```
/*********************************************************************
 *
 *                  BMS UI Main File
 *
 *********************************************************************
 * FileName:        BMS_UI_Main.c
 * Processor:       PIC18F25K80
 * Compiler:        Microchip C18 v3.41
 * Company:         KIT - CN - IPE
 *
 * Author           Date         Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Reiling V.       24.07.2012   Release
 * Reiling V.       19.11.2012   Current Offset corr.
 *********************************************************************/


/*********************************************************************
 *
 *                          Include Files
 *
 ********************************************************************/
#include "BMS_UI_Main.h"


/*********************************************************************
 *
 *                              Pragmas
 *
 *********************************************************************/
#pragma config XINST   = OFF
#pragma config FOSC    = HS1
#pragma config WDTEN   = SWDTDIS // on
#pragma config WDTPS   = 256     // WD TimeOut 1024ms
#pragma config SOSCSEL = DIG     // Port C, Pin 0 & 1 => Digital
#pragma config PLLCFG  = ON


/*********************************************************************
 *
 *                              Globals
 *
 *********************************************************************/
uint16_t gEvent   = 0;
uint16_t gVoltage = 0;
int16_t  gCurrent = 50;
uint32_t gOffset  = 0;
uint8_t  gCounter = 0;

CAN_CONFIG gCAN_CONFIG = {
    SlaveNo_VAL,
    BRP_VAL,
    PROPSEG_VAL,
    PHSEG1_VAL,
    PHSEG2_VAL,
```

```
/*********************************************************************
 *
 *                  BMS UI Main File
 *
 *********************************************************************
 * FileName:        BMS_UI_Main.c
 * Processor:       PIC18F25K80
 * Compiler:        Microchip C18 v3.41
 * Company:         KIT - CN - IPE
 *
 * Author           Date         Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Reiling V.       24.07.2012   Release

 *********************************************************************/


/*********************************************************************
 *
 *                          Include Files
 *
 ********************************************************************/
#include "BMS_UI_Main.h"


/*********************************************************************
 *
 *                              Pragmas
 *
 *********************************************************************/
#pragma config XINST   = OFF
#pragma config FOSC    = HS1
#pragma config WDTEN   = SWDTDIS //OFF     // SWDTDIS
#pragma config WDTPS   = 256     // WD TimeOut 1024ms
#pragma config SOSCSEL = DIG     // Port C, Pin 0 & 1 => Digital
#pragma config PLLCFG  = ON


/*********************************************************************
 *
 *                              Globals
 *
 *********************************************************************/
uint16_t gEvent   = 0;
uint16_t gVoltage = 0;
int16_t  gCurrent = 50;

uint8_t  gCounter = 0;

CAN_CONFIG gCAN_CONFIG = {
    SlaveNo_VAL,
    BRP_VAL,
    PROPSEG_VAL,
    PHSEG1_VAL,
    PHSEG2_VAL,
```

```c
    SJW_VAL,
    PHSEG2_MODE_VAL,
    BUS_SAMPLE_MODE_VAL,
    WAKEUP_MODE_VAL,
    FILTER_MODE_VAL};


/*********************************************************************
 *
 *                       Interrupt Vector Tabelle
 *
 *********************************************************************/
#pragma code low_vetor=0x18
void interrupt_at_low_vector(void)
{
  _asm GOTO my_isr _endasm
}
#pragma code                    // Return to default code section


/*********************************************************************
 *
 *                   Interrupt High-priority service
 *
 *********************************************************************/
#pragma interrupt my_isr

void my_isr(void)
{
  static int32_t gTimeSlotCount = 0;

  /***** Timer 1 Code *****/
  if ((PIE1bits.TMR1IE) && (PIR1bits.TMR1IF))
  {
    TMR1H = 178;                // reload Timer
    TMR1L = 0;                  // 10 ms bei 16 Mhz OSC
    PIR1bits.TMR1IF = 0;        // clear event flag

    if (!(gTimeSlotCount % MS_Count)) // jeder MS_Count Interrupt ist Event
    {
       gEvent |= EV__TimeSlot;
    }

    gTimeSlotCount++;           // naechster Slot
  }
}
#pragma interrupt my_isr

/*********************************************************************
 *
 *                   Intialisierung des BMS Slave
 *
 *********************************************************************/
void ini(void) {
    uint8_t   RCONcopy = RCON;
```

```c
    SJW_VAL,
    PHSEG2_MODE_VAL,
    BUS_SAMPLE_MODE_VAL,
    WAKEUP_MODE_VAL,
    FILTER_MODE_VAL};


/*********************************************************************
 *
 *                       Interrupt Vector Tabelle
 *
 *********************************************************************/
#pragma code low_vetor=0x18
void interrupt_at_low_vector(void)
{
  _asm GOTO my_isr _endasm
}
#pragma code                    // Return to default code section


/*********************************************************************
 *
 *                   Interrupt High-priority service
 *
 *********************************************************************/
#pragma interrupt my_isr

void my_isr(void)
{
  static int32_t gTimeSlotCount = 0;

  /***** Timer 1 Code *****/
  if ((PIE1bits.TMR1IE) && (PIR1bits.TMR1IF))
  {
    TMR1H = 178;                // reload Timer
    TMR1L = 0;                  // 10 ms bei 16 Mhz OSC
    PIR1bits.TMR1IF = 0;        // clear event flag

    if (!(gTimeSlotCount % MS_Count)) // jeder MS_Count Interrupt ist Event
    {
       gEvent |= EV__TimeSlot;
    }

    gTimeSlotCount++;           // naechster Slot
  }
}
#pragma interrupt my_isr

/*********************************************************************
 *
 *                   Intialisierung des BMS Slave
 *
 *********************************************************************/
void ini(void) {
    uint8_t   RCONcopy = RCON;
```

```
RCONcopy &= 0x3B;           // IPEN, SBOREN und /PD wegfiltern
switch (RCONcopy) {
    case 0x33:              // Watch Dog Timer Reset
        ClrWdt();           // WDT Reset
        break;
    case 0x38:              // Power On Reset
        break;
    case 0x3A:              // Brown Out Reset
        break;
    case 0x2B:              // Reset by Software
        break;
    case 0x1B:              // Configuration Mismatch Reset
        break;
    default:                // Stack Over/Under Flow Reset or Combinations
        break;
}
RCON |= 0x3F;              // Reset Flags clear

OSCCONbits.IRCF0  = 0;     // 2MHz Prescaler für SPI
OSCCONbits.IRCF1  = 1;
OSCCONbits.IRCF2  = 1;
OSCTUNEbits.PLLEN = 0;      // PLL on

ANCON0            = 0x0F;   // AN0, AN1, AN2, AN3 = Analog; AN4-7 = Digita
ANCON1            = 0x00;   // AN8-14 = Digital
ADCON0            = 0x0B;   // A/D Modul On, Channel 2 (AD2) selected
ADCON1            = 0x30;   // Negativ Channel = AVss, Vref- = AVss, Vref+
ADCON2            = 0x92;   // Conversion Clock = Fosc/32, Acquisition Tim

TRISAbits.TRISA0 = 1;       // PortA.0 =  Analog In (High Voltage)
TRISAbits.TRISA1 = 1;       // PortA.1 =  Analog In (LEM Current)
LATAbits.LATA5   = 1;       // AD_CS Off (UH SPI CS OFF)
TRISAbits.TRISA5 = 0;       // Output AD_CS (UH SPI CS)

TRISB            = 0xFB;    // Port B Input (PB.2 = CAN_1_TX Output)

LATCbits.LATC0   = 1;       // AD_CS_2 Off (IH SPI CS OFF)
TRISCbits.TRISC0 = 0;       // Output AD_CS_2 (IH SPI CS)
LATCbits.LATC1   = 1;       // SW_ON_OFF Off (I Integrator Reset)
TRISCbits.TRISC1 = 0;       // Output SW_ON_OFF (I Integrator Reset)
LATCbits.LATC2   = 1;       // DA_CS Off (LEM_VREF DAC SPI CS OFF)
TRISCbits.TRISC2 = 0;       // Output DA_CS (LEM_VREF DAC SPI CS)
TRISCbits.TRISC3 = 0;       // Output AD_CLK (SPI CLK)
TRISCbits.TRISC5 = 0;       // Output AD_DOUT (SPI SDO)

// Init SPI
SSPCON1bits.SSPM  = 0x02;   // SSPM<3:0> = 0010 => SPI Master mode, clock
SSPCON1bits.CKP   = 0;      // Idle state for clock is a high level
SSPCON1bits.SSPEN = 1;      // enables SPI and configures SDA, SDI, and SC
SSPSTATbits.CKE   = 0;      // SDI by SCL Low/High
PIR1bits.SSPIF    = 0;      // clear SPI IF

//Init Timer
T1CONbits.T1CKPS0 = 1;      // Timer 1 prescalar = 0b00
T1CONbits.T1CKPS1 = 1;      // = (Fosc/4) / 8 = 2MHz
IPR1bits.TMR1IP   = 0;      // 1 = make this a low priority interrupt
```

```
RCONcopy &= 0x3B;           // IPEN, SBOREN und /PD wegfiltern
switch (RCONcopy) {
    case 0x33:              // Watch Dog Timer Reset
        ClrWdt();           // WDT Reset
        break;
    case 0x38:              // Power On Reset
        break;
    case 0x3A:              // Brown Out Reset
        break;
    case 0x2B:              // Reset by Software
        break;
    case 0x1B:              // Configuration Mismatch Reset
        break;
    default:                // Stack Over/Under Flow Reset or Combinations
        break;
}
RCON |= 0x3F;              // Reset Flags clear

OSCCONbits.IRCF0  = 0;     // 2MHz Prescaler für SPI
OSCCONbits.IRCF1  = 1;
OSCCONbits.IRCF2  = 1;
OSCTUNEbits.PLLEN = 0;      // PLL on

ANCON0            = 0x03;   // AN0, AN1 = Analog; AN2-7 = Digital
ANCON1            = 0x00;   // AN8-14 = Digital
ADCON0            = 0x03;   // A/D Modul On, Channel 0 (AD0) selected
ADCON1            = 0x30;   // Negativ Channel = AVss, Vref- = AVss, Vref+
ADCON2            = 0x92;   // Conversion Clock = Fosc/32, Acquisition Tim

TRISAbits.TRISA0 = 1;       // PortA.0 =  Analog In (High Voltage)
TRISAbits.TRISA1 = 1;       // PortA.1 =  Analog In (LEM Current)
LATAbits.LATA5   = 1;       // AD_CS Off (UH SPI CS OFF)
TRISAbits.TRISA5 = 0;       // Output AD_CS (UH SPI CS)

TRISB            = 0xFB;    // Port B Input (PB.2 = CAN_1_TX Output)

LATCbits.LATC0   = 1;       // AD_CS_2 Off (IH SPI CS OFF)
TRISCbits.TRISC0 = 0;       // Output AD_CS_2 (IH SPI CS)
LATCbits.LATC1   = 1;       // SW_ON_OFF Off (I Integrator Reset)
TRISCbits.TRISC1 = 0;       // Output SW_ON_OFF (I Integrator Reset)
LATCbits.LATC2   = 1;       // DA_CS Off (LEM_VREF DAC SPI CS OFF)
TRISCbits.TRISC2 = 0;       // Output DA_CS (LEM_VREF DAC SPI CS)
TRISCbits.TRISC3 = 0;       // Output AD_CLK (SPI CLK)
TRISCbits.TRISC5 = 0;       // Output AD_DOUT (SPI SDO)

// Init SPI
SSPCON1bits.SSPM  = 0x02;   // SSPM<3:0> = 0010 => SPI Master mode, clock
SSPCON1bits.CKP   = 0;      // Idle state for clock is a high level
SSPCON1bits.SSPEN = 1;      // enables SPI and configures SDA, SDI, and SC
SSPSTATbits.CKE   = 0;      // SDI by SCL Low/High
PIR1bits.SSPIF    = 0;      // clear SPI IF

//Init Timer
T1CONbits.T1CKPS0 = 1;      // Timer 1 prescalar = 0b00
T1CONbits.T1CKPS1 = 1;      // = (Fosc/4) / 8 = 2MHz
IPR1bits.TMR1IP   = 0;      // 1 = make this a low priority interrupt
```

```
    PIE1bits.TMR1IE   = 1;      // enable Timer interrupt
    PIR1bits.TMR1IF   = 0;      // clear any pending events
    T1CONbits.RD16    = 1;      // 16 Bit read-write mode
    T1CONbits.TMR1ON  = 1;      // Timer A run

    CAN_Init(&gCAN_CONFIG);

    INTCONbits.GIE    = 1;
    INTCONbits.PEIE   = 1;
}



/*+++++++ Main BMS Slave +++++++++++++++++++++++++++++++++++++++++++++*/
void main(void)
{
  uint16_t i = 0;


  ini();                        // Ini des BMS UI
  Measure();                    // 1. Measure
  Measure();                    // 2. Measure

  gOffset = gCurrent;           // Offset corr

  while(1)                      // Main Loop
  {
    if (gEvent & EV__TimeSlot)  // wenn Time Slot Event
    {
      gEvent &= (~EV__TimeSlot);// reset Time Slot Event

      Measure();                // Ablaufsteuerung Messung ausführen

      CAN_Write_UI();           // Tx CAN UI-Werte
    }
    ClrWdt();                   // WDT Reset
  }
}
```

```
    PIE1bits.TMR1IE   = 1;      // enable Timer interrupt
    PIR1bits.TMR1IF   = 0;      // clear any pending events
    T1CONbits.RD16    = 1;      // 16 Bit read-write mode
    T1CONbits.TMR1ON  = 1;      // Timer A run

    CAN_Init(&gCAN_CONFIG);

    INTCONbits.GIE    = 1;
    INTCONbits.PEIE   = 1;
}



/*+++++++ Main BMS Slave +++++++++++++++++++++++++++++++++++++++++++++*/
void main(void)
{
  uint16_t i = 0;


  ini();                        // Ini des BMS UI

  CAL_LEM();                    // Kalibrierung des LEM



  while(1)                      // Main Loop
  {
    if (gEvent & EV__TimeSlot)  // wenn Time Slot Event
    {
      gEvent &= (~EV__TimeSlot);// reset Time Slot Event

      Measure();                // Ablaufsteuerung Messung ausführen

      CAN_Write_UI();           // Tx CAN UI-Werte
    }
    ClrWdt();                   // WDT Reset
  }
}
```