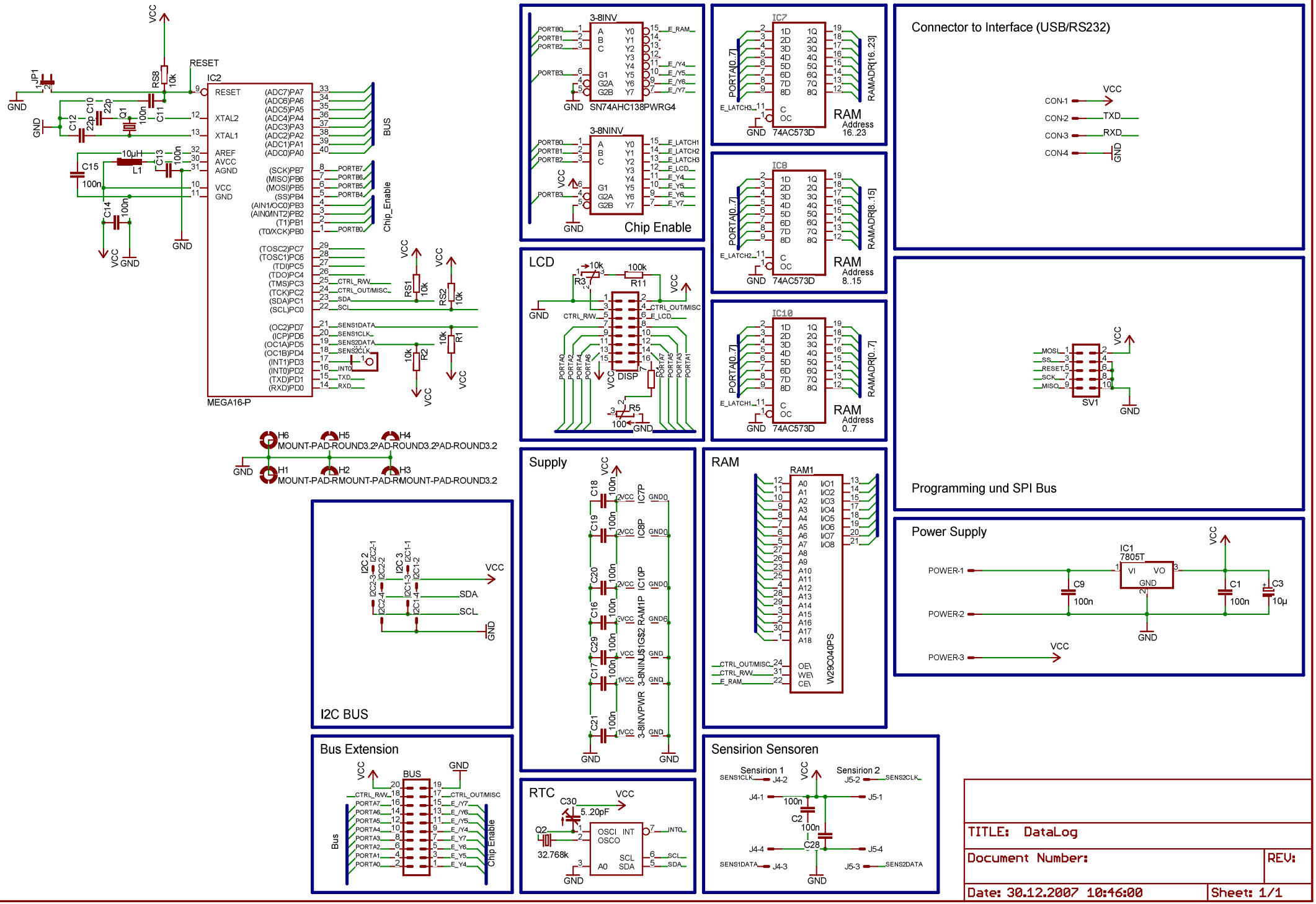

IC2
MEGA16-P
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
BUS
Chip_Enable
3-8INV
SN74AHC138PWRG4
Chip Enable
3-8NINV
IC7
IC8
IC10
RAM
Address 16..23
Address 8..15
Address 0..7
74AC573D
Connector to Interface (USB/RS232)
CON-1
CON-2
CON-3
CON-4
TXD
RXD
LCD
DISP
SV1
Programming und SPI Bus
Power Supply
IC1
7805T
POWER-1
POWER-2
POWER-3
Supply
RAM1
W29C040PS
I2C BUS
SDA
SCL
Bus Extension
RTC
32.768k
Sensirion Sensoren
Sensirion 1
Sensirion 2
SENS1CLK
SENS2CLK
SENS1DATA
SENS2DATA
TITLE: DataLog
Document Number:
REV:
Date: 30.12.2007 10:46:00
Sheet: 1/1

```
;---------------------------------------------------------------
;    Name : DataloggerMain.ASM
;Funktion : Hauptprogramm für den Datalogger
;
;
; Version 1.2  by Jan-Erik Schmutz
;
;
;
; letzte Änderung   17.03.2008
;---------------------------------------------------------------




; ++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++ Definitionen ++++++++++++++++++++++
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++

;#define DEBUG
.DEF    temp                = r16    ;temporary register
.DEF    temp_1              = r17
.def    SRAM_Addr           = r21
.def    SRAM_Page_Addr_L    = r24
.def    SRAM_Page_Addr_H    = r25
.def    SRAM_Data           = r23


; ++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++ Interrupt Vektor Tabelle ++++++++++++++++
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++

.CSEG
.org 0x000
        jmp RESET_MEGA16    ; Reset bzw. Einschalten der Stromversorgung
.org 0x002
        jmp EXTINT0         ; Externer Interrupt 0
.org 0x004
        reti                ; Externer Interrupt 1
.org 0x006
        reti                ; Timer/Counter 2 Compare Match
.org 0x008
        reti                ; Timer/Counter 2 Overflow
.org 0x00A
        reti                ; Timer/Counter 1 Capture Event
.org 0x00C
        reti                ; Timer/Counter 1 Compare Match A
.org 0x00E
        reti                ; Timer/Counter 1 Compare Match B
.org 0x010
        reti                ; Timer/Counter 1 Overflow
.org 0x012
        reti                ; Timer/Counter 0 Overflow
.org 0x014
        reti                ; SPI Übertragung abgeschlossen
.org 0x016
        jmp USART_RECEIVE   ; USART Empfang abgeschlossen
.org 0x018
        reti                ; USART Datenregister leer
.org 0x01A
        reti;rjmp USART_SEND; USART Sendung abgeschlossen
.org 0x01C
        reti                ; AD Wandlung abgeschlossen
.org 0x01E
        reti                ; EEPROM bereit
.org 0x020
        reti                ; Analogkomparator
.org 0x022
        reti;rjmp TWI_CONTROL   ; Two-Wire Interface
.org 0x024
        reti                ; Store Program Memory Ready
```

```
.org 0x026
        reti
.org 0x028
        reti

; +++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++++ Allgem. MACROS ++++++++++++++++++
; +++++++++++++++++++++++++++++++++++++++++++++++++++++
; M_Chip_Enable @0      ; @0:Enable_Address
; M_Chip_Disable @0
; M_Bus_Input
; M_Bus_Output
; +++++++++++++++++++++++++++++++++++++++++++++++++++++


.MACRO M_Chip_Enable                ; z.B. Chip_Enable LCD_enable
        push    r16

        in      r16, CHIPENABLE_IN  ; Portstatus einlesen (PIN)
        andi    r16, 0b11110000 ; unteres Nibble loeschen
        ori     r16, @0         ; Chip enable Adresse
        out     CHIPENABLE, r16 ; Port setzen

        pop r16
.ENDMACRO

.MACRO M_Chip_Disable               ; z.B. Chip_Disable LCD_enable
        push    r16

        in      r16, CHIPENABLE_IN  ; Portstatus einlesen (PIN)
        andi    r16, 0b11110000 ; unteres Nibble loeschen
        ori     r16, 0b00001001 ;
        out     CHIPENABLE, r16 ; Port setzen

        pop r16
.ENDMACRO


.MACRO M_Bus_Input
        push r16
        ldi r16, 0b00000000
        out BUS_DDR, r16    ; Bus ist INPUT
        out BUS, r16        ; Tristate input
        pop r16
.ENDMACRO


.MACRO M_Bus_Output
        push r16
        ldi r16, 0b11111111
        out BUS_DDR, r16    ; Bus ist OUTPUT
        pop r16
.ENDMACRO


; +++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++ INCLUDE FILES +++++++++++++++++++++
; +++++++++++++++++++++++++++++++++++++++++++++++++++++


.include "m16def.inc"       ; AT Mega16 Definitionsdatei
.include "EQUs.asm"         ; Equ Anweisungen
.include "Init_Variables.asm"
.include "Init_AVR.asm"     ; Microcontroller initialisieren
.include "INT_Save.asm"     ; Routine um alle Register bei Interrupt zu sichern
.include "Delays.asm"       ;
.include "Bin2BCD.asm"      ; Umwandlung von BIN nach BCD
.include "LCD.asm"          ; Display 4x20
.include "SRAM.asm"         ; 512kx8 CMOS Flash Memory W29C040
.include "Sensirion.asm"    ; Luftfeuchte-Temperaturmesser SHT75 (Sensirion)
.include "USART.asm"        ; Schnittstelle
.include "I2C.asm"          ; I²C Ansteuerung
.include "ADC_MAX127.asm"   ; Analog-Digital-Wandler
```

```
.include "RTC_PCF8583.asm"  ; Realtime Clock


; +++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++ Initialisierungen ++++++++++++++++
; +++++++++++++++++++++++++++++++++++++++++++++++++++++


RESET_MEGA16:
        M_INIT_AVR              ; AVR grundlegenden Einstellungen vornehmen
        ;M_INIT_SHT 0           ; Sensor 1 als Default
        rcall INIT_LCD          ; LCD initialisieren
        ;rcall SRAM_ReadProductID
        rcall SRAM_Unprotect
        ;rcall SRAM_Erase       ; SRAM komplett loeschen
        call Clear_iSRAM_Variables
    ;   rcall INIT_SRAM

        ldi r16,0
        sts Active_SHT, r16
        sts Active_ADC,r16
        ;sts SRAM_Store_Count, r16
        ;ldi r16,0xDF
        ;sts cur_iSRAM_Page_Addr, r16
        RCALL lcd_CLEAR

        ;ldi r16, 1
        ;sts SRAM_Store_Time, r16
        call Fill_iSRAM_Variables

    ;   sei

; +++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++ Hauptprogramm +++++++++++++++++++++
; +++++++++++++++++++++++++++++++++++++++++++++++++++++


main:

    ;rcall SRAM_Unprotect

    ldi SRAM_Page_Addr_L, 0b00001111
    ldi SRAM_Page_Addr_H, 0b00000000
    ldi SRAM_Addr, 0b01010101
    ldi SRAM_Data, 0b11001100
    rcall SRAM_Write_Page

    ldi SRAM_Page_Addr_L, 0
    ldi SRAM_Page_Addr_H, 0
    ldi SRAM_Addr, 0
    rcall SRAM_Read_Byte


    ldi SRAM_Page_Addr_L, 0b00001111
    ldi SRAM_Page_Addr_H, 0b00000000
    ldi SRAM_Addr, 0b01010101
    rcall SRAM_Read_Byte

rjmp main
```

```
.CSEG


SRAM_Load_Page_Addr:            ; Page Adresse laden (A08 bis A18), Latch2 und Latch3

            push SRAM_Page_Addr_L           ; Low Address -> Latch2
            push SRAM_Page_Addr_H           ; High Address -> Latch3

            M_Bus_Output                    ; BUS als Ausgang
            out BUS, SRAM_Page_Addr_L       ; A08 bis A15 auf den BUS
            M_Chip_enable Latch15_enable
            nop                             ; und ins Latch2 uebernehmen
            M_Chip_disable Latch15_enable

            out BUS, SRAM_Page_Addr_H       ; A16 bis A23 auf den BUS
            M_Chip_enable Latch23_enable
            nop                             ; und ins Latch3 uebernehmen
            M_Chip_disable Latch23_enable

            pop SRAM_Page_Addr_H
            pop SRAM_Page_Addr_L
            ret

SRAM_Load_Addr:                 ; Adresse innerhalb einer Page laden (A01 bis A07), Lat
ch1

            push SRAM_Addr
            M_BUS_OUTPUT                    ; BUS als Ausgang
            out BUS, SRAM_Addr              ; Adresse auf den BUS
            M_Chip_enable Latch07_enable    ;
            nop                             ; und ins Latch ubernehmen
            M_Chip_disable Latch07_enable;
            pop SRAM_Addr
            ret

SRAM_Read_Byte:
            M_Chip_disable SRAM_enable      ; SRAM /CE auf HIGH

            rcall SRAM_Load_Page_Addr       ; Adresse ins Latch laden
            rcall SRAM_Load_Addr            ;
            M_BUS_OUTPUT                    ; BUS als Ausgang

            sbi CONTROL, CTRL_RW            ; /WE auf HIGH
            sbi CONTROL, CTRL_OUT           ; /OE auf HIGH
            M_BUS_INPUT                     ; BUS als Eingang

            M_Chip_enable SRAM_enable       ; SRAM /CE auf LOW
            cbi CONTROL, CTRL_OUT           ; /OE auf LOW
            nop
            in SRAM_Data, BUS_IN            ; ein Byte vom Bus lesen

            M_Chip_disable SRAM_enable      ; SRAM /CE auf HIGH
            sbi CONTROL, CTRL_OUT           ; /OE auf HIGH
            ret

; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++ Page Address           : SRAM_Page_Addr_H:SRAM_Page_Addr_L
; ++++++ Address within Page    : SRAM_Addr
; ++++++ werden verändert !!!!!!!!!!!!!!!!!
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SRAM_Write_Page:
; writes internal DataArray into external RAM
            push r16
            push ZL
            push ZH
            rcall SRAM_SPDWrite         ; Software Protect Sequence zum schreiben

            M_BUS_OUTPUT                ; Bus als Ausgang schalten
            rcall SRAM_Load_Page_Addr   ; Page Adresse laden (SRAM_Page_Addr_H/L)

            ldi ZL, low(DataArray)      ; Anfangsadresse des internen SRAM_Page
```

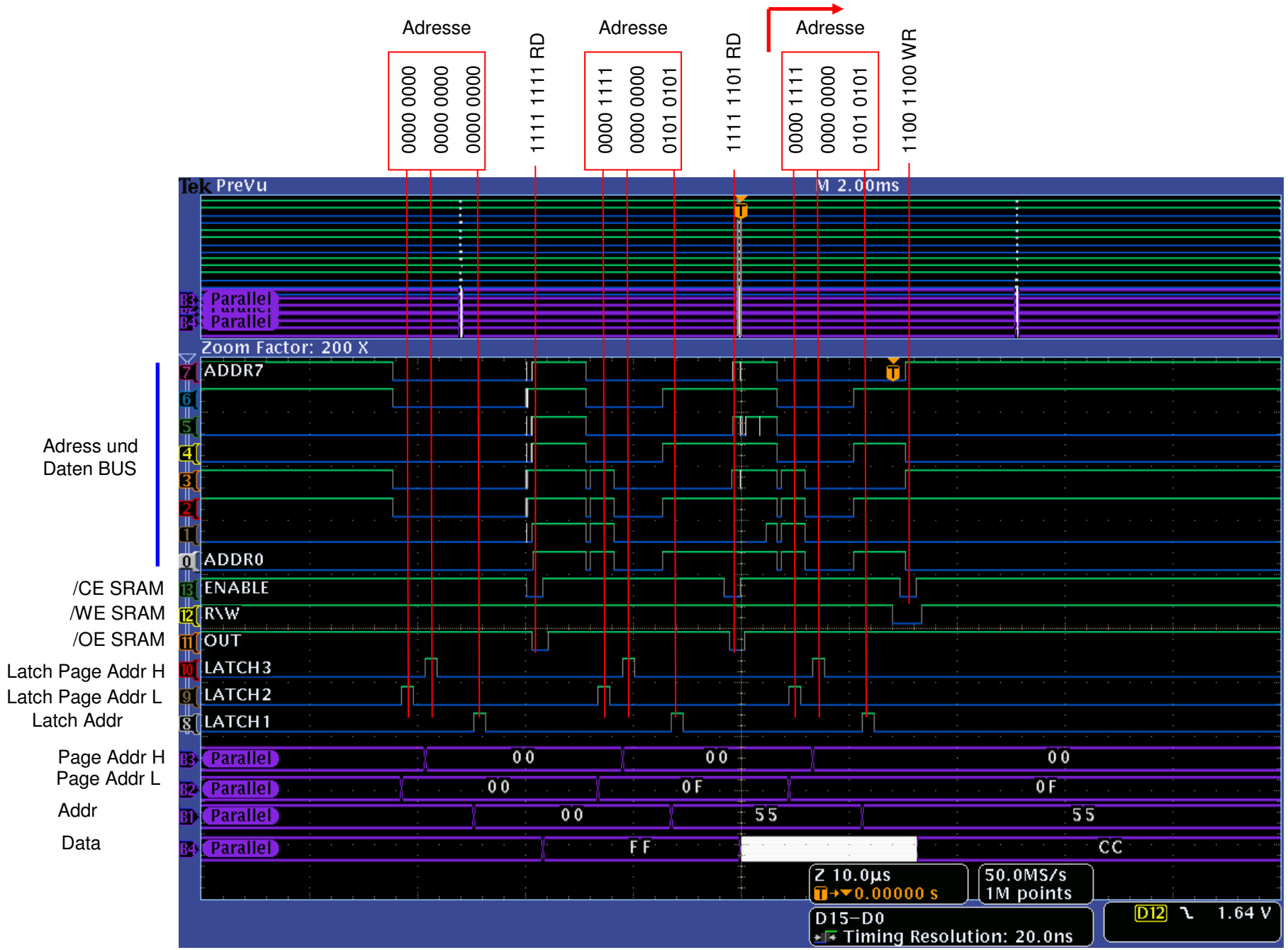
```
            ldi ZH, high(DataArray)     ; Speicherbereichs laden
            ldi SRAM_Addr, 0            ; Adresse auf Anfang einer Page

            sbi CONTROL, CTRL_OUT       ; /OE auf HIGH => Daten ins RAM schreiben
            cbi CONTROL, CTRL_RW        ; /WE auf LOW
write_loop:
            rcall SRAM_Load_Addr        ; SRAM Adresse innerhalb der Page laden (SRAM_A
ddr)
            M_Chip_enable SRAM_enable   ; SRAM /CE auf LOW

            ld SRAM_Data, Z+            ; Wert aus internem Speicher auslesen
            out BUS, SRAM_Data          ; Datenbyte auf den Bus legen
            nop
            nop
            nop
            M_Chip_disable SRAM_enable  ; SRAM /CE auf HIGH -> Daten ins eRAM schreiben

            inc SRAM_Addr               ; SRAM Adresse erhoehen
            breq write_exit
            rjmp write_loop
write_exit:
            rcall delay50us             ; 200us warten, damit der interne Schreibvorgan
g
            rcall delay50us             ; des eRAMs gestartet wird
            rcall delay50us
            rcall delay50us
            rcall delay5ms              ; Zeit um internen page write cycle zu ermoegli
chen

            sbi CONTROL, CTRL_RW        ; /WE auf HIGH

            pop ZH
            pop ZL
            pop r16
            ret

; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++++ SRAM write Byte ++++++++++++++++++++++
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SRAM_Write_Byte:

            rcall SRAM_Load_Page_Addr       ; Adresse laden
            rcall SRAM_Load_Addr

            sbi CONTROL, CTRL_RW            ; /WE auf HIGH
            sbi CONTROL, CTRL_OUT           ; /OE auf HIGH => Daten ins RAM schreiben

            cbi CONTROL, CTRL_RW            ; /WE auf LOW => ins SRAM schreiben
            M_Chip_enable SRAM_enable       ; SRAM /CE auf LOW

            nop
            out BUS, SRAM_Data              ; Datenbyte auf den Bus legen
            nop
            nop
            M_Chip_disable SRAM_enable      ; SRAM /CE auf HIGH
            sbi CONTROL, CTRL_RW            ; /WE auf HIGH

            rcall delay50us                 ; 200us warten, damit der interne Schreibvo
rgang
            rcall delay50us                 ; des eRAMs gestartet wird
            rcall delay50us
            rcall delay50us
            rcall delay5ms
            ret

; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


SRAM_Toggle_BIT:
        rcall delay50us
```

```
        M_BUS_INPUT
        ldi r16,10
        sbi CONTROL, CTRL_RW        ; /WE auf HIGH
SRAM_Toggle_BIT_loop:
        cbi CONTROL, CTRL_OUT
        M_Chip_enable SRAM_enable   ; SRAM enablen
        nop
        M_Chip_disable SRAM_enable  ; Daten ins eRAM schreiben
        sbi CONTROL, CTRL_OUT
        dec r16
        brne SRAM_Toggle_BIT_loop

SRAM_Toggle_BIT_exit:
        ret

; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ++++++++++++++++++SRAM Erase entire Chip ++++++++++++++++++
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


SRAM_SPDWrite:
                                        ; 0xAA nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00      ; Adresse uebergeben
            ldi SRAM_Page_Addr_L, 0x55      ; Adresse uebergeben
            ldi SRAM_Addr, 0x55             ; Adresse uebergeben
            ldi SRAM_Data, 0xAA             ; Datum laden (1 Byte)

            rcall SRAM_Write_Byte
                                        ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                        ; 0xA0 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0xA0

            rcall SRAM_Write_Byte
            ret



SRAM_Erase:
                                        ; 0xAA nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00      ; Adresse uebergeben
            ldi SRAM_Page_Addr_L, 0x55      ; Adresse uebergeben
            ldi SRAM_Addr, 0x55             ; Adresse uebergeben
            ldi SRAM_Data, 0xAA             ; Datum laden (1 Byte)

            rcall SRAM_Write_Byte
                                        ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                        ; 0x80 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0x80

            rcall SRAM_Write_Byte
                                        ; 0xAA nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
```

```
            ldi SRAM_Data, 0xAA

            rcall SRAM_Write_Byte
                                        ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                        ; 0x10 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0x10

            rcall SRAM_Write_Byte

            rcall delay50ms

            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x00
            sts cur_eRAM_Addr,SRAM_Page_Addr_H
            sts cur_eRAM_Addr+1,SRAM_Page_Addr_L
            ret




SRAM_Unprotect:
                                        ; 0xAA nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00      ; Adresse uebergeben
            ldi SRAM_Page_Addr_L, 0x55      ; Adresse uebergeben
            ldi SRAM_Addr, 0x55             ; Adresse uebergeben
            ldi SRAM_Data, 0xAA             ; Datum laden (1 Byte)

            rcall SRAM_Write_Byte
                                        ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                        ; 0x80 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0x80

            rcall SRAM_Write_Byte
                                        ; 0xAA nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0xAA

            rcall SRAM_Write_Byte
                                        ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                        ; 0x20 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0x20

            rcall SRAM_Write_Byte
```

```
            rcall delay50ms

            ret



SRAM_ReadProductID:
                                        ; 0xAA nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00  ; Adresse uebergeben
            ldi SRAM_Page_Addr_L, 0x55  ; Adresse uebergeben
            ldi SRAM_Addr, 0x55         ; Adresse uebergeben
            ldi SRAM_Data, 0xAA         ; Datum laden (1 Byte)

            rcall SRAM_Write_Byte
                                        ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                        ; 0x80 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0x80

            rcall SRAM_Write_Byte
                                        ; 0xAA nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0xAA

            rcall SRAM_Write_Byte
                                        ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                        ; 0x20 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0x60

            rcall SRAM_Write_Byte

            rcall delay5us
            rcall delay5us
            rcall delay5us

            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x00
            ldi SRAM_Addr, 0x00
            rcall SRAM_READ_BYTE

            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x00
            ldi SRAM_Addr, 0x01
            rcall SRAM_READ_BYTE

            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x00
            ldi SRAM_Addr, 0x02
            rcall SRAM_READ_BYTE

            ldi SRAM_Page_Addr_H, 0x07
            ldi SRAM_Page_Addr_L, 0xFF
```

```
            ldi SRAM_Addr, 0xF2
            rcall SRAM_READ_BYTE


            ldi SRAM_Page_Addr_H, 0x00      ; Adresse uebergeben
            ldi SRAM_Page_Addr_L, 0x55      ; Adresse uebergeben
            ldi SRAM_Addr, 0x55             ; Adresse uebergeben
            ldi SRAM_Data, 0xAA             ; Datum laden (1 Byte)

            rcall SRAM_Write_Byte
                                            ; 0x55 nach 0x2AAA laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x2A
            ldi SRAM_Addr, 0xAA
            ldi SRAM_Data, 0x55

            rcall SRAM_Write_Byte
                                            ; 0x80 nach 0x5555 laden
            ldi SRAM_Page_Addr_H, 0x00
            ldi SRAM_Page_Addr_L, 0x55
            ldi SRAM_Addr, 0x55
            ldi SRAM_Data, 0xF0

            rcall SRAM_Write_Byte

            ret
```


Adresse
0000 0000
0000 0000
0000 0000
1111 1111 RD
Adresse
0000 1111
0000 0000
0101 0101
1111 1101 RD
Adresse
0000 1111
0000 0000
0101 0101
1100 1100 WR
Tek PreVu
M 2.00ms
Parallel
Parallel
Zoom Factor: 200 X
ADDR7
ADDR0
ENABLE
R\W
OUT
LATCH3
LATCH2
LATCH1
Parallel 00 00 00
Parallel 00 0F 0F
Parallel 00 55 55
Parallel FF CC
Z 10.0µs
0.00000 s
50.0MS/s
1M points
D15–D0
Timing Resolution: 20.0ns
D12 1.64 V
Adress und Daten BUS
/CE SRAM
/WE SRAM
/OE SRAM
Latch Page Addr H
Latch Page Addr L
Latch Addr
Page Addr H
Page Addr L
Addr
Data