

```

/*****
/* Program      : COMMDEFS.H
/* Function     : Common macros and defines for PIC app's not defined in pic.h
/* Author      : John F. Fitter B.E.
/*
/*      Copyright © 1997 Eagle Air Australia Pty. Ltd. All rights reserved
*****/

#ifndef _COMMDEFS_H
#define _COMMDEFS_H

#include <pic.h>

// General PIC macros (additional to pic.h)
#define clrwdt()      CLRWDT()

// Macro to define the ID bytes
#define __ID(a,b,c,d)  asm("\tpsect absdata, abs ovrl, delta=2"); \
                      asm("\tglobal id_bytes"); \
                      asm("\torg 0x2000"); \
                      asm("id_bytes"); \
                      asm("\tdb \"__mkstr(a)"); \
                      asm("\tdb \"__mkstr(b)"); \
                      asm("\tdb \"__mkstr(c)"); \
                      asm("\tdb \"__mkstr(d)");

// General macros
#define HIBYTE(intValue)  ((intValue)>>8)
#define LOBYTE(intValue) ((intValue)&0xff)
#define HINIBBLE(charValue) ((charValue)>>4)
#define LONIBBLE(charValue) ((charValue)&0xf)

// Common bit defines
#define B_IN      1
#define B_OUT     0
#define B_HIGH   1
#define B_LOW    0

// Common byte defines
#define W_IN      0xff
#define W_OUT     0
#define W_HIGH   0xff
#define W_LOW    0

// Common peripheral control defines
#define P_ON      1
#define P_OFF     0

// Common user interface defines
#define I_UP      1
#define I_DOWN    0

// Logical defines
#define TRUE      1
#define FALSE     0
#define true      TRUE
#define false     FALSE

// Hardware pullup defines
#define port_b_pullups(flag)  RBPU=flag==0

// ASCII control character defines (usefull for comms)
#define A_NUL     0
#define A_SOH     1
#define A_STX     2
#define A_ETX     3
#define A_EOT     4
#define A_ENQ     5
#define A_ACK     6
#define A_BEL     7
#define A_BS      8
#define A_HT      9
#define A_LF     0xa
#define A_VT     0xb
#define A_FF     0xc
#define A_CR     0xd
#define A_SO     0xe

```

```

#define A_SI      0xf
#define A_DLE    0x10
#define A_DC1    0x11
#define A_DC2    0x12
#define A_DC3    0x13
#define A_DC4    0x14
#define A_NAK    0x15
#define A_SYN    0x16
#define A_ETB    0x17
#define A_CAN    0x18
#define A_EM     0x19
#define A_SUB    0x1a
#define A_ESC    0x1b
#define A_FS     0x1c
#define A_GS     0x1d
#define A_RS     0x1e
#define A_US     0x1f

// Timer 0 defines
#define RTCC_INTERNAL      0 // rtcc_state values (OR together)
#define RTCC_EXT_L_TO_H    0x20
#define RTCC_EXT_H_TO_L    0x30

#define RTCC_DIV_2         0 // ps_state values (OR together)
#define RTCC_DIV_4         1
#define RTCC_DIV_8         2
#define RTCC_DIV_16        3
#define RTCC_DIV_32        4
#define RTCC_DIV_64        5
#define RTCC_DIV_128       6
#define RTCC_DIV_256       7
#define WDT_18MS           8
#define WDT_36MS           9
#define WDT_72MS           0xa
#define WDT_144MS          0xb
#define WDT_288MS          0xc
#define WDT_576MS          0xd
#define WDT_1152MS         0xe
#define WDT_2304MS         0xf

#define setup_counters(rtcc_state,ps_state) OPTION=(OPTION&0xc0)|rtcc_state|ps_state
#define get_rtcc()          TMR0
#define set_rtcc(tvalue)    TMR0=tvalue
#define get_timer0()        get_rtcc()
#define set_timer0(tvalue)  set_rtcc(tvalue)

// Timer 1 defines
#define T1_ENABLED         1 // timer 1 modes (OR together)
#define T1_INTERNAL        1
#define T1_EXTERNAL        7
#define T1_EXTERNAL_SYNC   3
#define T1_CLK_OUT         8
#define T1_DIV_BY_1         0
#define T1_DIV_BY_2         0x10
#define T1_DIV_BY_4         0x20
#define T1_DIV_BY_8         0x30

static unsigned int TMR1 @ 0x0E;
static unsigned int CCPR1 @ 0x15;
static unsigned int CCPR2 @ 0x1B;

#define setup_timer_1(mode) T1CON=mode
#define enable_timer_1(state) TMR1ON=state
#define get_timer1()        TMR1
#define set_timer1(tvalue)  TMR1=tvalue

// Timer 2 defines
#define T2_DISABLED        0 // timer 2 modes (OR together)
#define T2_DIV_BY_1        4 // T2_DISABLED must be on its own
#define T2_DIV_BY_4        5 // postscale is 1 to 16
#define T2_DIV_BY_16       7

#define setup_timer_2(mode,postscale) T2CON=mode|((postscale-1)<<3)
#define get_timer2()        TMR2
#define set_timer2(tvalue)  TMR2=tvalue

// CCP defines

```

```

static volatile bit CCP1 @ (unsigned)&PORTC*8+2; // CCP1 port pin
static volatile bit CCP2 @ (unsigned)&PORTC*8+1; // CCP2 port pin

#define CCP_OFF 0
#define CCP_CAPTURE_FE 4
#define CCP_CAPTURE_RE 5
#define CCP_CAPTURE_DIV_4 6
#define CCP_CAPTURE_DIV_16 7
#define CCP_COMPARE_SET_ON_MATCH 8
#define CCP_COMPARE_CLR_ON_MATCH 9
#define CCP_COMPARE_INT 0xa
#define CCP_COMPARE_RESET_TIMER 0xb
#define CCP_PWM 0xc
#define CCP_PWM_PLUS_1 0x1c
#define CCP_PWM_PLUS_2 0x2c
#define CCP_PWM_PLUS_3 0x3c

#define setup_ccp1(mode) CCP1CON=mode
#define setup_ccp2(mode) CCP2CON=mode
#define set_ccp1(cvalue) CCPR1=cvalue
#define set_ccp2(cvalue) CCPR2=cvalue
#define get_ccp1() CCPR1
#define get_ccp2() CCPR2

// Interrupt defines

#define GLOBAL 0x80
// #define ADC_DONE
#define RTCC_ZERO 0x20
#define RB_CHANGE 0x08
#define EXT_INT 0x10

#define INT_TIMER1 0x0100
#define INT_TIMER2 0x0200
#define INT_CCP1 0x0400
#define INT_CCP2 0x10000
#define INT_SSP 0x0800
#define INT_PSP 0x8000
#define INT_RDA 0x2000
#define INT_TBE 0x1000

#define enable_peripheral_interrupts() PEIE=1
#define enable_global_interrupts() GIE=1
#define enable_interrupts(level) \
    INTCON|=(level)&0xff; \
    PIE1|=((level)>>8)&0xff; \
    PIE2|=(level)>>16
#define enable_rtcc_interrupt() T0IE=1

#define disable_peripheral_interrupts() PEIE=0
#define disable_global_interrupts() do GIE=0;while(GIE)
#define disable_interrupts(level) \
    INTCON&=~((level)&0xff); \
    PIE1&=~(((level)>>8)&0xff); \
    PIE2&=~((level)>>16)
#define disable_rtcc_interrupt() T0IE=0

// USART defines

#define SER_MASTER 0x80
#define SER_9BIT 0x40
#define SER_TX_ENABLE 0x20
#define SER_SYNCHRONOUS 0x10
#define SER_HIGH_BAUD 4
#define SER_TSR_EMPTY 2
#define SER_ENABLE 0x80
#define SER_RX_SGL 0x20
#define SER_RX_CON 0x10

#define kbhit() RCIF

#define set_lo_baud(baud) SPBRG=(unsigned char)((2*(long)_CLOCK_/64+(long)baud)/2/\
(long)baud-1);TXSTA &= ~SER_HIGH_BAUD
#define set_hi_baud(baud) SPBRG=(unsigned char)((2*(long)_CLOCK_/16+(long)baud)/2/\
(long)baud-1);TXSTA |= SER_HIGH_BAUD

#define setup_usart_async8_lo(baud) \

```

```

    TRISC |= 0xc0;\
    TXSTA=SER_MASTER|SER_TX_ENABLE|SER_TSR_EMPTY; \
    RCSTA=SER_ENABLE|SER_RX_CON; \
    set_lo_baud(baud)

#define setup_usart_async9_lo(baud) \
    TRISC |= 0xc0;\
    TXSTA=SER_MASTER|SER_TX_ENABLE|SER_TSR_EMPTY|SER_9BIT; \
    RCSTA=SER_ENABLE|SER_RX_CON|SER_9BIT; \
    set_lo_baud(baud)

#define setup_usart_async8_hi(baud) \
    TRISC |= 0xc0;\
    TXSTA=SER_MASTER|SER_TX_ENABLE|SER_TSR_EMPTY; \
    RCSTA=SER_ENABLE|SER_RX_CON; \
    set_hi_baud(baud)

#define setup_usart_async9_hi(baud) \
    TRISC |= 0xc0;\
    TXSTA=SER_MASTER|SER_TX_ENABLE|SER_TSR_EMPTY|SER_9BIT; \
    RCSTA=SER_ENABLE|SER_RX_CON|SER_9BIT; \
    set_hi_baud(baud)

static volatile bit txd      @ (unsigned)&PORTC*8+6;    // USART serial data out pin
static volatile bit rxd      @ (unsigned)&PORTC*8+7;    // USART serial data in pin

// SPI defines
#define SPI_MASTER          0x20
#define SPI_SLAVE           0x24
#define SPI_SS_DISABLED    0x25
#define SPI_L_TO_H         0
#define SPI_H_TO_L         0x10
#define SPI_CLK_DIV_4      0
#define SPI_CLK_DIV_16     1
#define SPI_CLK_DIV_64     2
#define SPI_CLK_T2         3
#define spi_enable(en)     SSPEN=!en                // enables the spi
#define spi_data_is_in()   BF                       // tests the buffer full bit

static volatile bit sdo      @ (unsigned)&PORTC*8+5;    // serial data out pin
static volatile bit sdo_dir  @ (unsigned)&TRISC*8+5;    // serial data out direction bit
static volatile bit sdi      @ (unsigned)&PORTC*8+4;    // serial data in pin
static volatile bit sdi_dir  @ (unsigned)&TRISC*8+4;    // serial data in direction bit
static volatile bit sck      @ (unsigned)&PORTC*8+3;    // serial clock pin
static volatile bit sck_dir  @ (unsigned)&TRISC*8+3;    // serial clock direction bit
static volatile bit ss_dir   @ (unsigned)&TRISA*8+5;    // synchronous slave mode enable bit
static volatile bit BF       @ (unsigned)&SSPSTAT*8+0;  // spi buffer full bit

#endif // COMMDEFS_H

```