

---

```

.include "8515def.inc"

; Warning: set up FUSE-Bits ! See Datasheet!

.def temp      = r16
.def address   = r21
.def value     = r22
.def mask      = r23                ; for BitModify
.def stack     = r24
.def stack2    = r25
; mcp2515 Instructions:
.equ WRITE     = 0b00000010
.equ READ      = 0b00000011
.equ RESET     = 0b11000000
.equ BITMODIFY = 0b00000101
; mcp2515 addresses:
.equ RXB0D0    = 0b01100110
.equ RXB0D1    = 0b01100111
.equ CANINTF    = 0b00101100
.equ CANINTE    = 0b00101011
.equ CNF1       = 0b00101010
.equ CNF2       = 0b00101001
.equ CNF3       = 0b00101000
.equ BFPCTRL    = 0b00001100
.equ CANCTRL    = 0b00001111
.equ TXB0SIDH    = 0b00110001
.equ TXB0SIDL    = 0b00110010
.equ TXB0DLC     = 0b00110101
.equ TXB0D0      = 0b00110110
.equ TXB0D1      = 0b00110111
.equ TXB0CTRL    = 0b00110000
.equ TEC        = 0b00011100
.equ REC        = 0b00011101
.equ EFLG       = 0b00101101

.org 0x000

    ; Stackpointer
    ldi     temp,      LOW(RAMEND)
    out     SPL,       temp
    ldi     temp,      HIGH(RAMEND)
    out     SPH,       temp

    ; mcp Startup Delay, now SPI can be used with mcp2515
    rcall   wait_500ms

    ; SPI Master Init
    ldi     temp,      0b10111000                ; Output = SCK & MOSI & /SS & LED
    out     DDRB,      temp
    sbi     PortB,     4                        ; /CS High

    ; SPIEnabled, MasterMode, SPI Clock Rate = OSC/128
    ; (0<<SPIE) | (1<<SPE) | (0<<DORD) | (1<<MSTR) | (0<<CPOL) | (0<<CPHA) | (1<<SPR1) | (1<<SPR0)
    ldi     temp,      0b01010011
    out     SPCR,      temp

    ; MCP2515 init
    rcall   mcp_reset
    rcall   wait_100ms                ; a must be

; ===== MCP2515 CONFIGURATION MODE =====

; MCP on Quarz, 4MHz, Can Speed 100kbps, 1 Bittime = 10 TQ

; CNF1:
    ldi     address,   CNF1
    ldi     value,     0x01
    rcall   sendbyte

; CNF2:
    ldi     address,   CNF2
    ldi     value,     0xA0

```

---

```
        rcall    sendbyte

; CNF3:
    ldi        address,    CNF3
    ldi        value,      0x02
    rcall    sendbyte

; ===== MCP2515 NORMAL -ONE SHOT- MODE =====
==

; Normal Mode:
    ldi        address,    CANCTRL
    ldi        value,      0b00001000        ; OSMODE
    rcall    sendbyte

; Vorgehen:
; 1. Transmit Buffer 0 als "leer" definieren (nötig, weil Init-Value="voll")
; 2. *send*
; 2.1. weiter, wenn TXREQ nicht gesetzt, sonst warten
; 2.2. weiter, wenn Transmit Buffer 0 "leer", sonst warten
; 2.3. Übertragen der [IDs], [DLC], [eigentliche Daten] zum MCP
; 2.4. Transmitt Buffer 0 als "voll" definieren
; 2.5. Sende-Befehl: TXREQ setzen für Transmit Buffer 0
; 2.6. warten... zurück zu *send*

; Set Transmitt Buffer = Empty
ldi        address,    CANINTF
ldi        mask,      0b00000100
ldi        value,      0b00000100        ; Transmit Buffer set to "empty" (TX0IF
)
    rcall    modifybyte

send:

; is TXREQ-Flag cleared?
ldi        address,    TXB0CTRL
rcall    getbyte
sbrl        temp,      3                ; bit 3 = TXREQ clear?
rjmp      send          ; NOT clear... keep on waiting

tb:
; is Transmit-BufferEmpty?
ldi        address,    CANINTF
rcall    getbyte
sbrs        temp,      2                ; bit 2 = TX0IF set?
rjmp      tb            ; NOT set.... keep on waiting

; all requirements for sending have been satisfied
; now: new sending-procedure

; define ID (Std. ID High) (XXXXXXXX)
ldi        address,    TXB0SIDH
ldi        value,      0b00000000
rcall    sendbyte

; define ID (Std. ID Low) (XXXdddd)
ldi        address,    TXB0SIDL
ldi        value,      0b00000000
rcall    sendbyte

; define Data Length and Remote/Data Frame
ldi        address,    TXB0DLC
ldi        value,      0b00000001
rcall    sendbyte

; Data to send
ldi        address,    TXB0D0
ldi        value,      0b11100111
rcall    sendbyte

; set Transmitt Buffer 0 = Full
ldi        address,    CANINTF
```

---

---

```

    ldi    mask,        0b00000100
    ldi    value,       0b00000000    ; Transmit Buffer set to "full"
    rcall  modifybyte

    sbi    PortB,       3              ; TRANSMIT-LED PB1 on

; Send-Instruction
    ldi    address,     TXB0CTRL
    ldi    value,       0b00001011    ; set TXREQ-Flag, Highest Priority
    rcall  sendbyte

; Transmission starts, when Bus is available.

    cbi    PortB,       3              ; TRANSMIT-LED PB1 off

; Interval Time:
    rcall  wait_500ms
    rcall  wait_500ms
    rcall  wait_500ms
    rcall  wait_500ms

    rjmp  send

;..... Sub-Procedures:
;
getbyte:
    in     stack,       SPCR
    push  stack
    nop
    nop
    cbi    PortB,       4              ; /SS low

; READ COMMAND
    ldi    temp,        READ
    out    SPDR,        temp
wait_spi_g1:
    sbis   SPSR,        SPIF          ; Transmission complete?
    rjmp   wait_spi_g1
    nop
    nop
    in     temp,        SPDR          ; release SPIF here

; SET ADDRESS
    out    SPDR,        address
wait_spi_g2:
    sbis   SPSR,        SPIF          ; Transmission complete?
    rjmp   wait_spi_g2
    nop
    nop
    in     temp,        SPDR          ; release SPIF here

; DUMMY BYTE
    ldi    temp,        0b10101010
    out    SPDR,        temp
wait_spi_g3:
    sbis   SPSR,        SPIF          ; Transmission complete?
    rjmp   wait_spi_g3

; RESULT in temp:
    nop
    nop
    in     temp,        SPDR          ; release SPIF here

    sbi    PortB,       4              ; /SS high
    nop
    nop
    pop    stack
    out    SPCR,        stack

```

---

---

```
        ret
;_____
modifybyte:
    in     stack,      SPCR
    push   stack
    nop
    nop
    cbi     PortB,      4                ; /SS low

    ; BITMODIFY COMMAND
    ldi     temp,      BITMODIFY
    out     SPDR,      temp
wait_spi_b1:
    sbis    SPSR,      SPIF                ; Transmission complete?
    rjmp    wait_spi_b1
    nop
    nop
    in      temp,      SPDR                ; release SPIF here

    ; SET ADDRESS
    out     SPDR,      address
wait_spi_b2:
    sbis    SPSR,      SPIF                ; Transmission complete?
    rjmp    wait_spi_b2
    nop
    nop
    in      temp,      SPDR                ; release SPIF here

    ; MASK BYTE
    out     SPDR,      mask
wait_spi_b3:
    sbis    SPSR,      SPIF                ; Transmission complete?
    rjmp    wait_spi_b3
    nop
    nop
    in      temp,      SPDR                ; release SPIF here

    ; BITS TO BE CHANGED
    out     SPDR,      value
wait_spi_b4:
    sbis    SPSR,      SPIF                ; Transmission complete?
    rjmp    wait_spi_b4
    nop
    nop
    in      temp,      SPDR                ; release SPIF here

    sbi     PortB, 4                ; /SS high
    nop
    nop
    pop     stack                ; restore SPCR
    out     SPCR,      stack
    ret
;_____
mcp_reset:
    in     stack,      SPCR
    push   stack
    nop
    nop
    cbi     PortB,      4                ; /SS low

    ; RESET-Instruction
    ldi     temp,      RESET
    out     SPDR,      temp
wait_spi_r:
    sbis    SPSR,      SPIF                ; Transmission complete?
    rjmp    wait_spi_r
    nop
    nop
    in      temp,      SPDR                ; release SPIF here

    sbi     PortB,      4                ; /SS high
    nop
    nop
```

---

---

```

        pop    stack
        out    SPCR, stack
        ret
;
;-----
sendbyte:
    in         stack, SPCR
    push       stack
    nop
    nop
    cbi        PortB, 4
; /SS low

    ; WRITE COMMAND
    ldi        temp, WRITE
    out        SPDR, temp

wait_spi_w1:
    sbis       SPSR, SPIF
    rjmp       wait_spi_w1
    nop
    nop
    in         temp, SPDR
; release SPIF here

    ; SET ADDRESS
    out        SPDR, address

wait_spi_w2:
    sbis       SPSR, SPIF
    rjmp       wait_spi_w2
    nop
    nop
    in         temp, SPDR
; release SPIF here

    ; DATA BYTE
    out        SPDR, value

wait_spi_w3:
    sbis       SPSR, SPIF
    rjmp       wait_spi_w3
    nop
    nop
    in         temp, SPDR
; release SPIF here

    sbi        PortB, 4
; /SS high

    pop        stack
    out        SPCR, stack
    ret
;
;-----
erroroutput:
    in         stack, SPCR
    push       stack

    ldi        address, TEC
    rcall      getbyte
    rcall      serout

    ldi        address, REC
    rcall      getbyte
    rcall      serout

    ldi        address, EFLG
    rcall      getbyte
    rcall      serout

    pop        stack
    out        SPCR, stack
; restore SPCR

;
;-----
wait_500ms:
    in         stack, SPCR
    push       stack
; 2000000 Zyklen:
; -----

```

---

---

```
; warte 1999998 Zyklen:
    ldi    R17, $12
WLOOP0v: ldi    R18, $BC
WLOOP1v: ldi    R19, $C4
WLOOP2v: dec    R19
        brne   WLOOP2v
        dec    R18
        brne   WLOOP1v
        dec    R17
        brne   WLOOP0v
; -----
; warte 2 Zyklen:
    nop
    nop
; =====
        pop    stack,          ; restore SPCR
        out    SPCR,          stack
        ret

; -----
wait_100ms:
    in      stack,          SPCR
    push    stack
;    400000 Zyklen:
; -----
; warte 399999 Zyklen:
    ldi    R17, $97
WLOOP0s: ldi    R18, $06
WLOOP1s: ldi    R19, $92
WLOOP2s: dec    R19
        brne   WLOOP2s
        dec    R18
        brne   WLOOP1s
        dec    R17
        brne   WLOOP0s
; -----
; warte 1 Zyklus:
    nop
; =====
        pop    stack,          ; restore SPCR
        out    SPCR,          stack
        ret

; -----
wait_10ms:
    in      stack,          SPCR
    push    stack

;    40000 Zyklen:
; -----
; warte 39999 Zyklen:
    ldi    R17, $43
WLOOP0:  ldi    R18, $C6
WLOOP1:  dec    R18
        brne   WLOOP1
        dec    R17
        brne   WLOOP0
; -----
; warte 1 Zyklus:
    nop
; =====
        pop    stack,          ; restore SPCR
        out    SPCR,          stack
        ret

; -----
serout:
    in      stack,          SPCR
    push    stack
wait_ser:
    sbis    USR,          UDRE
    rjmp    wait_ser
;    ; wait UDR
```

---

PC)	out	UDR,	temp	; SPI-Data Register to UDR (sending to
	pop	stack		
	out	SPCR,	stack	; restore SPCR
	ret			