長崎大学
NAGASAKI UNIVERSITY

情報処理実習（PICマイコン）

第3版

2012年 5月29日

長崎大学 教育学部　技術教育教室　武藤浩二

3rd edition released on 29 May, 2012.

For further information, refer the following sites;
Great Cow Graphical BASIC: http://gcbasic.sourceforge.net/index.html
Microchip Technology: http://www.microchip.com/

実習1：Hello, PIC

1.0 この実習で学ぶこと

・PICマイコンの概要
・Great Cow Graphical BASIC及びPICkit2の使用方法
・発光ダイオード（LED）の使用方法
・入出力ポートの使い方（ディジタル出力）と最初のプログラム
・実行コードの作成と書き込み

1.1 PICマイコンの概要

PICマイコンはMicrochip Technology Inc.が生産販売する1チップマイクロコンピュータ（1チップマイコン）[1]の総称である．PC等のような大きなシステムの中心に位置するものではないが，その周辺で外部とデータをやり取りする機器の制御等に用いる小規模のコンピュータである．

図1.1に本実習で用いるPIC16F88（以下，16F88）の内部構造を示す．一般的なコンピュータシステムと同様に，算術論理演算装置（ALU），プログラムメモリ，プログラムカウンタ，クロック発生器，命令デコーダ，各種レジスタ，バス等から構成されている．16F88ではこの他に10ビットA/D変換器，PWM，比較器，3種類のタイマー，同期及び非同期のシリアル通信といった固有の

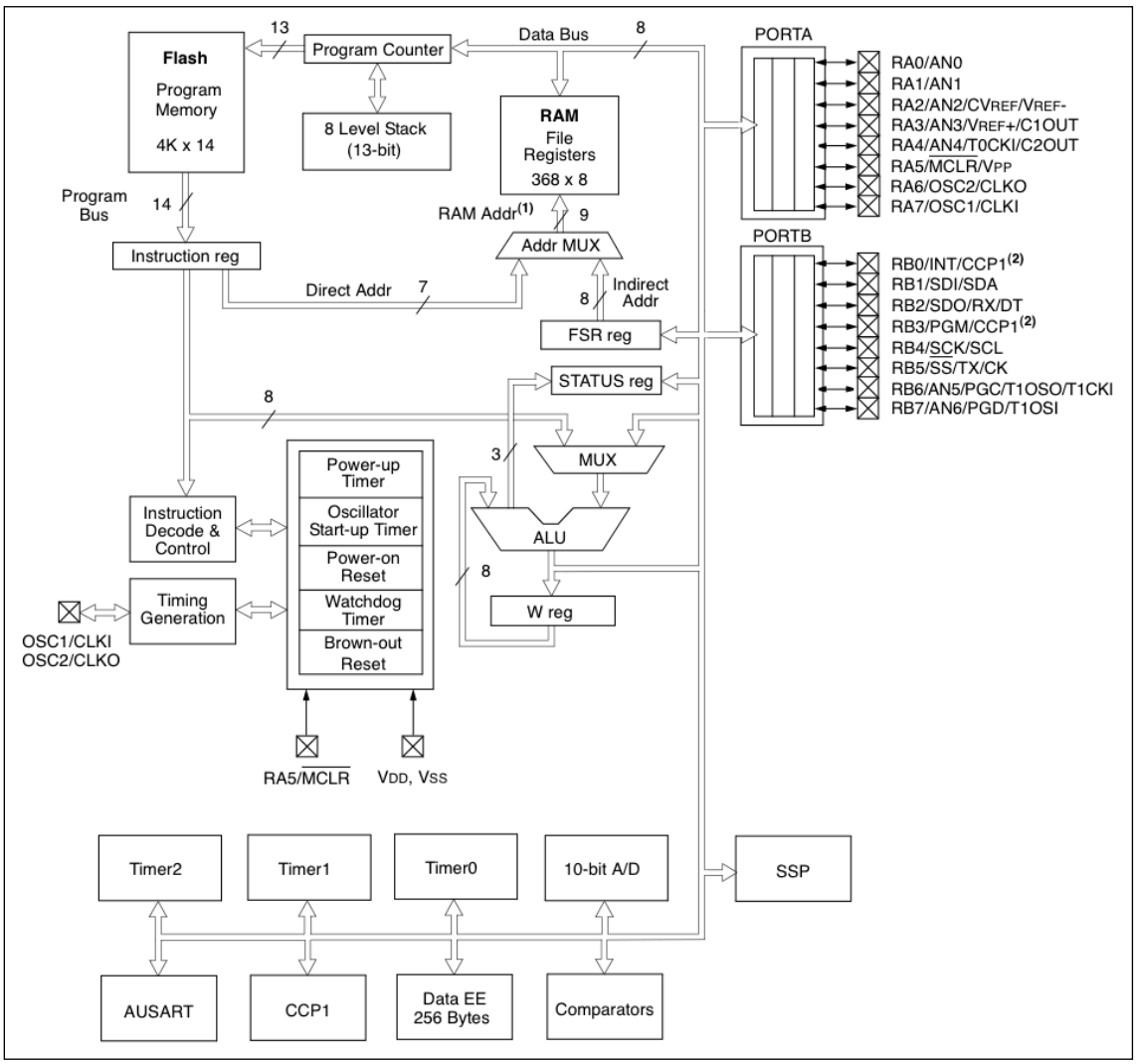


図1.1　PIC16F88のブロックダイアグラム（Microchip社 PIC16F87/88 Data Sheetより引用）

[1] 「マイクロコントローラ」と呼ぶ場合もある．

機能を有している．

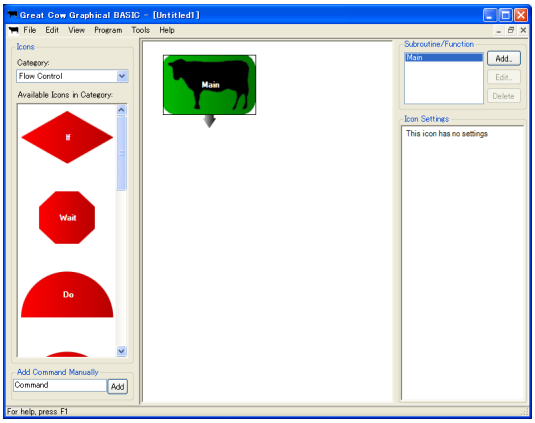
1チップマイコンにおいては，外部とのデータの入出力は「ポート」と呼ばれる端子を介して行われる．16F88は8ビットのPORTA，PORTBを有し，それぞれ各ビットごとに入力・出力を設定することができる．

1チップマイコンのプログラムは通常PC上で開発し，その実行コードを専用のハードウェア等を用いて転送する．プログラム開発のための言語としてはアセンブラ，C，BASIC等が用いられる．1チップマイコン開発に用いるCやBASICのような高級言語はPC用のプログラム開発に用いる開発環境とほぼ同様であるが，マイコン内部のポートやレジスタを直接触るため，これらの名称が変数として既定されていたり，マイコンの動作状態を細かく指定するレジスタ類の初期設定が必要である点が異なる．

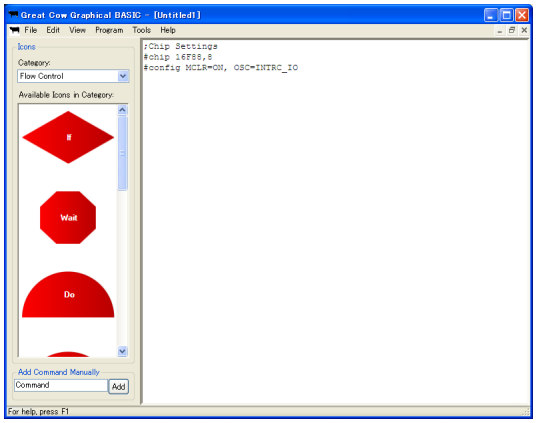
1.2 Great Cow Graphical BASIC及びPICkit2の概要

本実習でのプログラム開発は“Great Cow Graphical BASIC”（以下，GCGB）というBASIC言語開発環境及びPICkit2というプログラマ/デバッガを用いる．

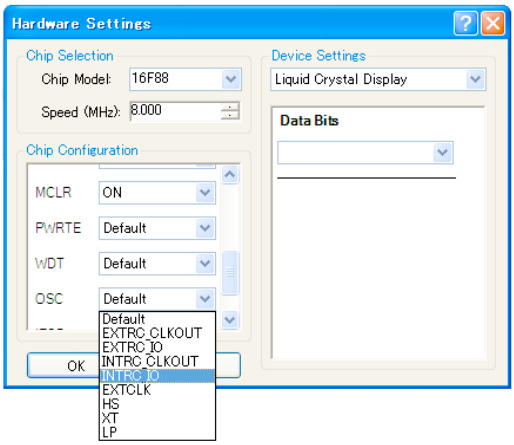
GCGBはBASIC言語で記述したソースコード（プログラムリスト）を一度アセンブリ言語に変換し，GPUtilsというアセンブラを呼び出して実行コード（HEXファイル．拡張子“.hex”）を作成するオープンソースの開発環境である．ソースコードの入力は画面からフローチャート形式のアイコンをドラッグ＆ドロップ（以下，D＆D）して必要なパラメータを与えることで行う（これが“Graphical”の所以である）．またキーボードからBASICの文法に従って入力することも可能であり，アイコンにない命令を入力したり細かな設定を行う場合に利用する．GCGBのドキュメントはオンライン形式でプログラム中から参照できるが，文書化したUser’s Guideを実習装置収納箱に添付しているので，必要に応じて参照すること．図1.2にGCGBの画面例を示す．



(a) GUI入力モード



(b) テキスト入力モード



(c) ハードウェア設定画面

図1.2　GCGBの画面例

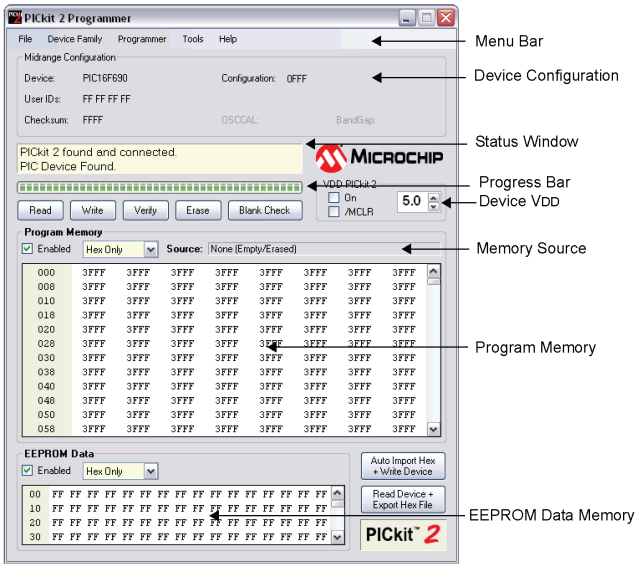


図1.3　PICkit2の起動画面

PICkit2はMicrochip社がリリースしている低価格のプログラマ/デバッガである．実行コードを作成したPCとはUSBで接続する．PICkit2とPICマイコンとの間は，本実習では5本の色分けした信号線で接続する．PICkit2の最大の特徴はIn-Circuit Serial Programmingといって，実際に使用可能な状態におかれている基板上のPICマイコンに接続して実行コードを書き込むことができる点にある．またPICkit2からPICマイコンに対して電源を供給する機能もあり，実行コードを書き込んだら直ちに動作を検証することができる．図1.3にPICkit2の起動画面を示す．

1.3 発光ダイオードの使用方法

発光ダイオード（Light Emitting Diode．以下，LED）は半導体素子の一種で，ある電圧を加えると2種類の半導体の接合面で発光するダイオードをいう．1962年に開発され，1970年代には赤および黄緑の2種類が各種電子機器の表示用途として用いられるようになった．1993年に日本で青色LEDが，1995年に緑色LEDが開発されたことで光の三原色が揃い，ディスプレイ用途への適用が始まった．また1996年には白色LEDが開発され，照明への適用も始まった．

LEDは図1.4のように「電流を流して」光らせるが，その際，LEDの両端には順方向電圧（図のV_F）と呼ばれる電圧が発生する．この電圧はLEDの種類ごとに異なり，赤や黄緑，黄色系のものは概ね1.7〜2.2[V]，青や緑，白系は3.0〜3.5[V]程度となる．LEDに流す電流は，電源電圧と電流制限抵抗R及び順方向電圧V_Fによって決まる．一例として電源電圧5[V]，順方向電圧1.7[V]，流す電流を2[mA]とすれば，電流制限抵抗Rの値は1.6[kΩ]となる．

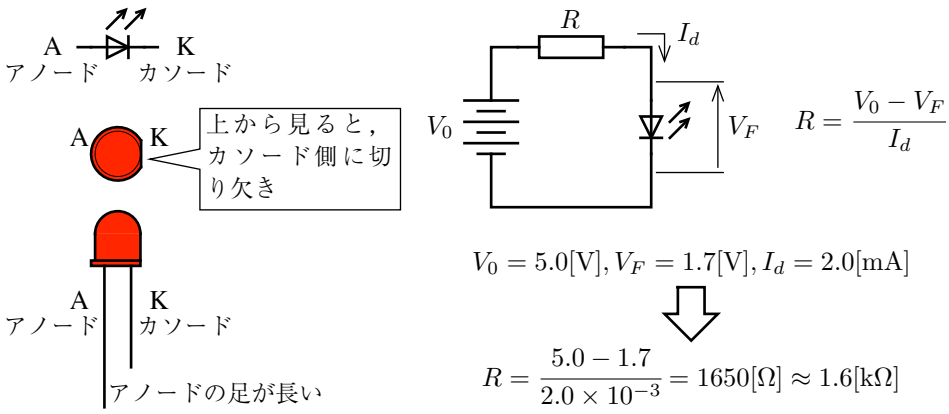
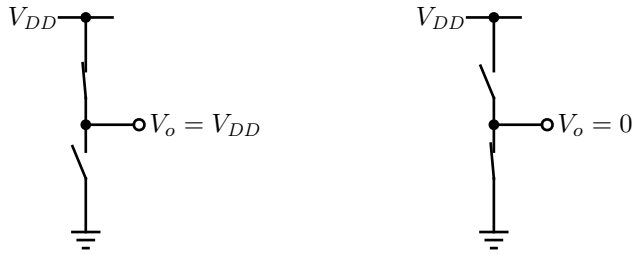


図1.4　発光ダイオードの使い方



(a) 1（あるいはH）　　(b) 0（あるいはL）

図1.5　ディジタル出力時のポート端子の考え方

1.4 入出力ポートの使い方（ディジタル出力）と最初のプログラム

PICマイコンのポートはディジタル出力として使用する場合，図1.5のような簡単なスイッチ回路のモデルで考えることができる．あるポート端子の出力値が‘1’の場合，その端子にはPICマイコンの電源電圧が出力され，外部に電流を流すことができる．一方出力値が‘0’の場合，その端子はPICマイコンの基準電位に接続され0[V]となり，外部から電流が流れこむこととなる．

本実習で使用する16F88の場合，それぞれ8ビットのPORTA及びPORTBを有している．ポートの各端子はRA0～RA7，RB0～RB7という名称を有するが，PORTA.0～PORTA.7，PORTB.0～PORTB.7と呼ぶ場合もある．各ポートの出力値はPORTA，PORTBという変数（実体はレジスタ）に8ビット整数値を与えることで出力される．PORTA及びPORTBの初期値は0である．

一方，ポートの各端子はそれぞれ個別に入力端子と出力端子に設定することができる．この設定は，TRISA，TRISBという変数（実体はレジスタ）に8ビット整数値を与えることで設定でき，0が設定されているビットが出力，1の設定されているビットが入力となる．なお，RA5はTRISAの値にかかわらず「常に入力」となっている．TRISA及びTRISBの初期値は0（出力）である．

GCGBでは，8ビットの整数値（0～255），例えば10進数の85を

・2進数では　`b'01010101'`

・16進数では `0x55`

のように，接頭語“b”に続けて2進数数値をシングルクォーテーションマークで囲んだり，あるいは“0x”を先頭につけて16進数で表す[2]．各ポートやレジスタの値を示すときは，10進数で表すより2進数や16進数で表した方が各ビットの状態を的確に把握することができる．表1.1に0～15までの10進数～4ビット2進数～16進数相互変換を示す．

表1.1　10進数，4ビット2進数及び16進数の相互変換

10進	2進	16進	10進	2進	16進
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

[2] 2進数と同様に，`h'55'` と記述しても良い（接頭語の “h”は hexadecimal の略）．

最初の実習は，ここまで説明してきた内容を用い，赤と黄緑のLEDを一定時間間隔で交互に点滅させるプログラムを考えることとする．

回路を図1.6に示す．赤LEDをRA1，緑LEDをRA6に接続している．電流制限用の抵抗値が赤と緑で異なるのは，LEDの明るさが異なるためである．この回路を実習用ブレッドボード上に構成する．

実習用ブレッドボードには，図1.7に示すようにあらかじめPICマイコンとその電源，リセットスイッチの回路が実装されている．そのため，ここで実装するのはLEDと電流制限抵抗のみとなる．図1.8のように，追加する部品をブレッドボードに挿入して図1.6の回路を実装する．この際，LEDの極性に注意する（リード線の長いほうが＋：アノード）．

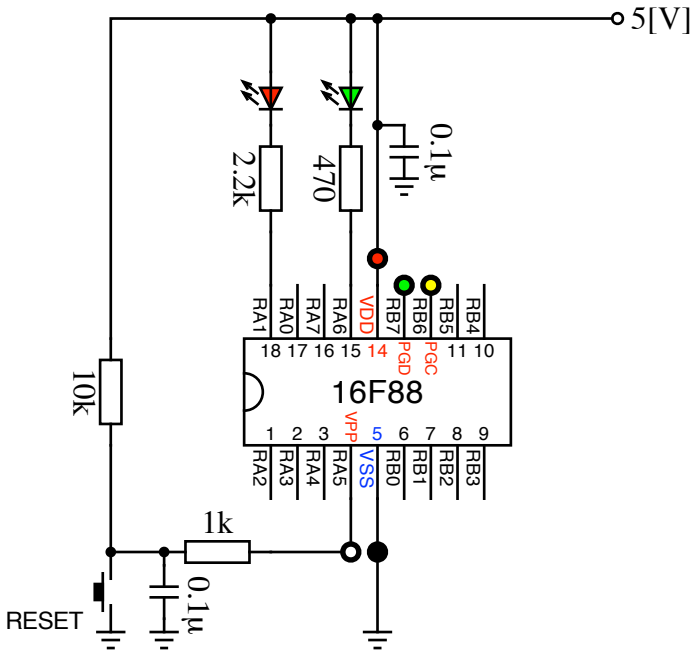


図1.6　Hello, PIC回路図

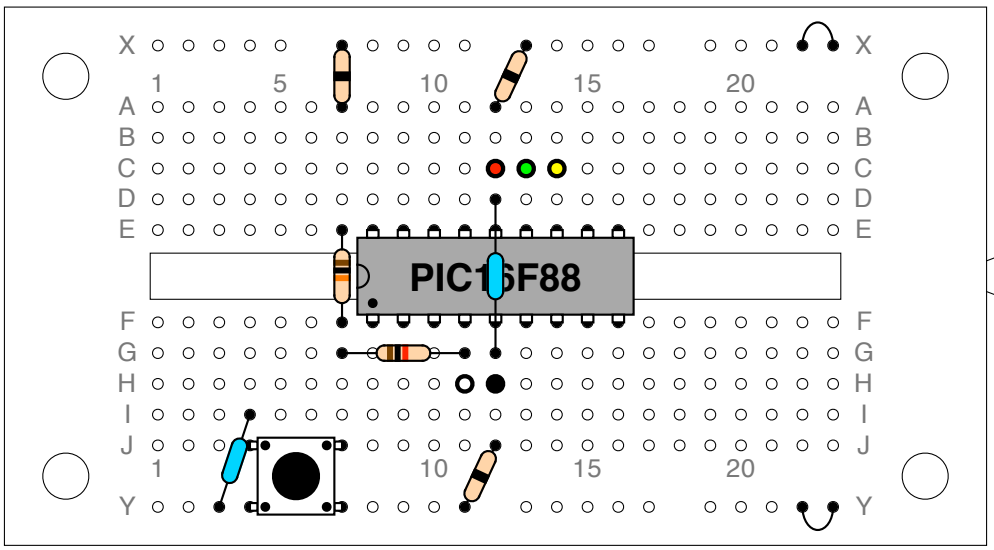


図1.7　実習用ブレッドボード（中央部のみ）

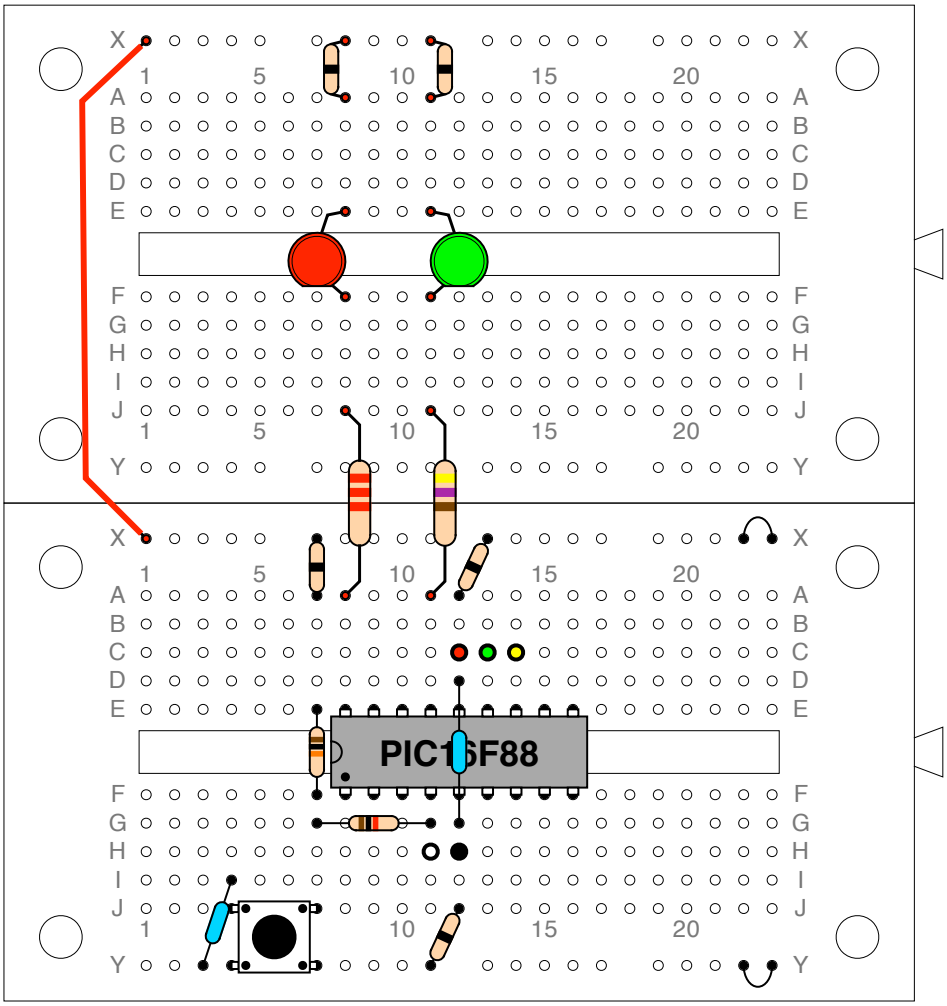


図1.8　実習ボード実体配線図

プログラムの基本的な考え方としては
手順1：RA1=0，RA6=1として赤LEDだけを点灯させる．
手順2：ある時間（例えば1秒）待つ．
手順3：RA1=1，RA6=0として緑LEDだけを点灯させる．
手順4：ある時間（例えば1秒）待つ．
手順5：手順1に戻る．
でよい．しかし，ここでは次のようにする；

手順1’：PORTA = 0xf0 とする（赤LEDが点灯，緑LEDは消灯）．
手順2’：ある時間（例えば1秒）待つ．
手順3’：PORTAと0xffの排他的論理和をとり，各ビットの値を反転する．
手順4’：手順2’に戻る．

GCGBによる上記手順のプログラミングは次のとおりである；
1. GCGBを起動する．
2. ProgramメニューからHardware Settings...を選択する．
   (1) Chip Modelは16F88，Speed (MHz)は8.000とする．
   (2) Chip ConfigurationはMCLRをONに，OSCをINTRC_IOに設定する（その他はDefault）．
   (3) Device Settingsは何も触らない．
   (4) OKボタンをクリックする．
3. IconsのCategoryからVariablesを選択する．

(1) Set Variableのアイコンをプログラムウィンドウ上にD&D する．
(2) Icon SettingsのVariableにPORTA，New Valueに0xf0を入力する．

4. IconsのCategoryからFlow Controlを選択する（起動段階でそうなっているはず）．
   (1) Doのアイコンをプログラムウィンドウ上にD&Dする．
   (2) Icon SettingsのModeはそのまま（Forever）．
5. IconsのCategoryからFlow Controlを選択する．
   (1) Waitのアイコンをプログラムウィンドウ上のDo ForeverとLoopのアイコンの間にD&Dする．
   (2) Icon SettingsのLengthに1を入力し，Unitsはs（秒）を選択する．
6. IconsのCategoryからVariablesを選択する．
   (1) Set Variableのアイコンをプログラムウィンドウ上のWait 1 sのアイコンの下にD&D する．
   (2) Icon SettingsのVariableにPORTA，New ValueにPORTA xor 0xffを入力する．
7. FileメニューからSave　Asを選択し，適当な場所（例えばHelloPICフォルダを作ってその中に）にHelloPICという名前で保存する（拡張子“.gcb”が自動的に付加される）． □

GCGBのViewメニューからView as Textを選択すると，リスト1.1のようなプログラムが表示されるはずである．同一でない場合は上記の手順のどこかで誤っているので，View as Iconsに戻して該当箇所を修正するか，あるいはテキスト表示状態で直接該当箇所を修正する．

1.5 実行コードの作成と書き込み

ソースコードが完成したら，実行コード（hexファイル）を作成してマイコンに書き込む．ソースコードから実行コードを作成する作業を「コンパイル」と呼ぶ．

GCGBのToolメニューから“Compile”を選択すると新しい窓（いわゆるDOS窓）が開き，コンパイルの進行状況を表示する．作業が終了すればDOS窓は自動的に消え，成功した場合はソースコードと同一階層に“HelloPIC.hex”というファイルができる．これが実行コードである． 一方コンパイル中にエラーを発生したり何らかの警告が発生した場合は，そのメッセージを表示する窓が残るので，内容を確認して所要の修正を行う．

リスト1.1　Hello, PICプログラム

```
;Chip Settings
#chip 16F88,8
#config MCLR=ON, OSC=INTRC_IO

PORTA = 0xf0
Do Forever
    Wait 1 s
    PORTA = PORTA xor 0xff
Loop
```

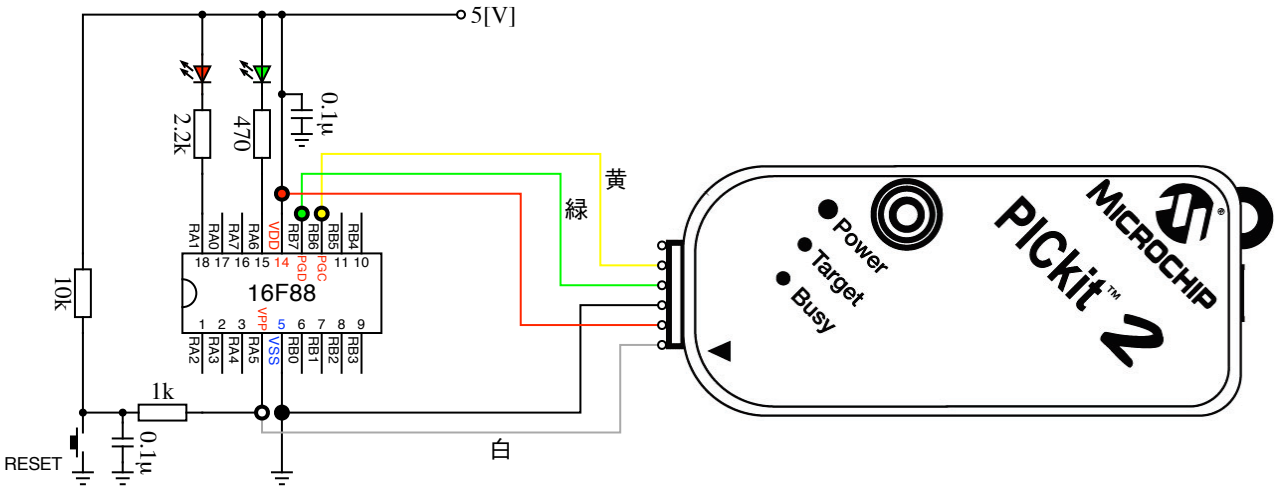


図1.9　回路図及びPICkit2の接続要領

実行コードができたら，これをPICマイコンに書き込む．書き込みの手順は以下のとおりである；

1. 図1.9のようにマイコン回路とPICkit2を接続し，PICkit2とPCをUSBケーブルで接続する．
2. PICkit2のアイコンをダブルクリックして起動する．接続が正常であれば，状態ウィンドウに“PICkit2 found and connected. PIC Device Found.”と表示される．
3. メインウィンドウの“VDD PICkit2”（会社ロゴマークの下にある）にある“On”のチェックボックスにチェックをつけ，その右側の窓の数値を5.0にする（マイコン回路への電源をPCから供給する）．
4. “File”メニューから“Import Hex”を選択し，書き込むべきhexファイル(HelloPIC.hex)を選択する．
5. “Write”ボタンをクリックしてhexファイルをPICマイコンに書き込む．正常に書き込めれば，自動的にプログラムが動作する．

1.5 研究課題

歩行者用信号のプログラムを作成して実行せよ．提出物はプログラムリストの印刷出力とする．

実習2：ルーレット

2.1 この実習で学ぶこと

・PICのI/Oポートの使い方（デジタル入力）
・割り込み

2.2 プログラムと回路の概要

このプログラムは先に実習したHelloPICプログラムをベースに，スイッチによるLED点滅の停止機構を加えたものである．

回路及びブレッドボードへの実装を図2.1及び2.2に示す．Hello PICで製作した回路に点滅停止スイッチを追加し，これをポートBの4番に接続する．図2.1及び2.2に基づいて，ブレッドボード上に回路を実装する．

プログラムのソースコードをリスト2.1に示す．HelloPICとの主な違いはDo〜Loopの終了判断にある．HelloPICでは無限ループになっていたが，rouletteではポートBの4番端子（PORTB.4）の値が1である間だけに限定されている．すなわち，ポートBの4番端子の値が0になる（停止スイッチが押される）とDo〜Loopを終了する．その次の処理はただの無限ループであり，LEDの点滅状態を変更することはないので，LEDの点滅が停止する．

リスト2.1　rouletteプログラム（その1：単純版）

```
;Chip Settings
#chip 16F88,8
#config MCLR=ON, OSC=INTRC_IO

TRISB = 0x10
PORTA = 0xf0
PORTB = 0xff
Do While PORTB.4 = 1
    PORTA = PORTA xor 0xff
    Wait 5 ms
Loop
Do Forever
Loop
```

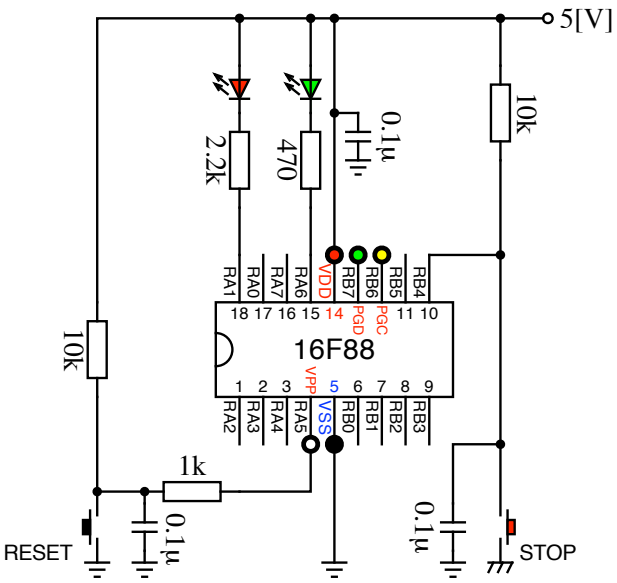


図2.1　ルーレット回路図及びPICkit2接続

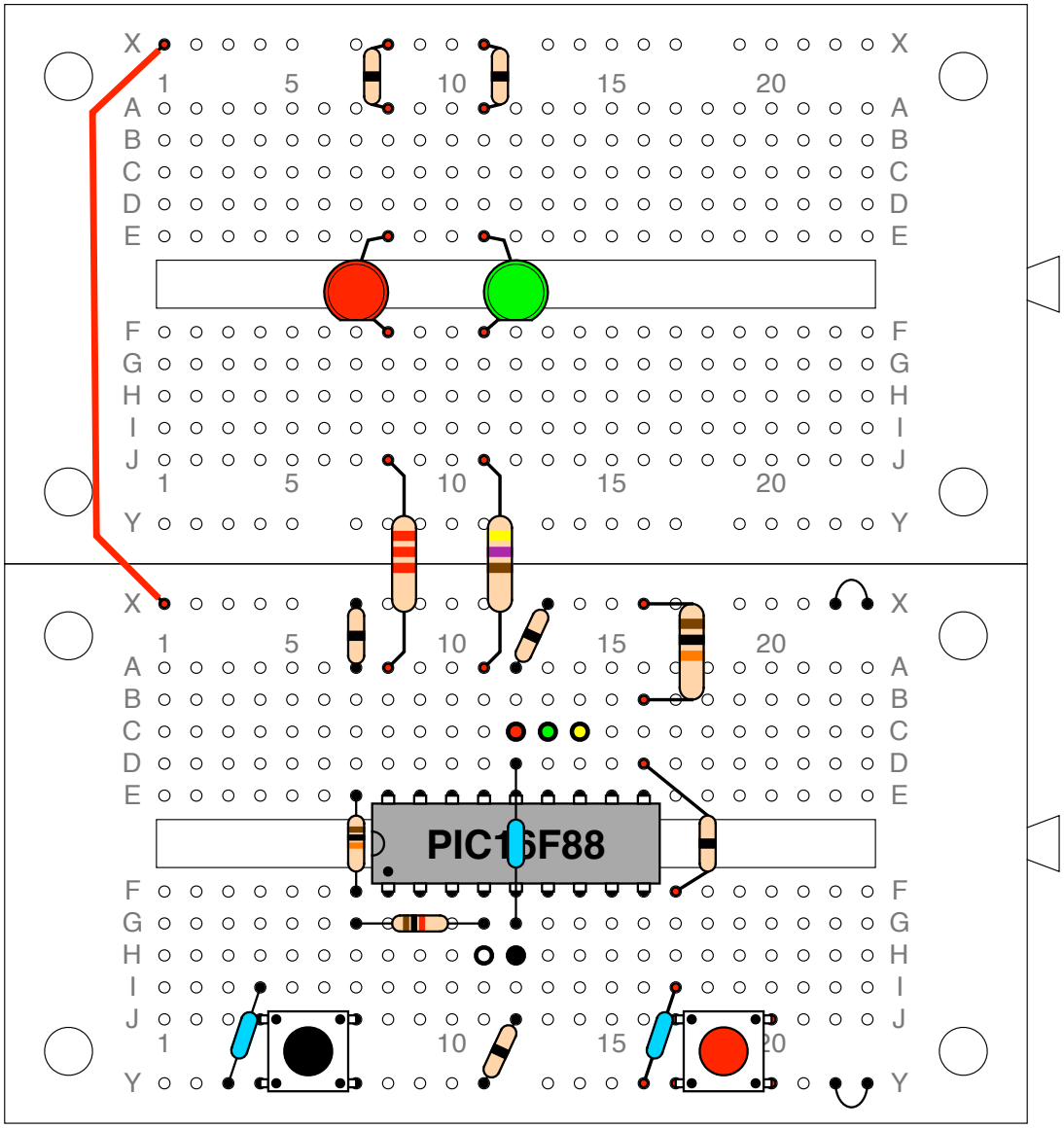


図2.2　ルーレット実体配線図

2.3 このプログラムの致命的欠陥と対策

このプログラムには致命的な欠陥がが一箇所存在する．
ソースコード上の時間待ちを5ミリ秒から1秒に変更；

```
Wait 5 ms  →  Wait 1 s
```

して動作させてみよう．こうするとほとんどの場合，スイッチを押してもLEDの交互点滅は停止しない．停止するのはストップスイッチを長押して，LEDの点灯状態が変化した時に限られる．

これはソースコードにおいて，LED点滅のループを抜ける判断が “Do While”の時に行われるためで，その時以外はスイッチの状態を見ていないからである．

このため，スイッチの状態を常時監視するようなプログラムでは，「割込み」という手法を用いてこのような欠点を克服する．割込みは図2.3に示すように，ある特別な事象が生じたら，それまでの処理を一時中断して「割込処理」と呼ばれる特別な処理にプログラムの動作を移行する．そして割込処理を終了したら，中断箇所に復帰するものである．

割り込みを用いたルーレットのプログラムをリスト2.2に示す．

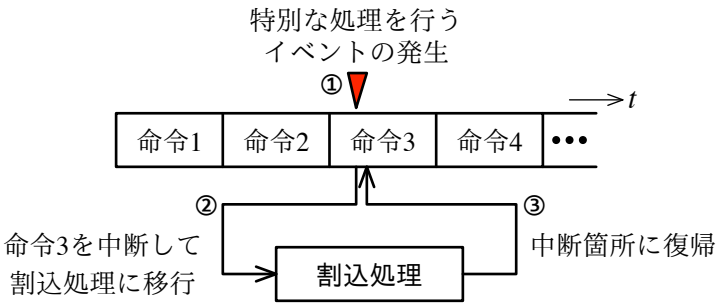


図2.3　割込みの概念

リスト2.2　rouletteプログラム（その2：割り込み版）

```
;Chip Settings
#chip 16F88,8
#config MCLR=ON, OSC=INTRC_IO

;Variables
Dim fstop As bit

fstop = 0
TRISB = 0x10
PORTA = 0xf0
PORTB = 0xff
INTCON = 0x88
Do While fstop = 0
    PORTA = PORTA xor 0xff
    Wait 1 s
Loop
Do Forever
Loop

Sub Interrupt
    GIE = 0
    If RBIF = 1 Then
           fstop = 1
           RBIF = 0
    End If
    GIE = 1
End Sub
```

`Dim fstop As bit`
は，*fstop*という変数を「ビット型」として定義していることを示す．TBWと異なり，GCGBでは使用する変数は全てDim文を使って「名前」と「型」を定義しなければならない．

GCGBで利用可能な変数の型は
・ビット型（Bit）　：0または1
・バイト型（Byte）　：0～255までの整数
・ワード型（Word）　：0～65535までの整数
・アレイ型（Array）　：0～255までのサイズを規定されたバイト型の配列変数
の4種類である．

```
INTCON = 0x88
```
は割込み制御レジスタの設定であり，「ポートBの4〜7番ピンの入力値が変化したら割込みをかける」という意味である．

`Sub Interrupt` 〜 `End Sub` までが割込み処理部である．処理の内容は以下のとおりである；

(1) GIE = 0 で割込みを受け付けないようにする．これは割込処理中に新たな割込みが入ることで処理全体が正常な動作から逸脱するのを防ぐためである．
(2) RBIF はポートBの4〜7番ピンに変化があったかどうかを示すレジスタ変数で，1の時に「変化があった」ことを示す．ここでは変化があった場合に *fstop* を1に設定し，RBIF の値を0に戻しておく．
(3) GIE = 1 で割込みを受付けるようにして，割込み処理を終了する．

2.4 研究課題

LEDの数を8つ（円周上に等角度で配置）にしたルーレットプログラムを作成せよ．LEDはポートA及びボートBのそれぞれ下位4ビット（RA0〜RA3及びRB0〜RB3）に接続するものとする．

実習3：キャラクタ液晶表示器（LCD）の使用

3.1 この実習で学ぶこと

・キャラクタ液晶表示器の使い方

3.2 液晶表示器

LCDはテレビやコンピュータの表示装置としてはもちろん，携帯電話や各種家電機器，自動車等，身の回りにある様々な機器において用いられている．LCDは液晶パネル及びそれを制御する回路（この制御回路も小さなコンピュータである）から構成されている．

使用するLCDモジュールはDB0～DB7の7本のデータ線，RS・R/W・Enableの3本の制御信号線，コントラスト調整端子及び電源の14本の端子が出ているが，この講義ではDB0～DB3を使用しない．またR/WはLCDに対してデータを送るか，LCDからデータを読み出すかの動作選択信号線であるが，今回はLCDに対してデータを送り出す（文字表示する）だけなのでGND線に接続している．

3.3 回路の概要

このプログラムはPICマイコンからキャラクタ液晶表示器（Liquid Crystal Display; LCD）を制御して文字を表示するものである．

回路を図3.1に示す．キャラクタLCDはサブ基板の上に実装されており，ここから6本の信号線と2本の電源線（VccとGND）でPICマイコンと接続される．サブ基板上に搭載されている青い部品は液晶表示のコントラストを調整する半固定抵抗である．

PICマイコンのRB4～RB7をLCDのDB4～DB7に割り当て，RB2をLCDのRSに，RB3をLCDのEnableに割り当てている．この接続関係はプログラム上に記述（または“Hardware Settings...”で設定）して，GCGBに対して指示する必要がある．

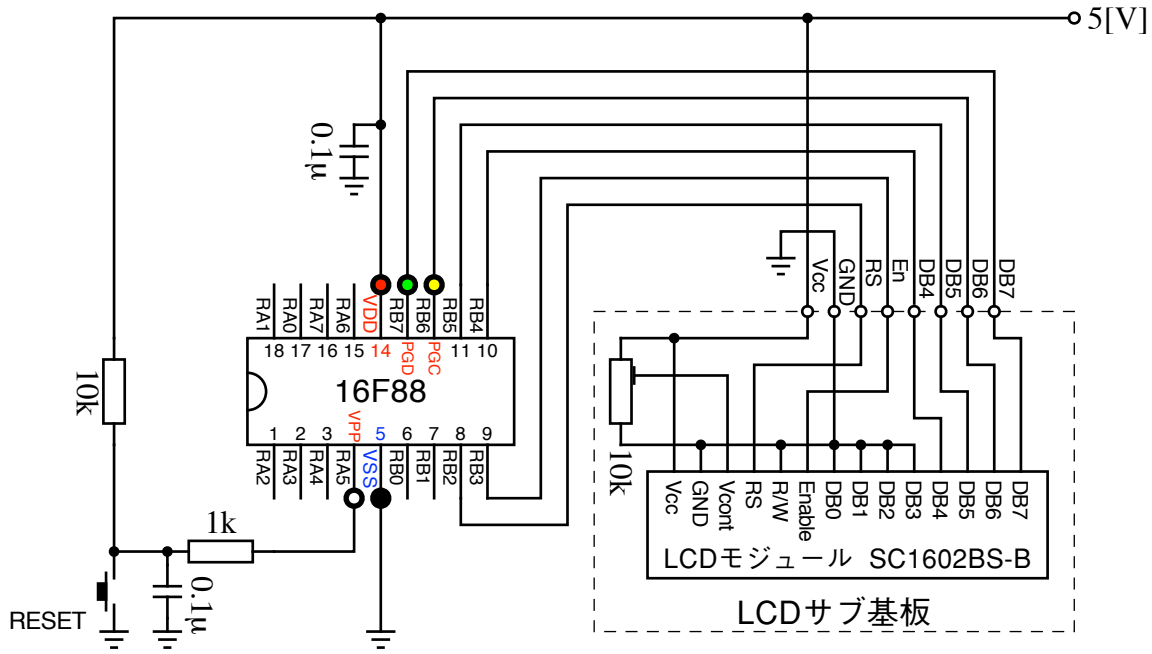


図3.1　Hello LCD回路図

リスト3　Hello LCDプログラム

```
;Chip Settings
#chip 16F88,8
#config MCLR=ON, OSC=INTRC_IO

;Defines (Constants)
#define LCD_IO 4
#define LCD_RS PORTB.2
#define LCD_Enable PORTB.3
#define LCD_NO_RW
#define LCD_DB4 PORTB.4
#define LCD_DB5 PORTB.5
#define LCD_DB6 PORTB.6
#define LCD_DB7 PORTB.7

Do Forever
   Print "Hello, world."
   Wait 1 s
   Locate 1, 0
   Print "de GCGB on LCD"
   Wait 1 s
   CLS
   Wait 1s
Loop
```

3.4 プログラムの動作

リスト3に示したプログラムは次のように動作する；

(1) #define *foo goo* は，*foo*という名の定数を*goo*と定義する命令である．

(2) LCD_IO は，LCDのデータ線本数を規定する定数である．ここでは4を設定する．

(3) LCD_RS はLCDのレジスタ選択線を送出するPIC出力端子を規定する定数である．

(4) LCD_DB4 ～ LCD_DB7 は，LCDへデータを送出するPIC出力端子を規定する定数である．

(5) LCD_IO，LCD_RS，LCD_DB4 ～ LCD_DB7 は，GCGBのHardware Settings...メニューで設定する項目である．

(6) LCD_NO_RW は，LCDのR/W制御線を使用しないという宣言である．この文はGCGBのHardware Settingメニューから設定できないので，Program → Constants...メニューから設定するか，GCGBをView as Textモードにして手入力する必要がある．

(7) Print "*foo*" は，LCD上の現在のカーソル位置から文字列*foo*を表示する．

(8) Locate *row*, *column* は，*row*行 *column*文字目にカーソルを移動する（始点は 0, 0）．

(9) Do Forever ～ Loop の無限ループがプログラムの実体部分である．1行目を表示した後に1秒待ち，次に2行目を表示して1秒間待つ．最後にLCD表示を消去して，さらに1秒間待つ．これを永遠に繰り返している．

本来であればLCDモジュール内の制御回路がどのような動作をしているのか規格表を参照した上でプログラムを作成する必要があるが，GCGBではその手順を内蔵しているためプログラマが意識する必要はない．

3.5 回路の構成

図3.1の回路図を図3.2のように配線する．LCDサブ基板をブレッドボードに挿入する際は，最下段ブレッドボードのA行が見えるようにする．

3.6 研究課題

LCDの2行目に自分の名前が一文字ずつ表示されるようなプログラムを作成せよ.

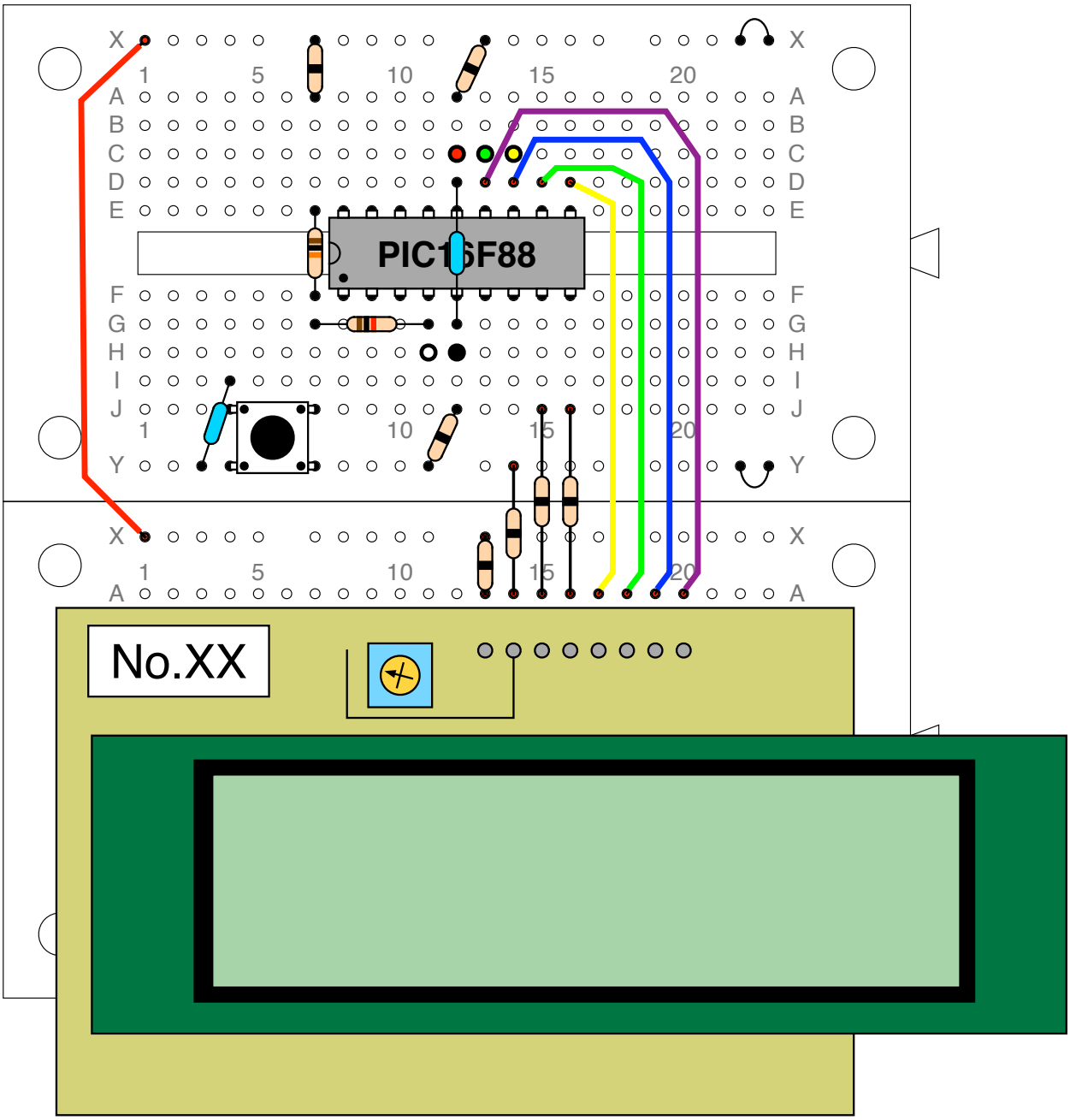


図3.2　Hello LCD実体配線図

実習4：A/D変換の利用―ディジタル温度計―

4.1 この実習で学ぶこと

・温度センサの概要及び使い方
・A/D変換の利用方法

4.2 温度センサ

温度センサは数あるセンサの中でも最も広く用いられているセンサの一つである．種類としては熱電対，サーミスタ，半導体等があり，本実習では半導体温度センサを用いる．

半導体温度センサはその出力電圧 V_o が絶対温度に比例するようにトランジスタ回路を構成した集積回路で，セルシウス温度，温度係数及びオフセット電圧をそれぞれt, T_c及びV_{offset}とすると

$$V_o = T_c t + V_{offset} \tag{4.1}$$

で示される．本実習で用いるMCP9701–Eでは，T_c及びV_{offset}はそれぞれ19.5[mV/°C]，400[mV]である．

4.3 A/D変換

A/D (Analog to Digital) 変換は，アナログ値をディジタル値に変換する機能のことをいう．PICをはじめとする1チップマイコンではこのA/D変換機能を持つものが多く，各種センサで取得したアナログ値のデータをディジタル値に変換して処理できるようにしている．

A/D変換において最も重要な要素は分解能であり，通常，NビットA/Dという形で示される．これは入力可能な最大電圧を2^Nで割った値となる．例えば10ビットA/Dで最大入力電圧5[V]の場合，分解能は5/1024=4.8828[mV]となり，これ以下の電圧は検出・測定できないし，測定値も4.8828[mV]おきの値となる．この分解能は，今回用いる温度センサでは約0.25[°C]に相当する．A/D変換された測定値は，以後，0～2^N–1の整数値（10ビットであれば，0～1023）として取り扱われる．

4.4 回路の概要

本実習回路は前回ＬＣＤへの文字表示を実習した回路に温度センサを追加（温度センサ出力をRA0に接続）しただけであり，これを図4.1に示す．

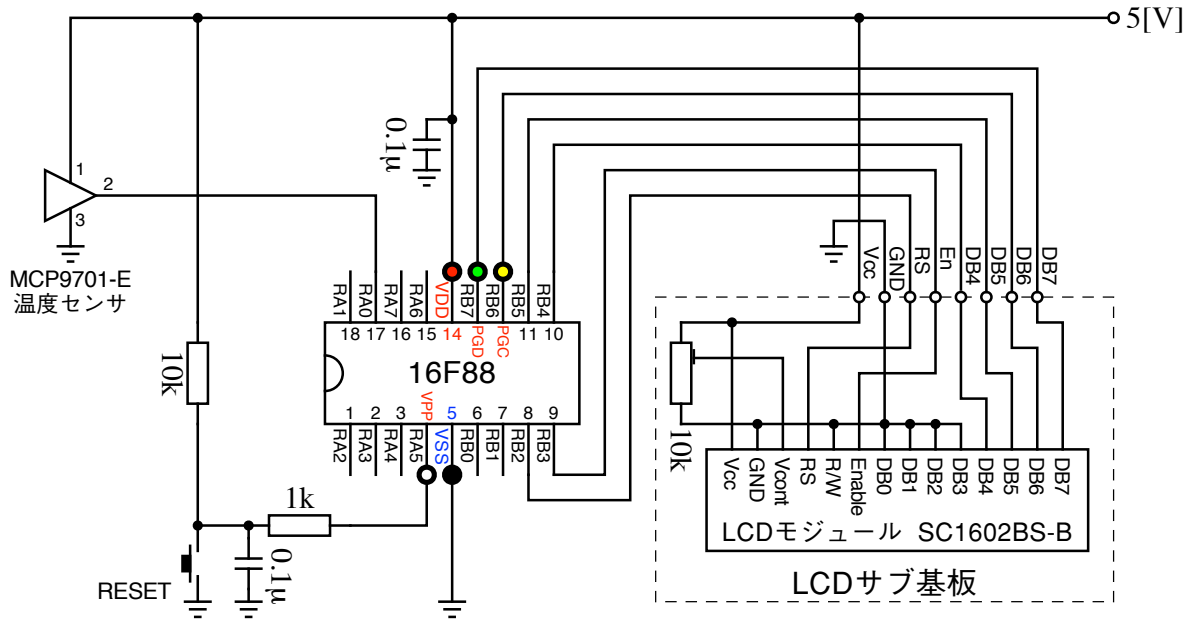


図4.1　ディジタル温度計回路図

リスト4　ディジタル温度計プログラム

```
;Chip Settings
#chip 16F88,8
#config MCLR=ON, OSC=INTRC_IO

;Defines (Constants)
#define LCD_IO 4
#define LCD_RS PORTB.2
#define LCD_Enable PORTB.3
#define LCD_NO_RW
#define LCD_DB4 PORTB.4
#define LCD_DB5 PORTB.5
#define LCD_DB6 PORTB.6
#define LCD_DB7 PORTB.7
#define degree 0xdf

;Variables
Dim ADvalue As word
Dim t100 As word
Dim tint As byte
Dim tflt As byte

Dir PORTA.0 in
Do Forever
    ADvalue = ReadAD10(AN0)
    CLS
    Print "A/D = "
    LCDWord ADvalue
    Locate 1, 0
    Print "t = "
    t100 = ADvalue * 25 - 2050
    tint = t100 / 100
    tflt = t100 - tint * 100
    LCDInt tint
    Print "."
    LCDInt tflt
    Locate 1, 9
    Print "[ C]"
    Put 1, 10, degree
    Wait 1 s
Loop
```

4.5 プログラムの説明

本プログラムは1秒ごとに温度を計測し，LCDの1行目にA/D変換された数値を，2行目に温度を表示する．リスト4にプログラムのソースを示す．

(1) LCD関係の定数定義までは従前のHelloLCDプログラムと同じである．

(2) #define *degree* 0xdf は，温度単位の「°」がキーボードから明示的に入力できる文字ではないので，LCDのデータシートに記載されている文字コードで表すための定数として定義している．

(3) ;Variables の欄では，PICマイコン自身のレジスタ等ではなく，ユーザがプログラム内で独自に使用する変数を定義する．Program→Variables...メニューから定義する．ADvalueはA/D変換した値を格納する変数（Word型），t100は気温の100倍の値を格納する変数（Word型），tintは

気温の整数部を格納する変数（Byte型），tfltは気温の小数部を格納する変数（Byte型）である．

(4) Dir PORTA.0 in は，PORTAの0番ピンを入力モードで使用するという設定である（TRISA = 0x01と同じ）．

(5) *ADvalue* = ReadAD10(AN0) は，AN0端子（RA0）のアナログ電圧を10ビット精度でA/D変換して変数ADvalueに格納する．

(6) LCDWord *ADvalue* は，LCDの現在のカーソル位置（この時点では何も指定していないので，初期値の$0,0$となる）からワード型変数*ADvalue*の値を表示する．

(7) Locate x, y はLCDのカーソル位置をx行 y桁に移動する．始点は$(0,0)$である．

(8) 温度の計算は以下の要領で算出している；
MCP9701E-1の出力電圧は

$$V_o[\mathrm{mV}] = 19.5t + 400 \tag{4.2}$$

で与えられる．5[V]電源，10bit精度のA/D変換で得た値をnとすれば，

$$\frac{5000}{1024}n = 19.5t + 400 \tag{4.3}$$

となる．式(4.3)より，温度 t の100倍の値を導出すると

$$100t \approx 25n - 2050 \tag{4.4}$$

を得る．

(9) LCDInt *tint* は，現在のカーソル位置からByte型変数 *tint* の値を表示する．

(10) Put 1, 10, *degree* は，カーソル位置(1,10)（すなわち2行目11桁目）にキャラクタ文字*degree*を表示する．

A/D変換については，本来はPICマイコンの内部動作を考慮してプログラムする必要があるが，GCGBではそれを意識せずにReadAD()（もしくはReadAD10()）命令一つで実行することができる．ただし雑音の影響を考慮しなければならないA/D変換では，別途アセンブリ言語による検討が必要である．

4.6 回路の構成

図4.1の回路図を図4.2のように配線する．実習3で構成したものに，温度センサと関連する配線を追加するのみである．

4.7 研究課題

(1) リスト4のプログラムでは，気温が氷点下になった時に正確な値を表示しない．氷点下でも正確に動作させるためには，プログラムリストをどのように変更すればよいかを考えよ．

【注】GCGBでの整数は負の数を取り扱うことができないので，Byte型では「–1」としたつもりでも255ととして取り扱われる（Word型では65535）．

(2) 標本化定理について調べ，これよりA/D変換を行うときに留意すべき事項を列挙せよ．

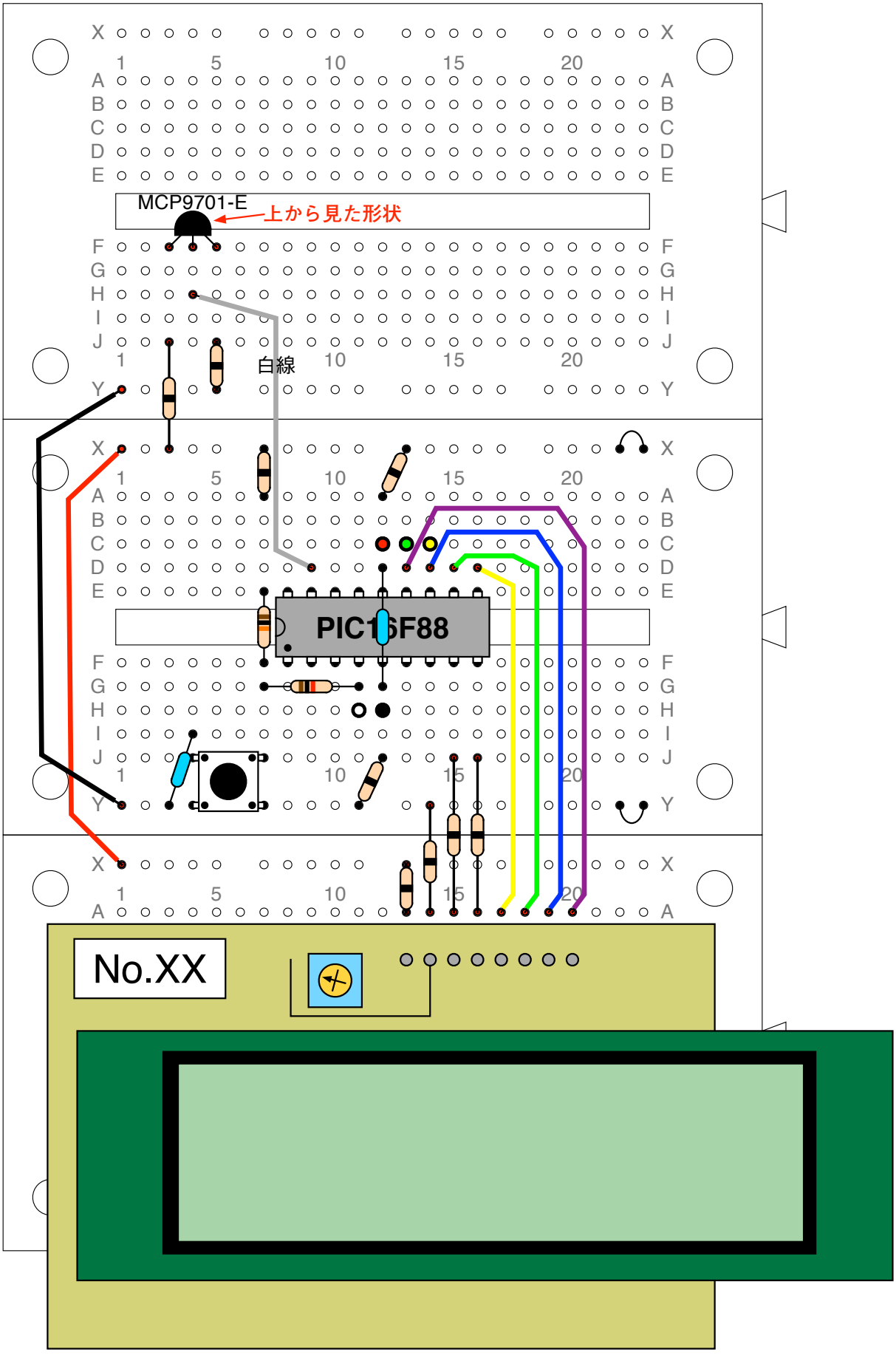


図4.2　ディジタル温度計実体配線図