C library functions

- reading/writing from eeprom
- delay functions (in PRO compiler only)
- … more to be found in compiler manual, see Appendix A
- … details in include headers

Fuses configuration

```
#include <htc.h>
__CONFIG(data)
```

Description

This macro is used to program the configuration fuses that set the device into various modes of operation. The macro accepts the 16-bit value it is to update it with. 16-Bit masks have been defined to describe each programmable attribute available on each device. These attribute masks can be found tabulated in this manual in the Features and Runtime Environment section. Multiple attributes can be selected by ANDing them together. Example

```
#include <htc.h>

// fuses
__CONFIG(MCLRE_OFF & WDTE_OFF & FOSC_INTRCIO);
```

Predefined constants for PIC12F675

… see `pic12f675.h`.

```
// Config Register: CONFIG
#define CONFIG               0x2007
// Oscillator Selection bits
// RC oscillator: CLKOUT function on GP4/OSC2/CLKOUT pin, RC on GP5/OSC1/CLKIN
#define FOSC_EXTRCCLK        0x31FF
// RC oscillator: I/O function on GP4/OSC2/CLKOUT pin, RC on GP5/OSC1/CLKIN
#define FOSC_EXTRCIO         0x31FE
// INTOSC oscillator: CLKOUT function on GP4/OSC2/CLKOUT pin, I/O function on GP5/OSC1/CLKIN
#define FOSC_INTRCCLK        0x31FD
// INTOSC oscillator: I/O function on GP4/OSC2/CLKOUT pin, I/O function on GP5/OSC1/CLKIN
#define FOSC_INTRCIO         0x31FC
// EC: I/O function on GP4/OSC2/CLKOUT pin, CLKIN on GP5/OSC1/CLKIN
#define FOSC_EC              0x31FB
// HS oscillator: High speed crystal/resonator on GP4/OSC2/CLKOUT and GP5/OSC1/CLKIN
#define FOSC_HS              0x31FA
// XT oscillator: Crystal/resonator on GP4/OSC2/CLKOUT and GP5/OSC1/CLKIN
#define FOSC_XT              0x31F9
// LP oscillator: Low power crystal on GP4/OSC2/CLKOUT and GP5/OSC1/CLKIN
#define FOSC_LP              0x31F8
// Watchdog Timer Enable bit
// WDT enabled
#define WDTE_ON              0x31FF
// WDT disabled
#define WDTE_OFF             0x31F7
// Power-Up Timer Enable bit
// PWRT disabled
#define PWRTE_OFF            0x31FF
// PWRT enabled
#define PWRTE_ON             0x31EF
// GP3/MCLR pin function select
// GP3/MCLR pin function is MCLR
#define MCLRE_ON             0x31FF
// GP3/MCLR pin function is digital I/O, MCLR internally tied to VDD
#define MCLRE_OFF            0x31DF
// Brown-out Detect Enable bit
// BOD enabled
#define BOREN_ON             0x31FF
// BOD disabled
#define BOREN_OFF            0x31BF
// Code Protection bit
```

```
// Program Memory code protection is disabled
#define CP_OFF               0x31FF
// Program Memory code protection is enabled
#define CP_ON                0x317F
// Data Code Protection bit
// Data memory code protection is disabled
#define CPD_OFF              0x31FF
// Data memory code protection is enabled
#define CPD_ON               0x30FF
```

Formatting & strings

```
// formatted output
#include<stdio.h>
unsigned char sprintf(char *buf, const char *fmt, ...);


// converting strings to number
#include <stdlib.h>
double atof (const char * s);
int atoi (const char * s);
long atol (const char * s);

// string manupulation
char * strcpy (char * s1, const char * s2);
char * strncpy (char * s1, const char * s2, size_t n);
char * strncat (char * s1, const char * s2, size_t n);
...
```

Math

```
#include <stdlib.h>
int abs (int j);


#include<math.h>
double sqrt(double f);

// goinometric

double sin (double f);
double asin (double f);
double cos (double f);
double acos (double f);
double atan (double x);
double atan2 (double x, double x);

double cosh (double f)
double sinh (double f)
double tanh (double f)
```

EEPROM handling

EEPROM_READ, EEPROM_WRITE

Synopsis

```
   #include <htc.h>
   unsigned char eeprom_read (unsigned char addr);
   void eeprom_write (unsigned char addr, unsigned char value);
```

Description These function allow access to the on-chip eeprom (when present). The eeprom is not in the directlyaccessible memory space and a special byte sequence is loaded to the eeprom control registers to access the device. Writing a value to the eeprom is a slow process and the eeprom_write() function polls the appropriate registers to ensure that any previous writes have completed before writing the next datum. Reading data is completed in the one cycle and no polling is necessary to check for a read completion.

Example

```
#include <htc.h>
void main (void){

    unsigned char data;
    unsigned char address;
    address = 0x10;
    data = eeprom_read(address);

}
```

Note: It may be necessary to poll the eeprom registers to ensure that the write has completed if an eeprom_ write() call is immediately followed by an eeprom_read(). The global interrupt enable bit (GIE) is now restored by the eeprom_write() routine. The EEIF interrupt flag is not reset by this function.

auth/picembc/pic12_c_library.txt · Last modified: 2013/03/27 14:42 by res