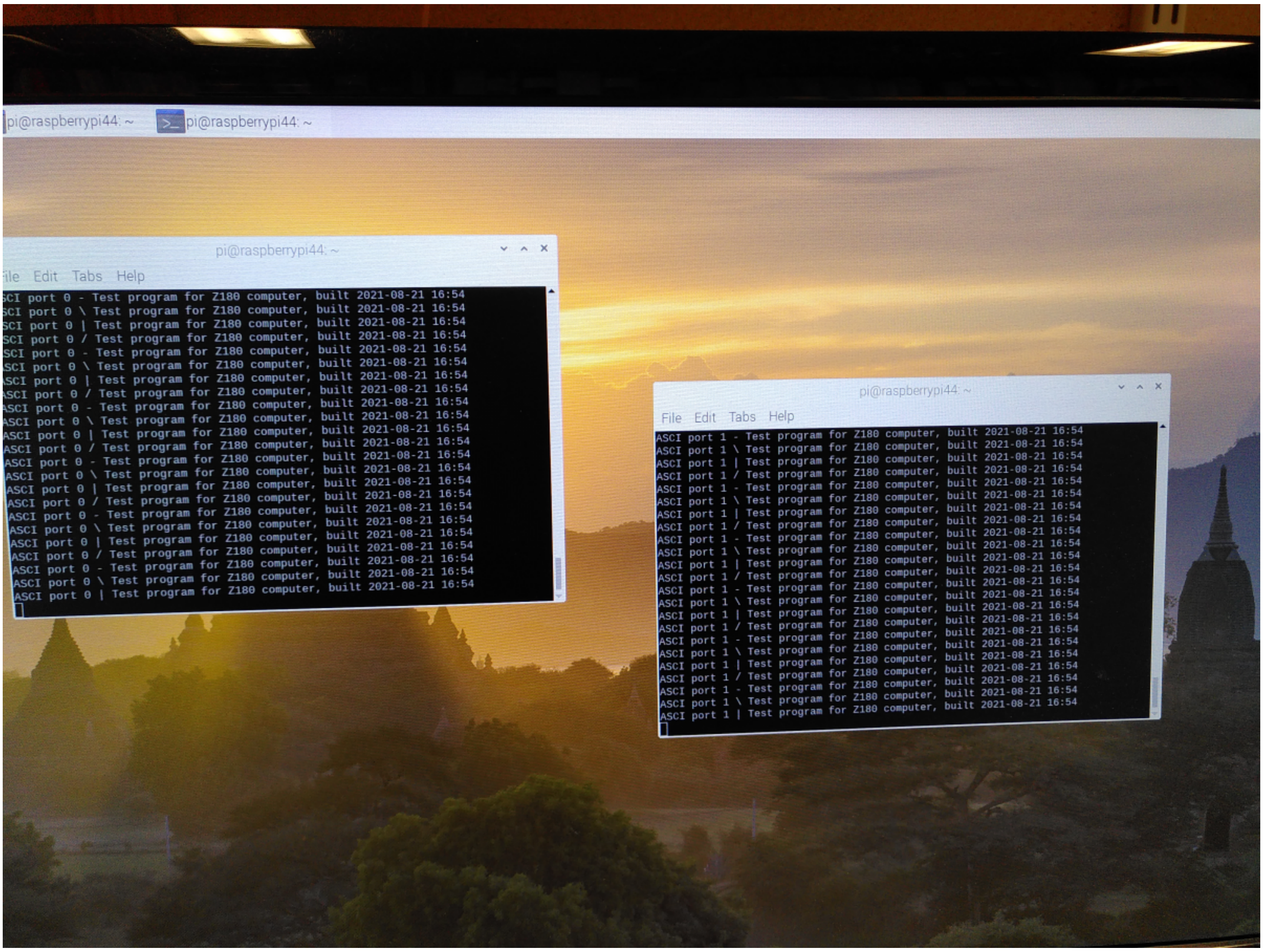
```
f15b
f15b c3 dc f0               jp testloop
f15e
f15e                    asci0txt:
f15e ..                     db "ASCI port 0 "
f16a 00                     db 0
f16b                    asci1txt:
f16b ..                     db "ASCI port 1 "
f177 00                     db 0
f178
f178                    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 recieved"
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:


test1180
27C64 to 27C2001
Adapter


ATTEN
ON-LINE
READY
USB BUSY
TRIGGER WAIT
BUSY
TRIGGER DETECT
PIN CHANGE
SIGMA

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                    ;   - interupt test
0000
0000                    ; Internal ports
0000
0000                    ; ASCI Registers port 0 and 1
0000                    CNTLA0: equ 0x0000  ;ASCI Channel Control Register A 0
0000                    CNTLA1: equ 0x0001  ;ASCI Channel Control Register A 1
0000                    CNTLB0: equ 0x0002  ;ASCI Control Register B 0
0000                    CNTLB1: equ 0x0003  ;ASCI Control Register B 1
0000                    STAT0:  equ 0x0004  ;ASCI Status Register 0
0000                    STAT1:  equ 0x0005  ;ASCI Status Register 1
0000                    TDR0:   equ 0x0006  ;ASCI Transmit Data Register 0
```

```
0000                    TDR1:   equ 0x0007  ;ASCI Transmit Data Register 1
0000                    RDR0:   equ 0x0008  ;ASCI Receive Data FIFO 0
0000                    RDR1:   equ 0x0009  ;ASCI 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 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 - 0xefff
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                    cploop:
001e 78                     ld a,b
001f b1                     or c
0020 ca 2b 00               jp z,cpend
0023 7e                     ld a,(hl)
0024 23                     inc hl
0025 12                     ld (de),a
0026 13                     inc de
0027 0b                     dec bc
0028 c3 1e 00               jp cploop
002b                    cpend:
002b c3 00 f0               jp HIRAM    ; jump to the copied code
002e
002e                    prgineprom:
002e
002e                        org HIRAM
f000
f000                    prgstart:
f000
f000                    ; Set up Stack Pointer (first push/call will wrap to 0xffff)
f000 31 00 00               ld sp, 0x0000
f003
f003                    ; Initialize devices
f003                    ; and blink LED
f003
f003 06 01                  ld b, 1         ;1 blink, MMU initialized
f005 cd aa f0               call ledblink
f008 06 02                  ld b, 2
f00a cd c7 f0               call delays
```

```
f00d
f00d cd 34 f0               call asci0init
f010 06 02                  ld b, 2         ;2 blinks, ASCI 0 initialized
f012 cd aa f0               call ledblink
f015 06 02                  ld b, 2
f017 cd c7 f0               call delays
f01a
f01a cd 6f f0               call asci1init
f01d 06 03                  ld b, 3         ;3 blinks, ASCI 1 initialized
f01f cd aa f0               call ledblink
f022 06 02                  ld b, 2
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                    ; ASCI 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               ld bc, CNTLA0
f039 ed 79                  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               ld bc, CNTLB0
f040 ed 79                  out (c), a
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               ld bc, STAT0
f046 ed 78                  in a, (c)
f048 e6 02                  and 00000010b    ;test bit 1 = TDRE: Transmit Data Register Empty
f04a 28 f7                  jr z, asci0putc  ;not empty yet
f04c 7b                     ld a, e
f04d 01 06 00               ld bc, TDR0
f050 ed 79                  out (c), a       ;output character
f052 c9                     ret
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                  ld e, 0
f055 01 04 00               ld bc, STAT0
f058 ed 78                  in a, (c)
f05a e6 80                  and 10000000b    ;test bit 7 = RDRF: Recieve data in FIFO
f05c c8                     ret z            ;empty
f05d 7b                     ld a, e
f05e 01 08 00               ld bc, RDR0
f061 ed 78                  in a, (c)        ;input character
f063 5f                     ld e, a
f064 c9                     ret
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                     ld a, (hl)
f066 b7                     or a
f067 c8                     ret z
f068 5e                     ld e, (hl)
f069 23                     inc hl
f06a cd 43 f0               call asci0putc
f06d 18 f6                  jr asci0pstr
f06f
f06f                    ; Initialize port 1
f06f                    asci1init:
f06f 3e 64                  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 01 01 00               ld bc, CNTLA1
f074 ed 79                  out (c), a
f076
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, CNTLB1
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                    asci1putc:
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, asci1putc  ;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                    asci1getc:
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                    asci1pstr:
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 asci1putc
f0a8 18 f6                  jr asci1pstr
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                    ledonloop:
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                    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
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
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                    ramtstlop:
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, ramtstlop
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                    asci0noin:
f13a
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                    asci1noin:
f165
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 asci1pstr
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, ramtxterr
f19d cd 65 f0               call asci0pstr
f1a0 21 a7 f2               ld hl, ramtxterr
f1a3 cd a0 f0               call asci1pstr
f1a6 18 0c                  jr ramt0end
f1a8                    ramt0ok:
f1a8 21 9f f2               ld hl, ramtxtok
f1ab cd 65 f0               call asci0pstr
f1ae 21 9f f2               ld hl, ramtxtok
f1b1 cd a0 f0               call asci1pstr
f1b4                    ramt0end:
f1b4
f1b4                    ; Test physical RAM 0x0f000 - 0x1cfff
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 asci1pstr
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, ramtxterr
f1d6 cd 65 f0               call asci0pstr
f1d9 21 a7 f2               ld hl, ramtxterr
f1dc cd a0 f0               call asci1pstr
f1df 18 0c                  jr ramt1end
f1e1                    ramt1ok:
f1e1 21 9f f2               ld hl, ramtxtok
f1e4 cd 65 f0               call asci0pstr
f1e7 21 9f f2               ld hl, ramtxtok
f1ea cd a0 f0               call asci1pstr
f1ed                    ramt1end:
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 asci1pstr
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, ramtxterr
f20f cd 65 f0               call asci0pstr
f212 21 a7 f2               ld hl, ramtxterr
f215 cd a0 f0               call asci1pstr
f218 18 0c                  jr ramt9end
f21a                    ramt9ok:
f21a 21 9f f2               ld hl, ramtxtok
f21d cd 65 f0               call asci0pstr
f220 21 9f f2               ld hl, ramtxtok
f223 cd a0 f0               call asci1pstr
f226                    ramt9end:
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                    asci1txt:
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 recieved"
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 - 0x1cfff"
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 - 0xcdfff"
f4cd 00                     db 0
f4ce
f4ce                    ramtxt15:
f4ce ..                     db "Test physical RAM 0xce000 - 0xdbfff"
f4f1 00                     db 0
f4f2
f4f2                    ramtxt16:
f4f2 ..                     db "Test physical RAM 0xdc000 - 0xe9fff"
f515 00                     db 0
f516
f516                    ramtxt17:
f516 ..                     db "Test physical RAM 0xea000 - 0xf7fff"
f539 00                     db 0
f53a
f53a                    ramtxt18:
f53a ..                     db "Test physical RAM 0xf8000 - 0xfefff"
f55d 00                     db 0
f55e
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                    ; 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                    ;   - interupt test
0000
0000                    ; Internal ports
0000
0000                    ; ASCI Registers port 0 and 1
0000                    CNTLA0: equ 0x0000  ;ASCI Channel Control Register A 0
0000                    CNTLA1: equ 0x0001  ;ASCI Channel Control Register A 1
0000                    CNTLB0: equ 0x0002  ;ASCI Control Register B 0
0000                    CNTLB1: equ 0x0003  ;ASCI Control Register B 1
0000                    STAT0:  equ 0x0004  ;ASCI Status Register 0
0000                    STAT1:  equ 0x0005  ;ASCI Status Register 1
0000                    TDR0:   equ 0x0006  ;ASCI Transmit Data Register 0
0000                    TDR1:   equ 0x0007  ;ASCI Transmit Data Register 1
0000                    RDR0:   equ 0x0008  ;ASCI Receive Data FIFO 0
0000                    RDR1:   equ 0x0009  ;ASCI Receive Data FIFO 1
0000
0000                    ; DMA/WAIT Control Register
0000                    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 - 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 - 0x00fff, 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 - 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 98 05               ld bc,prgend - prgstart
0018 21 2e 00               ld hl,prgineprom
001b 11 00 f0               ld de,HIRAM
001e                    cploop:
001e 78                     ld a,b
001f b1                     or c
0020 ca 2b 00               jp z,cpend
0023 7e                     ld a,(hl)
0024 23                     inc hl
0025 12                     ld (de),a
0026 13                     inc de
0027 0b                     dec bc
0028 c3 1e 00               jp cploop
002b                    cpend:
002b c3 00 f0               jp HIRAM    ; jump to the copied code
002e
002e                    prgineprom:
002e
002e                        org HIRAM
f000
f000                    prgstart:
f000
f000                    ; Set up Stack Pointer (first push/call will wrap to 0xffff)
f000 31 00 00               ld sp, 0x0000
f003
f003                    ; When running from RAM configure no memory wait states
f003 01 32 00               ld bc, DCNTL
f006 ed 78                  in a, (c)
f008 e6 3f                  and 00111111b  ; reset MVI0 and MVI1
f00a ed 79                  out (c), a
f00c
f00c                    ; Initialize devices
f00c                    ; and blink LED
f00c
f00c 06 01                  ld b, 1         ;1 blink, MMU initialized
f00e cd b3 f0               call ledblink
f011 06 02                  ld b, 2
```

```
f013 cd d0 f0               call delays
f016
f016 cd 3d f0               call asci0init
f019 06 02                  ld b, 2         ;2 blinks, ASCI 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 asci1init
f026 06 03                  ld b, 3         ;3 blinks, ASCI 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                    ; ASCI 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                     ret
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                     ld a, e
f056 01 06 00               ld bc, TDR0
f059 ed 79                  out (c), a       ;output character
f05b c9                     ret
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                     ret z            ;empty
f066 7b                     ld a, e
f067 01 08 00               ld bc, RDR0
f06a ed 78                  in a, (c)        ;input character
f06c 5f                     ld e, a
f06d c9                     ret
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                     ld e, (hl)
f072 23                     inc hl
f073 cd 4c f0               call asci0putc
f076 18 f6                  jr asci0pstr
f078
f078                    ; Initialize port 1
f078                    asci1init:
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, CNTLA1
f07d ed 79                  out (c), a
f07f
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                    asci1putc:
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, asci1putc  ;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                    asci1getc:
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                    asci1pstr:
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 asci1putc
f0b1 18 f6                  jr asci1pstr
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                    delay1loop:
f0d3 2b                     dec hl
f0d4 7c                     ld a, h
f0d5 b5                     or l
```

```
f0d6 c2 d3 f0               jp nz, delay1loop
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                    ramtstlop:
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                    ramtstff:
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                    ramtstnxt:
f104 23                     inc hl
f105 0b                     dec bc
f106 78                     ld a,b
f107 b1                     or c
f108 20 e0                  jr nz, ramtstlop
f10a 3a 9a f5               ld a, (ramerr)
f10d c9                     ret
f10e
f10e                    ; Main test loop
f10e                    testloop:
f10e 06 01                  ld b, 1
f110 cd b3 f0               call ledblink
f113 06 01                  ld b, 1
f115 cd d0 f0               call delays
f118
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                    asci0noin:
f143
f143 21 f4 f1               ld hl, asci1txt
f146 cd a9 f0               call asci1pstr
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 asci1putc
f157 21 05 f2               ld hl, built
f15a cd a9 f0               call asci1pstr
f15d cd 97 f0               call asci1getc
f160 7b                     ld a, e
f161 b7                     or a
f162 ca 6e f1               jp z, asci1noin
f165 cd 87 f0               call asci1putc
f168 21 3f f2               ld hl, inptxt
f16b cd a9 f0               call asci1pstr
f16e                    asci1noin:
f16e
f16e                    ;Indicator index for messages
f16e 3a 98 f5               ld a, (indindex)
f171 3c                     inc a
f172 e6 03                  and 00000011b
```

```
f174 32 98 f5               ld (indindex), a
f177
f177                    ;Chip Select and reset outputs
f177 3a 99 f5               ld a, (cspattern)
f17a d3 44                  out (CSPORT), a
f17c 07                     rlca
f17d 32 99 f5               ld (cspattern), a
f180
f180                    ; Test RAM
f180
f180                    ; Enable RAM in low memory
f180 3e 01                  ld a, 1
f182 d3 41                  out (RAMSEL), a   ; select RAM instead of EPROM from address 0
f184
f184                    ; Test physical RAM
f184 dd 21 52 f2            ld ix, ramtsttab
f188                    ramtloop:
f188 dd 6e 00               ld l, (ix + 0)
f18b dd 66 01               ld h, (ix + 1)
f18e 7c                     ld a, h
f18f b5                     or l
f190 28 47                  jr z, ramtend   ; end of test table
f192 dd 7e 02               ld a, (ix + 2)
f195 01 39 00               ld bc, BBR
f198 ed 79                  out (c), a
f19a
f19a cd 6e f0               call asci0pstr
f19d dd 6e 00               ld l, (ix + 0)
f1a0 dd 66 01               ld h, (ix + 1)
f1a3 cd a9 f0               call asci1pstr
f1a6 dd 6e 03               ld l, (ix + 3)
f1a9 dd 66 04               ld h, (ix + 4)
f1ac dd 4e 05               ld c, (ix + 5)
f1af dd 46 06               ld b, (ix + 6)
f1b2 cd e5 f0               call ramtest
f1b5 b7                     or a
f1b6 28 0e                  jr z, ramisok
f1b8 21 e1 f2               ld hl, ramtxterr
f1bb cd 6e f0               call asci0pstr
f1be 21 e1 f2               ld hl, ramtxterr
f1c1 cd a9 f0               call asci1pstr
f1c4 18 0c                  jr ramtnxt
f1c6                    ramisok:
f1c6 21 d9 f2               ld hl, ramtxtok
f1c9 cd 6e f0               call asci0pstr
f1cc 21 d9 f2               ld hl, ramtxtok
f1cf cd a9 f0               call asci1pstr
f1d2                    ramtnxt:
f1d2 01 07 00               ld bc, 7
f1d5 dd 09                  add ix, bc
f1d7 18 af                  jr ramtloop
f1d9                    ramtend:
f1d9
f1d9                    ; Enable EPROM in low memory
f1d9 3e 00                  ld a, 0
f1db d3 40                  out (ROMSEL), a
f1dd
f1dd                    ; Restore Bank Area
f1dd 3e 40                  ld a, 0x40
f1df 01 39 00               ld bc, BBR
f1e2 ed 79                  out (c), a
f1e4
f1e4 c3 0e f1               jp testloop
f1e7
f1e7                    asci0txt:
f1e7 ..                     db "ASCI port 0 "
f1f3 00                     db 0
f1f4                    asci1txt:
f1f4 ..                     db "ASCI port 1 "
f200 00                     db 0
f201
f201                    indicator:
f201 .. 2f 2d 5c            db '|', '/', '-', '\\'
f205
f205                    built:
f205 ..                     db " Test program for Z180 computer"
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                    inptxt:
f23f ..                     db " <- was recieved"
f24f .. 0a                  db '\r', '\n'
f251 00                     db 0
f252
f252                    ; RAM test table
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
f259 10 f3                  dw ramtxt1  ;test message
f25b 0e                     db 0x0e     ;BBR
f25c 00 10                  dw 0x1000   ;logical test start address
f25e 00 e0                  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
f2e1                    ramtxterr:
f2e1 ..                     db " - ERROR"
f2e9 .. 0a                  db '\r', '\n'
f2eb 00                     db 0
f2ec
f2ec                    ramtxt0:
f2ec ..                     db "Test physical RAM 0x00000 - 0x0efff"
f30f 00                     db 0
f310
f310                    ramtxt1:
f310 ..                     db "Test physical RAM 0x0f000 - 0x1cfff"
f333 00                     db 0
f334
f334                    ramtxt2:
f334 ..                     db "Test physical RAM 0x1d000 - 0x2afff"
f357 00                     db 0
f358
f358                    ramtxt3:
f358 ..                     db "Test physical RAM 0x2b000 - 0x38fff"
f37b 00                     db 0
f37c
f37c                    ramtxt4:
f37c ..                     db "Test physical RAM 0x39000 - 0x46fff"
f39f 00                     db 0
f3a0
f3a0                    ramtxt5:
f3a0 ..                     db "Test physical RAM 0x47000 - 0x54fff"
f3c3 00                     db 0
f3c4
f3c4                    ramtxt6:
f3c4 ..                     db "Test physical RAM 0x55000 - 0x62fff"
f3e7 00                     db 0
f3e8
f3e8                    ramtxt7:
f3e8 ..                     db "Test physical RAM 0x63000 - 0x70fff"
f40b 00                     db 0
f40c
f40c                    ramtxt8:
f40c ..                     db "Test physical RAM 0x71000 - 0x7efff"
f42f 00                     db 0
f430
f430                    ramtxt9:
f430 ..                     db "Test physical RAM 0x7f000 - 0x8cfff"
f453 00                     db 0
f454
f454                    ramtxt10:
f454 ..                     db "Test physical RAM 0x8d000 - 0x9afff"
f477 00                     db 0
f478
f478                    ramtxt11:
f478 ..                     db "Test physical RAM 0x9b000 - 0xa8fff"
f49b 00                     db 0
f49c
f49c                    ramtxt12:
f49c ..                     db "Test physical RAM 0xa9000 - 0xb6fff"
f4bf 00                     db 0
f4c0
f4c0                    ramtxt13:
f4c0 ..                     db "Test physical RAM 0xb7000 - 0xc4fff"
f4e3 00                     db 0
f4e4
f4e4                    ramtxt14:
f4e4 ..                     db "Test physical RAM 0xc5000 - 0xd2fff"
f507 00                     db 0
f508
f508                    ramtxt15:
f508 ..                     db "Test physical RAM 0xd3000 - 0xe0fff"
f52b 00                     db 0
f52c
f52c                    ramtxt16:
f52c ..                     db "Test physical RAM 0xe1000 - 0xeefff"
f54f 00                     db 0
f550
f550                    ramtxt17:
f550 ..                     db "Test physical RAM 0xef000 - 0xfcfff"
f573 00                     db 0
f574
f574                    ramtxt18:
f574 ..                     db "Test physical RAM 0xfd000 - 0xfefff"
f597 00                     db 0
f598
f598                    ; The rest of RAM is for variables
f598                    prgend:
f598
f598                    indindex:
f598 00                     db 0
f599
f599                    cspattern:
```

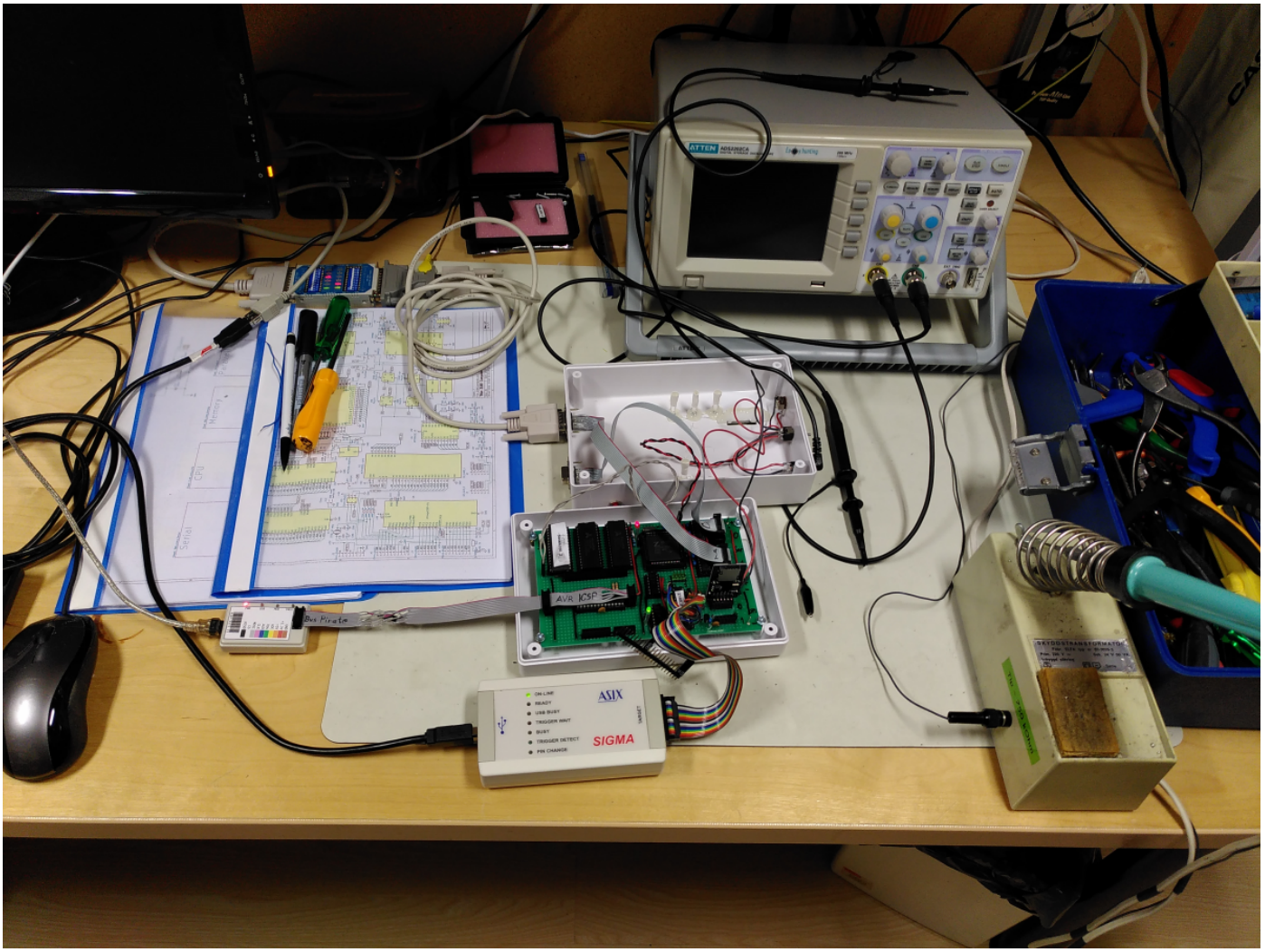
```
f599 00                     db 0
f59a
f59a                    ramerr:
f59a 00                     db 0
f59b
# End of file test180.z80
f59b
```

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 Arduini 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;
}
```