C言語関数一覧

**(1) ライブラリ関数一覧**

|  |  |  |
| --- | --- | --- |
| 関数名 | 機能と基本形 | 使用例 |
| BusyUSART | 送信中の時１を返す TXSTAレジスタのTRMTフラグ状態を返す | while(BusyUSART( )); |
| char BusyUSART(void); |
| CloseUSART | 送受信を終了し割り込みも禁止する | CloseUSART( ); |
| void CloseUSART(void); |
| DataRdyUSART | 受信バッファに受信完了したデータがあるとき １を返す。 PIRレジスタのRCIFフラグ状態を返す | while(!DataRdyUSART( )); |
| char DataRdyUSART(void); |
| getsUSART | lenで指定された文字数の文字列を連続して受信 しbufferで ポイントされたRAMメモリに格納する。 bufferはlen+1のエリア が必要。 永久に受信完了を待ちタイムアウトは無い。 | char x[10]; getsUSART(x,5); |
| void getsUSART(char \*buffer, unsigned char len); |
| OpenUSART | USARTの使用モードを設定し動作を開始させる。 設定内容は、割込み、通信速度、同期/非同期、 8ビット/９ビット、 マスター/スレーブ、単発/連続受信 | OpenUSART(USART\_TX\_INT\_OFF & USART\_RX\_INT\_OFF & USRT\_ASYNC\_MODE & USART\_EIGHT\_BIT & USART\_CONT\_RX  & USART\_BRGH\_HIGH, 25); |
| void OpenUSART(unsigned char config, char spbrg); |
| putsUSART putsrUSART | RAM、ROMエリアの連続データを送信する。 終了はnullで判定 | char mybuff[20]; putsUSART(mybuff); |
| void putsUSART(char \*data); void putrsUSART(const rom char \*data); |
| ReadUSART | 受信バッファより１個のデータを取り出し返す。 ９ビットモードにも対応し、 USART\_Status.RX\_NINEに格納 | char x; x = ReadUSART( ); |
| char ReadUSART(void); |
| WriteUSART | 送信バッファに１個のデータを書く、９ビットモード にも対応。 ９ビット目のデータは USART\_Status.TX\_NINEにセットする。 | char x; WriteUSART(x); |
| void WriteUSART(char data); |

**(2) OpenUSART用パラメータ**  
　　OpenUSART関数にはUSARTの動作モードを設定するための多くのパラメータが  
　　必要となりますが、これらは下記のようになっています。

|  |  |  |
| --- | --- | --- |
| 設定モード | パラメータ名称 | 意味内容 |
| 割込みの許可禁止 | USART\_INT\_TX\_ON USART\_INT\_TX\_OFF USART\_INT\_RX\_ON USART\_INT\_RX\_OFF | 送信割込みの許可 　　　　　　　　　禁止 受信割込みの許可 　　　　　　　　　禁止 |
| 同期／非同期 | USART\_ASYNCH\_MODE USART\_SYNCH\_MODE | 非同期（調歩）モード 同期モード |
| ８／９ビットモード | USART\_EIGHT\_BIT USART\_NINE\_BIT | ８ビットモード ９ビットモード |
| スレーブ／マスタ | USART\_SYNC\_SLAVE USART\_SYNC\_MASTER | 同期スレーブ 同期マスタ |
| 単発／連続受信 | USART\_SINGLE\_RX USART\_CONT\_RX | 単発受信モード 連続受信モード |
| 高速／低速速度 | USART\_BRGH\_HIGH USART\_BRGH\_LOW | 高速ボーレート 低速ボーレート |

**【タイマー用Cライブラリ関数】**

MPLAB-C18で用意されているタイマー用の関数ライブラリがあります。  
各タイマー毎に用意されていて、タイマー０，１，２，３の数値で区別しています。

**(1) タイマー用C関数ライブラリ一覧**

|  |  |  |
| --- | --- | --- |
| 関数名 | 機能と基本形 | 使用例 |
| CloseTimer0 CloseTimer1 CloseTimer2 CloseTimer3 | 指定タイマーを停止し割込みを禁止する | CloseTimer0(); |
| void CloseTimer0(void); |
| OpenTimer0 OpenTimer1 OpenTimer2 OpenTimer3 | 指定のタイマーの初期設定を行う。割込み、 クロック選択、プリスケーラ値などの設定を含む | OpenTimer0(TIMER\_INT\_OFF & T0\_SOURCE\_INT &  T0\_PS\_1\_32); |
| void OpenTimer0(unsigned char config); |
| ReadTimer0 ReadTimer1 ReadTimer2 ReadTimer3 | 各タイマーのカウンタレジスタの内容を返す ８ビットと１６ビットのものがある 　　Timer0：　int(16-bits)　　TMR0H、TMR0Lレジスタ 　　Timer1：　int(16-bits)　　TMR1H、TMR1Lレジスタ 　　Timer2：　char(8-bits)　　TMR2レジスタ 　　Timer3：　int(16-bits)　　TMR3H、TMR3Lレジスタ | unsigned int result; result = ReadTimer0(); |
| unsigned int ReadTimer1(void); unsigned char ReadTimer2(void); |
| WriteTimer0 WriteTimer1 WriteTimer2 WriteTimer3 | 各タイマのタイマレジスタに値を設定する ８ビットと１６ビットのものがある 　　Timer0：　int(16-bits)　　TMR0H、TMR0Lレジスタ 　　Timer1：　int(16-bits)　　TMR1H、TMR1Lレジスタ 　　Timer2：　char(8-bits)　　TMR2レジスタ 　　Timer3：　int(16-bits)　　TMR3H、TMR3Lレジスタ | WriteTimer0(0); |
| unsigned WriteTimer0(unsigned int timer); unsigned WriteTimer2(unsigned char timer); |

**(2) OpenTimer用パラメータ一覧**  
　　OpenTimer関数には各タイマの動作モードを設定するための多くのパラメータが  
　　必要となりますが、これらは下記のようになっています。

|  |  |  |  |
| --- | --- | --- | --- |
| タイマー種別 | 項目種別 | パラメータ名称 | 意味内容 |
| 全部共通 | 割込み許可禁止 | TIMER\_INT\_ON TIMER\_INT\_OFF | タイマー割込み許可 　　　　　　　　　禁止 |
| タイマ０ | モード設定 | T0\_8BIT T0\_16BIT | ８ビットモード指定 １６ビットモード指定 |
| 外部クロックエッジ | T0\_EDGE\_FALL T0\_EDGE\_RISE | 外部クロック立下り 外部クロック立上り |
| クロック選択 | T0\_SOURCE\_EXT T0\_SOURCE\_INT | 外部クロック指定（I/Oピン) 内部クロック指定(Tosc) |
| プリスケーラ値 | T0\_PS1\_1 T0\_PS1\_2 T0\_PS1\_4 T0\_PS1\_8 T0\_PS1\_16 T0\_PS1\_32 T0\_PS1\_64 T0\_PS1\_128 T0\_PS1\_256 | １：１ １：２ １：４ １：８ １：１６ １：３２ １：６４ １：１２８ １：２５６ |
| タイマ１ | モード設定 | T1\_8BIT\_RW T1\_16BIT\_RW | ８ビットモード指定 １６ビットモード指定 |
| クロック選択 | T1\_SOURCE\_EXT T1\_SOURCE\_INT | 外部クロック指定（I/Oピン) 内部クロック指定(Tosc) |
| プリスケーラ値 | T1\_PS\_1\_1 T1\_PS\_1\_2 T1\_PS\_1\_4 T1\_PS\_1\_8 | １：１ １：２ １：４ １：８ |
| 発振回路許可禁止 | T1\_OSC1EN\_ON T1\_OSC1EN\_OFF | 発振回路を使用する 発振回路を使用しない |
| 外部クロック入力 の内部同期の指定 | T1\_SYNC\_EXT\_ON T1\_SYNC\_EXT\_OFF | 内部クロックに同期させる 　　　　　　　　　させない |
| CCP用のクロック指定 | T1\_SOURCE\_CCP T1\_CCP1\_T3\_CCP2 | CCP1,2両方のクロックに使用 CCP１だけのクロックに使用 |
| タイマ２ | プリスケーラ値 | T2\_PS1\_1 T2\_PS1\_4 T2\_PS1\_16 | １：１ １：４ １：１６ |
| ポストスケーラ値 | T2\_POST\_1\_1 T2\_POST\_1\_2 | １：１ １：２ |
| タイマ３ | クロック選択 | T3\_SOURCE\_EXT T3\_SOURCE\_INT | 外部クロック指定（I/Oピン) 内部クロック指定(Tosc) |
| モード設定 | T3\_8BIT\_RW T3\_16BIT\_RW | ８ビットモード指定 １６ビットモード指定 |
| プリスケーラ値 | T3\_PS\_1\_1 T3\_PS\_1\_2 T3\_PS\_1\_4 T3\_PS\_1\_8 | １：１ １：２ １：４ １：８ |
| クロックの選択 | T3\_OSC1EN\_ON T3\_OSC1EN\_OFF | タイマー１のクロックを使う 　　　　　　　　　使わない |
| 外部クロック入力 の内部同期の指定 | T3\_SYNC\_EXT\_ON T3\_SYNC\_EXT\_OFF | 内部クロックに同期させる 　　　　　　　　　させない |
| CCP用のクロック指定 | T3\_SOURCE\_CCP T1\_CCP1\_T3\_CCP2 | CCP1,2両方のクロックに使用 CCP２だけのクロックに使用 |

**【A/D変換用C言語関数】**

MPLAB-C18で用意されているA/D変換モジュール用のC言語ライブラリを  
説明します。  
A/D変換の制御は比較的やさしいのですが、Acquisition Timeを十分確保  
しないと精度が出なくなるので要注意です。  
  
**(1) ライブラリ関数の種類と機能**  
　　MPLAB-C18にあらかじめ用意されているA/D変換モジュール用の関数は  
　　下記表となっています。

|  |  |  |
| --- | --- | --- |
| 関数名 | 機能と基本形 | 使用例 |
| BusyADC | A/D変換中の時１を返す ADCON0レジスタのGOビットの状態を返す | while(BusyADC( )); |
| char BusyADC(void); |
| CloseADC | まずADCON0のADONビットをクリアしてA/Dコンバータを 未使用状態にしてから割り込みも禁止する | CloseADC( ); |
| void CloseADC(void); |
| ConvertADC | A/D変換を開始する。（ADCON0のGOビットを１にする） | ConvertADC(); |
| void ConvertADC(void); |
| OpenADC | A/D変換機能の使用モードを設定し動作を開始させる。 設定内容は、変換クロック、割込み、データ配置、 基準電圧、チャンネル数、チャネル選択 | OpenADC(ADC\_FOSC\_32 & 　ADC\_RIGHT\_JUST &  　ADC\_1ANA\_0REF, 　ADC\_CH & ADC\_INT\_OFF); |
| void OpenADC(unsigned char config, 　　　　　　　　　unsigned char config2); |
| ReadADC | A/D変換した結果の符号付２バイトのデータを返す。 右詰、左詰はOpenADCの設定による | int result; result = ReadADC( ); |
| int ReadADC(void); |
| SetChanADC | 一度チャンネル０に戻してから指定したチャンネルを選択する。 | SetChanADC(ADC\_CH0); |
| void SetChanADC(unsigned char channel); |

**(2) OpenADC用パラメータ**  
　　OpenADC関数にはA/D変換モジュールの動作モードを設定するための多くのパラメータが  
　　必要となりますが、これらは下記のようになっています。

|  |  |  |
| --- | --- | --- |
| 設定項目 | パラメータ名称 | 意味内容 |
| クロック選択 | ADC\_FOSC\_2 ADC\_FOSC\_4 ADC\_FOSC\_8 ADC\_FOSC\_16 ADC\_FOSC\_32 ADC\_FOSC\_64 ADC\_FOSC\_RC | Fosc/2 Fosc/4 Fosc/8 Fosc/16 Fosc/32 Fosc/64 内部クロック |
| 結果の詰め方 | ADC\_RIGHT\_JUST ADC\_LEFT\_JUST | 右詰め 左詰め |
| 使用チャンネル数 とリファレンス | ADC\_8ANA\_0REF  ADC\_7ANA\_1REF  ADC\_5ANA\_0REF ADC\_4ANA\_1REF ADC\_3ANA\_0REF ADC\_2ANA\_1REF ADC\_0ANA\_0REF ADC\_6ANA\_2REF ADC\_6ANA\_0REF ADC\_5ANA\_1REF ADC\_4ANA\_2REF ADC\_3ANA\_2REF ADC\_2ANA\_2REF ADC\_1ANA\_0REF ADC\_2ANA\_0REF | Vref+=Vdd,Vref-=Vss ８チャンネル全部使用 AN3=Vref+　他全チャン ネル使用 Vref=vdd Vref-=Vss AN3=Vref+ Vref+=Vdd、Vref-=Vss AN3=Vref+ 全部ディジタル AN3=Vref+、AN2=Vref- Vref+=Vdd、Vref-=Vss AN3=Vref+、Vref-=Vss AN3=Vref+、AN2=Vref- AN3=Vref+、AN2=Vref- AN3=Vref+、AN2=Vref- AN0のみ AN3=Vref+、AN2=Vref- |
| チャンネル指定 | ADC\_CH0 ADC\_CH1 ADC\_CH2 ADC\_CH3 ADC\_CH4 ADC\_CH5 ADC\_CH6 ADC\_CH7 | SetChanADC関数の パラメータとしても使用する |
| 割込み | ADC\_INT\_ON ADC\_INT\_OFF | 割込みを使用 割込み使わず |

|  |  |
| --- | --- |
| ■**MCC18 ライブラリ <delays.h>** | [http://sky.geocities.jp/home_iwamoto/imgs/next1.gif](http://sky.geocities.jp/home_iwamoto/usb_libc_02.htm) |
| |  |  | | --- | --- | |  | 2011年1月8日 　PIC18F14K50のプログラム開発で良く使用するMPLAB C18 C COMPILER LIBRARIESのdelay関数を説明します。  　このライブラリのインストラクションクロックを何クロック分遅延に使用するかを指定します。そのため、クロック周波数により各関数の遅延時間が変化します。代表的なクロック周波数と、遅延時間を一覧にまとめました。なお、クロックの1/4がインストラクションクロックになります。  ＨＰ内のプログラム例　[**LCD表示器**](http://sky.geocities.jp/home_iwamoto/usb_14_02.htm) | |  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 関数名 | 48Mhz | 16Mhz | 4Mhz | 引数 | | Delay1TCY | 0.08μSec | 0.25μSec | 1μSec | なし | | Delay10TCYx | 0.83μSec | 2.5μSec | 10μSec | あり | | Delay100TCYx | 8.3μSec | 25μSec | 100μSec | あり | | Delay1KTCYx | 83μSec | 250μSec | 1mSec | あり | | Delay10KTCYx | 0.83mSec | 2.5mSec | 10mSec | あり |   引数は１～２５５、遅延時間は表の引数倍となる | |  | 使用例　（クロック周波数４Mhzの場合）   |  |  | | --- | --- | | #include <delays.h> // //------ Sample // Delay1TCYx(); Delay1KTCYx(100); Delay10KTCYx(200); | // ライブラリを参照する    // 1μSecの遅延  // 100mSecの遅延 // ２Secの遅延 | | | |
|  | |
|  |  |

参考URL　http://www.picfun.com/mod1800.html