PicBasic Pro Compiler

microEngineering Labs, Inc.

完全日本語訳サンプル(テック・ハンゾウガネ訳)

シリアル EEPROM 命令　I2CREAD

LCD　　命令　LCDOUT

RS-232C アウト命令　SEROUT

尚、日本語訳文から派生する利用者のいかなる不利益もテック・ハンゾウガネ諸橋義明は責任を負いません。

I2CREAD

I2CREAD

DataPin,ClockPin,Control,{*Address*,}[*Var*{,*Var...*}]{,*Label*}

Control とオプションの Address バイトを ClockPin と DataPin に送出し、受け取ったバイトを変数の中に格納してください。 ClockPin と DataPin は定数、ナンバー0-15（e.g. B0 ）あるいはピン名前（e.g. PORTA.0 ）を含む0-15、あるいは変数であることができます。I2CREAD と I2CWRITE がマイクロチップ 24LC01B と類似のデバイスのような2ワイヤーの I2C インタフェースでシリアル EEPROM にデータを読み書きに使うことができます。 それが、電源が止められた後さえ、保持されるように、データは外部の不揮発メモリに格納されることができます。これらのコマンドは I2C マスターモードで動作して、そして同じく温度センサや A/D コンバータのような I2C インタフェースで他のデバイスと通話をするためにもわれます。

12ビットのコアのために PICmicro MCU のみ、I2C クロックとデータピンが DEFINE によってコンパイル時に固定されます。 このインフォメーションがコンパイラによって無視されるけれども、それらはまだ I2CREAD 命令で指定されなければなりません。

```
DEFINE I2C_SCL PORTA,1      'For 12-bit core only
DEFINE I2C_SDA PORTA,0      'For 12-bit core only
```

コントロールバイトの上7ビットはチップ選択あるいは追加のアドレスインフォメーションとともに、特定のデバイスによって制御コードを含んでいます。 下位 bit はそれが読みまたは書き込み命令であって、そしてクリアしておかれるべきであるかどうかを示す内部のフラグです。コントロールバイトのためのこのフォーマットはオリジナルの PicBasic コンパイラによって使われるフォーマットと異なっています。PBP I2C オペレーションでこのフォーマットを使うことを確認してください。例えば、24LC01B とコミュニケートするとき、制御コードは%1010です、そしてチップ選択は使われていません、それでコントロールバイトは%10100000あるいは $A0 であるでしょう。 様々な役割のコントロールバイトのフォーマットを次に上げます：

Device	Capacity	Control	Address size
24LC01B	128 bytes	%1010xxx0	1 byte
24LC02B	256 bytes	%1010xxx0	1 byte
24LC04B	512 bytes	%1010xxb0	1 byte
24LC08B	1K bytes	%1010xbb0	1 byte
24LC16B	2K bytes	%1010bbb0	1 byte
24LC32B	4K bytes	%1010ddd0	2 bytes
24LC65	8K bytes	%1010ddd0	2 bytes

bbb =ブロックセレクト(ハイ順アドレス)ビット

ddd =デバイスセレクトビット

xxx =気にかけない

アドレスサイズ送出（バイトあるいはワード）は使われる変数の大きさによって決定されます。 もしバイトサイズの変数がアドレスに使われるなら、8ビットのアドレスが送られます。 もしワードサイズの変数が使われるなら、16ビットのアドレスが送られます。 あなたがそれとコミュニケートすることを望むデバイスのために適切なサイズの変数を使うことを明示してください。定数は定数のサイズにたいへん依存したアドレスのために使われるべきではない。また表現は送出される不適当なアドレスサイズとして使用されるべきではない。もしワードサイズの変数が指定されるなら、2バイトが読まれて、ハイバイトが最初に格納されます。ローバイトが続きます。 この順序は変数を通常に格納する、ローバイトが最初の方法と異なっています。修飾子、STR、は可変的な名前の前に含まれます。 これは同時にに全部の配列（文字列）をロードすることができます。 もしSTR が指定されるなら、次の変数はバックスラッシュ（\）と数に続いて、ワードあるいはバイト配列の名前でなくてはいけません：

```
a var byte[8]

I2CREAD PORTC.4,PORTC.3,$a0,0,[STR a¥8]
```

もしワードサイズ配列が指定されるなら、それぞれの要素を構成する2バイトはローバイトは最初に読み込まれる。 これは単純なワードが読まれる方法の正反対であって、そしてコンパイラが通常ワードサイズ変数を格納する方法と一致している。もしオプションラベルが含まれるなら、このラベルはアクノリッジが I2C デバイスから受信されないならジャンプされるだろう。I2C インストラクションは12CExxx と16CExxx デバイスのオンチップシリアル EEPROM にアクセスするために使われることができる。 ただ適切な内部のラインのピン名を I2C コマンドの部分とプログラムの最上部において DEFINE に続く場所の一部として明示しなさい：

```
DEFINE I2C_INTERNAL 1
```

12CE67xデバイスのために、データラインはGPIO.6である、そしてクロックラインはGPIO.7である。16CE62x デバイスのために、データラインは EEINTF.1である、そしてクロックラインは EEINTF.2である。 もっと多くの情報のためそれらのデバイスのマイクロチップデータシートを見なさい。I2C インストラクションのタイミングは（100KHz）から8MHz までのクロック速度でアクセス可能であるであろう標準スピードデバイスにセットされる。 高速モードデバイス（400KHz）は20MHz まで使える。 もし8MHz 以上において標準スピードデバイスにアクセスすることが望まれるなら、次の DEFINE はプログラムに加えられるべきである：

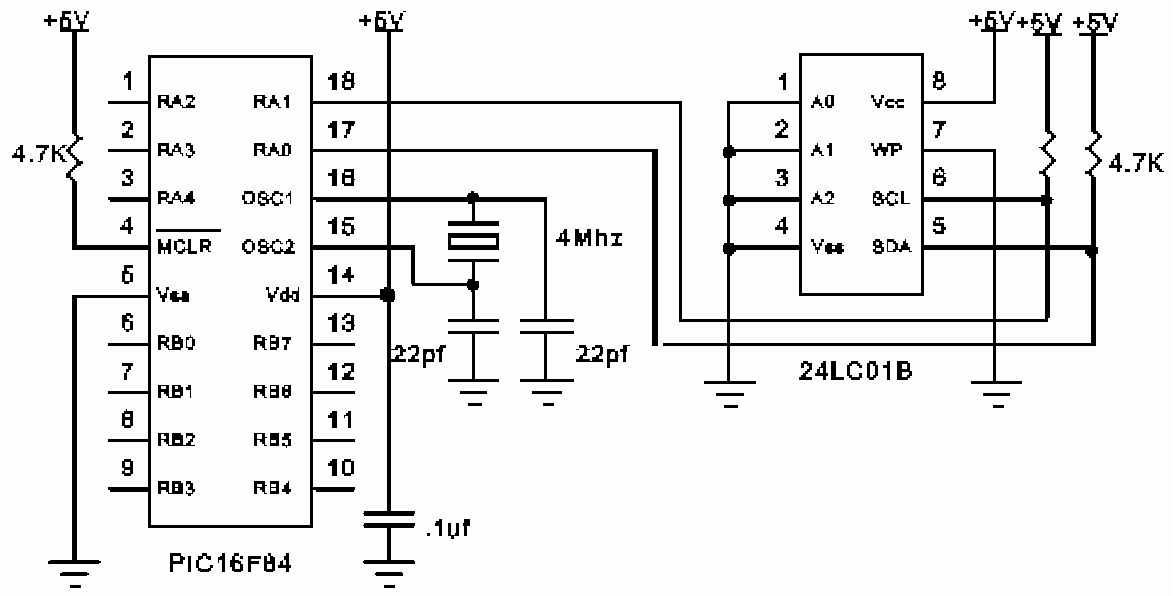
```
DEFINE I2C_SLOW 1
```

受信デバイスによって休止できるI2C バス上でその保持クロックライン low のため遷移させなさい。 これを可能にするために、次の DEFINE はプログラムに加えられるべきである：

```
DEFINE I2C_HOLD 1
```

I2C クロックとデータ行は次の回路図でそれらはまた双方向性のオープンコレクター規則で共に動作しているとして4．7K抵抗でVccにプルアップされるべきだ。2極式のI2C クロック線を作製するために、オープンコレクターのかわりに次の DEFINE をプログラムに追加できる

```
DEFINE I2C_SCLOUT 1
```




```
addr    var     byte
cont    con     %10100000
        addr = 17       ' Set address to 17
                        ' Read data at address 17 into B2
        I2CREAD PORTA.0,PORTA.1,cont,addr,[B2]
```

さらなる情報のため、それらや **I2CREAD** や **I2CWRITE** コマンドが使われているだろうデバイス上で Microchip ANon-Volatile Memory （不揮発メモリ） Products Data Book@を見なさい。

LCDOUT

LCDOUT *Item*{,Item...}

インテリジェント液晶上に Item を表示します。 PBP は日立44780コントローラーあるいは同等品で LCD モジュールをサポートします。 これらの LCD は通常1つのエッジにおいて14あるいはの16ピンの単列あるいは二重横列のヘッダーがあります。もしポンドサイン（ # ）が Item に前置されるなら、それぞれのデジットの ASCII 表記は LCD に送られます。 LCDOUT（12ビットのコア以外のすべてのデバイスの上で）は同じく SEROUT2 と共に使われる修飾子のいずれも使うことができます。 このインフォメーションのために SEROUT2 の上にセクションを見てください。

Modifier	Operation
{I}{S}BIN{1..16}	2進数を送信しなさい
{I}{S}DEC{1..5}	10進数を送信しなさい
{I}{S}HEX{1..4}	16進数を送信しなさい
REP c¥n	文字cがn回繰り返し送信しなさい
STR ArrayVar{¥n}	n個の文字ストリングを送信しなさい

最初のコマンドを LCD に送る前に、プログラムが少なくとも 0.5 秒を待つべきです。 それは LCD がスタートアップする間を取ります。いろんな文字あるいはコマンドが LCDOUT を使って送られる最初に、LCD は初期化されます。もし動作中に何らかの理由でそれが電源停止やそれから電源バックアップされるなら、LCDOUT を使う次のとき、内部のフラグを再初期化することをプログラムに示すためリセットできます：

```
FLAGS = 0
```

コマンドは、$FE に続くコマンドを送出することによって LCD に送られます。若干の有用なコマンドが次の表に載せられています：

Command	Operation
$FE, 1	クリアディスプレイ
$FE, 2	リターンホーム（最初のラインの始め）
$FE, $0C	カーソルオフ
$FE, $0E	アンダーラインカーソルオン
$FE, $0F	ブリンキングカーソルオン
$FE, $10	カーソルを左へ1つポジション移動してください

$FE, $14	カーソルを右へ1つポジション移動してください
$FE, $C0	カーソルを2番目のラインの始めに移動してください
$FE, $94	カーソルを3番目のラインの始めに移動してください
$FE, $D4	カーソルを4番目のラインの始めに移動してください

カーソルを2ラインのディスプレイの2番目のラインの始まりに動かすコマンドがあることに注意してください。 たいていの LCD に、表示された文字と行はディスプレイメモリーで連続していません－ロケーション間でブレークすることができます。 たいていの 16x2 ディスプレイのため、最初のラインは$0で始まります、そして2番目のラインは$40で始まります。 コマンド：

```
LCDOUT $FE, $80 + 4
```

は最初のラインの4番目の位置で書き始めることを設定します。
16x1 ディスプレイが通常1番目と2番目の8文字のためにメモリー配置のあいだのブレークで 8x2 ディスプレイとしてフォーマットされます。 上の表に示されるように、4ラインのディスプレイはまた複合しているメモリーマップを持っています。文字記憶場所と追加のコマンドのために特定の LCD デバイスのデータシートを見てください。.

```
LCDOUT $FE, 1, "Hello"    'ディスプレイをクリアして、そして「 Hello 」を表示してください
LCDOUT $FE,$C0,"World"    '2番目のラインにジャンプして"World"を表示してください
LCDOUT B0, #B1            'B0 と B1 の十進 ASCII 値を表示してください
```

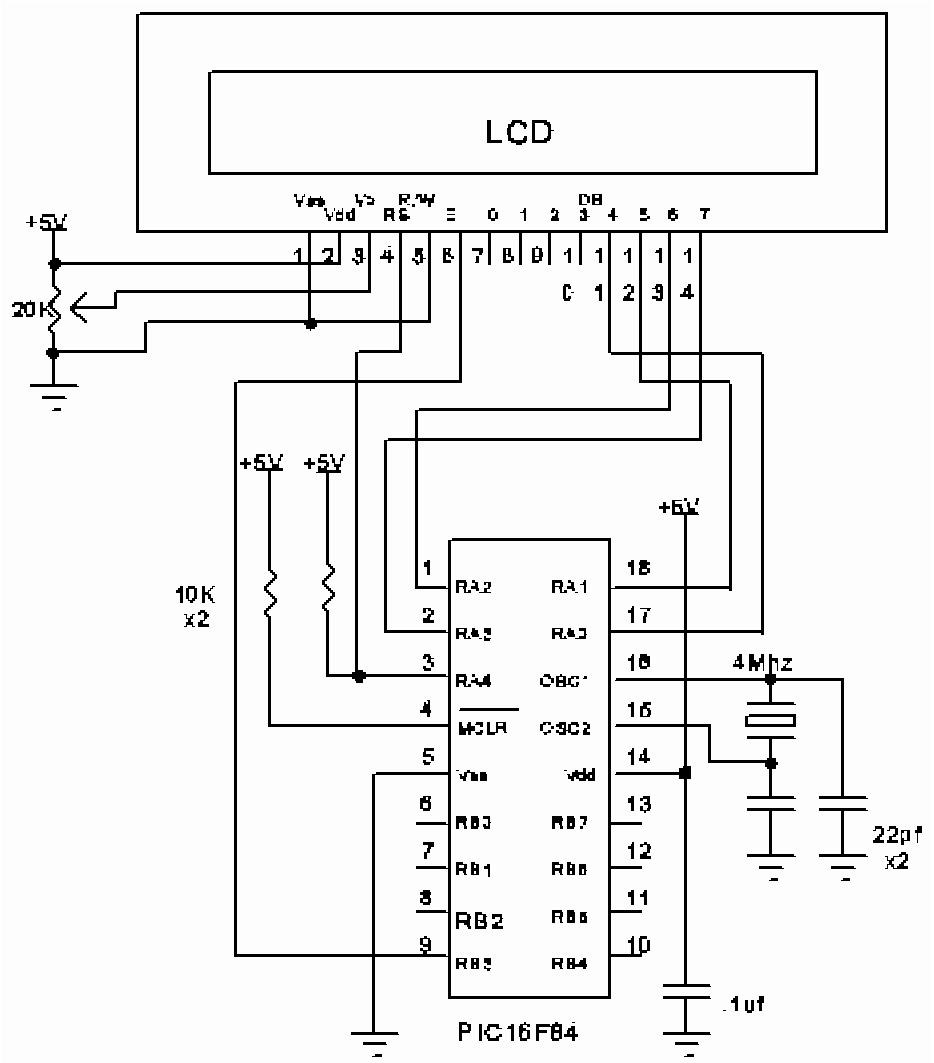
LCD は4ビットのバスあるいは8ビットのバスを使って PICmicro に接続できる。 もし8ビットのバスが使われるなら、すべての8ビットは1つのポートの上になければなりません。 もし4ビットのバスが使われるなら、最下位4あるいは1つのポートの最上位4bit に接続しなくてはいけません。 イネーブルとレジスタセレクトはどんなポートピンとでも接続されても良い。 LCDOUT コマンドが書き込みのみである時、 R/W はグランドに結線されるべきです。
LCD は4ビットのバスあるいは8ビットのバスを使って PICmicro に接続できる。 もし8ビットのバスが使われるなら、すべての8ビットは1つのポートの上になければなりません。 もし4ビットのバスが使われるなら、最下位4あるいは1つのポートの最上位4bit に接続しなくてはいけません。イネーブルとレジスタセレクトはどんなポートピンとでも接続されても良い。 LCDIN コマンドが使われない時、 R/W はグランドに結線されるべきです。
PBP は、示されないなら LCD が特定の Pin に接続していると想定します。 それは PICmicro の PORTA.0 - PORTA.3, レジスタセレクト PORTA.4 とイネーブル PORTB.3 が接続されたデータライン DB4 - DB7 を持った4ビットバスで使われるであろうと想定します。それはまた LCD を2ラインのディスプレイに初期化するようにあらかじめセットされています。このセットアップを変更するため、次の DEFINE の1つまたはさらに多くの文ををあなたの PicBasic Pro プログラムのトップに置いてください：

```
' Set LCD Data port
DEFINE LCD_DREG    PORTB
```

```
' Set starting Data bit (0 or 4) if 4-bit bus
DEFINE LCD_DBIT    4
' Set LCD Register Select port
DEFINE LCD_RSREG   PORTB
' Set LCD Register Select bit
DEFINE LCD_RSBIT   1
' Set LCD Enable port
DEFINE LCD_EREG    PORTB
' Set LCD Enable bit
DEFINE LCD_EBIT    0
' Set LCD bus size (4 or 8 bits)
DEFINE LCD_BITS    4
' Set number of lines on LCD
DEFINE LCD_LINES   2
' Set command delay time in us
DEFINE LCD_COMMANDUS        2000
' Set data delay time in us
DEFINE LCD_DATAUS 50
```

このセットアップは PBP に 2 ラインの LCD が PORTB.0 の上に PORTB の最上位 4 ビット、 PORTB.1 上のレジスタセレクトとイネーブル上の 4 ビットのモードでデータバスに接続していると示すでしょう。次の回路図は、デフォルトを使って、LCD を PICmicro に接続する 1 つの方法を示します：

(訳者注記)：下図は PORTA をデータバスとしたものであり上の DEFINE 文は PORTB をデータバスとしたものです。


LCD
+5V
20K
10K
x2
+5V
+5V
+5V
4Mhz
22pf
x2
.1uf
RB2
PIC16F84

SEROUT

SEROUT *Pin*,*Mode*,[*Item*{,*Item...*}]

標準的な非同期フォォーマットを使っている8データビット、ノーパリティと1ストップビット（8N1）で Pin 上に1つかそれ以上の Item を受け取ってください。　SEROUT は BS1 Serout コマンドに類似しています。 Pin は自動的にアウトプットにされます。 Pin は定数、0－15、あるいは番0－15（e.g. B0 ）あるいはピン名（e.g. PORTA.0）を含んでいる変数であることができます。
Mode 名（T2400）はファイル MODEDEFS.BAS で定義されます。 それらを使うために、PICBasic Pro プログラムのトップに

```
Include "modedefs.bas"
```

のラインを付加して下さい。

BS1DEFS.BAS と BS2DEFS.BAS がすでに MODEDEFS.BAS を含みます。 もしこれらのファイルの1つがすでに含まれているなら、再びそれを含まないでください。 このファイルを含まないで、Mode ナンバーは使われることができます。

Mode	***Mode* No.**	**Baud Rate**	**State**
T2400	0	2400	Driven True
T1200	1	1200	
T9600	2	9600	
T300	3	300	
N2400	4	2400	Driven Inverted
N1200	5	1200	
N9600	6	9600	
N300	7	300	
OT2400	8	2400	Open True*
OT1200	9	1200	
OT9600	10	9600	
OT300	11	300	
ON2400	12	2400	Open Inverted*
ON1200	13	1200	

ON9600	14	9600	
ON300	15	300	

* Open モードは 12 ビットコア PicMicro MCU でサポートしない。

SEROUT はミックスして、一つの SEROUT 文中で自由に組み合わされる3つの異なったデータ型をサポートします。

1) 文字列定数が文字のリテラル列として出力されます。

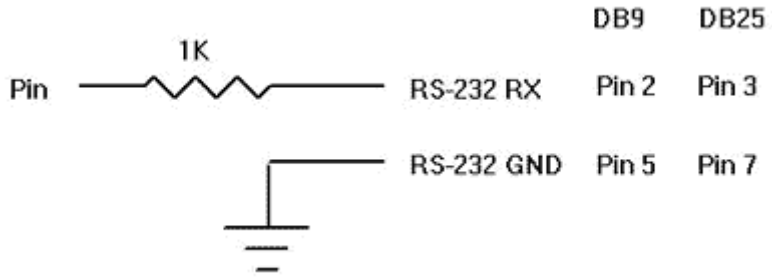
2) 数値（変数あるいは定数）が対応するＡＳＣII文字を送出する。 最も注記すること、１３が改行です、そして１０はリターンです。

3) ポンドサインを（ # ）前置がその十進値の ASCII の表現を送出する。 例えば、もし W0 ＝ １２３であるなら、それから # W0 （あるいは # １２３）が「1」、「2」、「3」を送るでしょう。

そのビットタイミングを生成するとき、SEROUT は 4MHz オシレータを想定します。 他のオシレータ値で適切なボーレートタイミングを保持するために、新しいオシレータ値に OSC 設定を DEFINE で明らかにしてください。
ある場合には、SEROUT 命令の送信速度は受信デバイスにあまりにも速く文字を渡すことができます。
DEFINE がシリアル出力伝送に文字間隔を付加します。 それらが伝送されるとき、文字間の追加時間を割り当てます。DEFINE の文字間隔は伝送されたそれぞれの文字間に1から６５,５３５マイクロ秒（.００１から６５.５３５ミリ秒）のディレーを割り当てます。
例えば、それぞれの文字の送信の間に1ミリ秒ポーズするため：

```
DEFINE CHAR_PACING 1000
```

シングルチップ RS -２３２レベルコンバータが普通で、そして高価でない間に、PICmicro の素晴らしい I/O 仕様はたいていのアプリケーションがレベルコンバータ無しで動作することを可能にします。どちらかと言えば、電流制限抵抗器の接続で反転インプット（ N300. .N9600 ）が使える。
（RS-232C は短絡許容であると仮定する）




```
SEROUT 0,N2400,[#B0,10]     ' B0 の ASCII 値を Pin0 の外にシリアルにラインフィードで送信して下さい
```