MPLAB XC8によるダイナミック点灯プログラムの作成例

山﨑　勝翁

技術部

PICプログラムをC言語で作成する場合，有料と無料のコンパイラの選択肢がある．有料の場合は導入コストが必要である．本研究では，無料のPIC用MPLAB XC8コンパイラを利用して，アセンブラより効率よくプログラム作成可能なC言語に注目し，プログラムの作成を行った．C言語は高水準言語と呼ばれ，人にやさしい言語として期待される．文字データは，2次元配列の利用が適切で，繰返しの多重ループで取り出すと効率的である．確実性を増すため配列の初期化と繰返しをprintf関数により出力し検証し，printf関数をPICのPORTA，PORTBへの出力に変更すると文字表示が可能なことを確認したので報告する．

キーワード：MPLAB XC8コンパイラ，2次元配列，繰返し，多重ループ

１．はじめに

この報告はPIC16F648Aによる文字表示回路の製作[1)]の続編である．ここでは従来からの変更点とMPLAB XC8コンパイラによるCプログラムの1つの記述例について行う．現環境は，回路ではドットマトリクスLEDがカソードコモンに変更され，Microchip technology社の統合環境MPLAB XC8 IDEのXC8コンパイラの設定である．このPIC用C言語はANSI規格準拠であるが，プログラムを動作させるにはPIC用の特殊な記述を理解する必要がある．アセンブラは記述が長いので，繰返しが多いダイナミック点灯プログラムをC言語でとの要望があり，それに応えるため検証した．

最初にBorland C++CompilerやVisualStudio2012コンソールアプリケーション(以下VC++)を利用して基礎的なCプログラム[2)~4)]を作成し配列の確認を行った．4文字表示のデータ化の際，ポートBのデータを表にまとめ7行4列で作成したが，正しくは4行7列の2次元配列が適切であることがわかった．検証はポートAの1次元配列とポートBの2次元配列の初期化が確実にされているか，繰返しに間違いがないか，よく使われるprintfの出力で確認した．XC8コンパイラによるCプログラムではprintf関数を各ポートへの出力に変更することでPICの各ピンからの出力が制御可能となる．遅延時間(ウェイト)のdelay関数[5)]は簡単に使用できるのでその記述にも触れる．最後に低水準言語のアセンブラと高水準言語のC言語プログラムでのソースファイルや生成された実行型のhexファイルについて比較を行った．アセンブラとC言語でそれぞれの特徴や利点があることが分かったので最後に考察する．

２．方法

(1) ソフトウェア

Windows7　Enterprise　64ビット
Windows7　professional　32ビット
WindowsXP professional　32ビット
PICC Lite　v9.80　v9.81　XC8 v1.01
MPLAB IDE　v8.63　MPLAB X　IDE v1.30

(2) ハードウェア

PIC16F648A,PIC18F628A
ドットマトリクスLED　OSL641501-ARA
抵抗ネットワーク 220Ω　BI N3-221

回路の配線では，ポートAのRA4ピンがオープン・ドレインのために抵抗を介して電源と接続した．バイパスコンデンサ0.1μFを取り付け，不要なノイズのカット，電源電圧の安定を図った．図-1は作成例，図-2は回路図である．

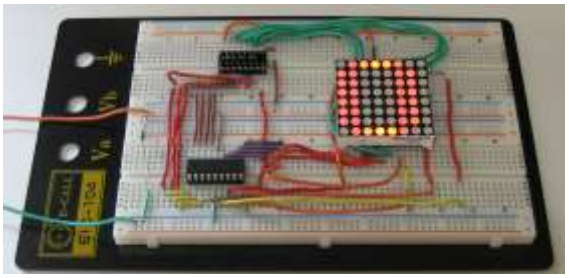


図-1　作成例（回路）

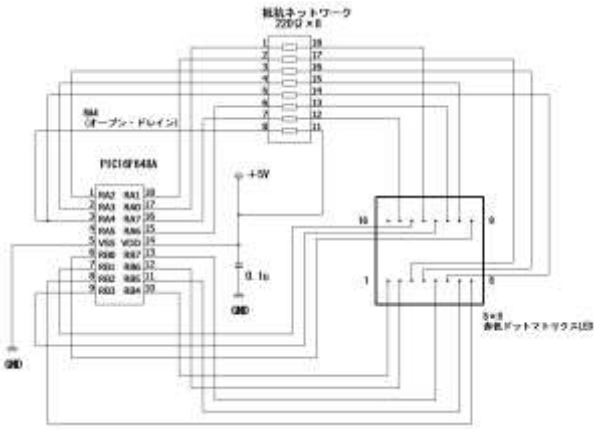


図-2　回路図

３．配列と繰返し

この章ではBorland C++CompilerやVC++を使用した基本プログラムで，配列の初期化と繰返しが確実に実行されているか確認するプログラムについて記述する．

リスト-1は１次元配列と２次元配列の初期化例である．**リスト-2**はprintfの出力例，**リスト-3**はfor文による繰返し例で**図-3**，**図-4**が実行結果である．a[0]～a[6]はa[列番号]，b[0][0]～b[3][6]はb[行番号][列番号]を表している．

実行結果から配列要素のとおりにデータが初期化されていることがわかる．配列aはポートAを，配列bはポートBを想定したものである．変数は行がi，列がj，繰り返し回数をnとした．列を示すjを配列aと配列bに使うのがポイントである．**リスト-3**はfor文による3重ループで多重ループと呼ばれ，以下のような処理が行われる．

①iが0のとき，
②nの残り繰返し5回表示のとき，　jを0から6まで増やし
③配列a[0]からa[6]をprintfで出力
④配列b[0][0]からb[0][6]をprintfで出力
繰り返しのため省略して
nの残り繰返し4回から1回まで②③④を繰り返す
iが1のときからiが3のときで①②③④を繰り返す

リスト-1　配列の初期化例

```
#include <stdio.h>

   int a[7] = {1,2,4,8,10,40,80};
   int b[4][7] = {
            {00,01,02,03,04,05,06} ,
            {10,11,12,13,14,15,16} ,
            {20,21,22,23,24,25,26} ,
            {30,31,32,33,34,35,36} ,
     };
```

リスト-2　printfの出力例（b[1][0]～b[3][6]省略）

```
int main(void)
{
  printf("¥n¥n配列要素とデータ格納の確認¥n");
  printf("配列a¥n");
  printf("%2d %2d %2d %2d %2d %2d %2d¥n¥n",a[0],a[1],a[2
],a[3],a[4],a[5],a[6]);

  printf("配列b¥n");
  printf("%2d %2d %2d %2d %2d %2d %2d¥n",b[0][0],b[0][1],
b[0][2],b[0][3],b[0][4],b[0][5],b[0][6]);
}
```

表-1　配列a

1	2	4	8	10	40	80

表-2　配列b

00	01	02	03	04	05	06
10	11	12	13	14	15	16
20	21	22	23	24	25	26
30	31	32	33	34	35	36

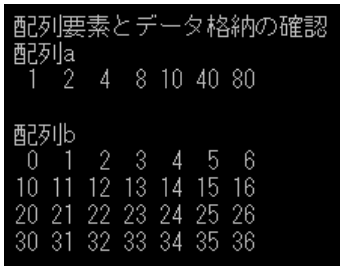


図-3　リスト-1 ＋ リスト-2の実行結果

リスト-3　for文による繰返し例

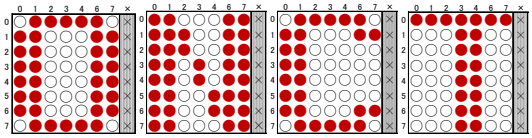
```
int main(void)
{
   int i,j,n;
   for (i= 0; i < 4; i++) {
      for (n = 5 ; n > 0 ; n--) {
         printf("¥n残り繰返し = %d回¥n", n);
            for(j = 0 ;j < 7; j++) {
               printf("a[%2d] = %2d → ",j,a[j]);
               printf("   b[%2d][%2d] = %2d¥n", i,j,b[i][j]);
         }
      }
   }
   return 0;
}
```

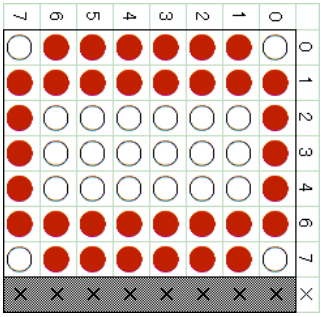
```
残り繰返し = 5回
a[ 0] =  1 →   b[ 0][ 0] =  0
a[ 1] =  2 →   b[ 0][ 1] =  1
a[ 2] =  4 →   b[ 0][ 2] =  2
a[ 3] =  8 →   b[ 0][ 3] =  3
a[ 4] = 10 →   b[ 0][ 4] =  4
a[ 5] = 40 →   b[ 0][ 5] =  5
a[ 6] = 80 →   b[ 0][ 6] =  6

残り繰返し = 4回
a[ 0] =  1 →   b[ 0][ 0] =  0
a[ 1] =  2 →   b[ 0][ 1] =  1
a[ 2] =  4 →   b[ 0][ 2] =  2
a[ 3] =  8 →   b[ 0][ 3] =  3
a[ 4] = 10 →   b[ 0][ 4] =  4
a[ 5] = 40 →   b[ 0][ 5] =  5
a[ 6] = 80 →   b[ 0][ 6] =  6
```

図-4　リスト-1 ＋ リスト-3の実行結果（一部省略）

4．文字のデータ化





2 進数		16 進数
10000001	→	0x81
00000000	→	0x00
01111110	→	0x7E
01111110	→	0x7E
01111110	→	0x7E
00000000	→	0x00
10000001	→	0x81

図-5　文字パターンのデータ化

図-5は文字パターンOのポートBのデータ化を示したものである．文字パターンは自作のExcelのテンプレートにより，赤ドットを点灯としてコピー&貼付して作成した．1文字のデータはポートBの接続からこれを右へ90°回転させ，以前とは論理が逆の消灯を「1」，点灯を「0」として8ビットの2進数で表し，その後16進数へ変換した．2進数では点灯状態がわかり，16進数では桁数が少なくなり間違いを起こしにくい利点もある．**表-3**配列aはポートAのデータ，**表-4**はその配列要素,**表-5** 配列bはポートBのデータ、**表-6**はその配列要素である．

表-3 配列a

0x01	0x02	0x04	0x08	0x10	0x40	0x80

表-4 配列要素

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]

表-5 配列b

0x81	0x00	0x7E	0x7E	0x7E	0x00	0x81
0x00	0x00	0xF9	0xE7	0x9F	0x00	0x00
0x81	0x00	0x7E	0x7E	0x7E	0x3C	0xBD
0xFE	0xFE	0xFE	0x00	0x00	0xFE	0xFE

表-6 配列要素

b[0][0]	b[0][1]	b[0][2]	b[0][3]	b[0][4]	b[0][5]	b[0][6]
b[1][0]	b[1][1]	b[1][2]	b[1][3]	b[1][4]	b[1][5]	b[1][6]
b[2][0]	b[2][1]	b[2][2]	b[2][3]	b[2][4]	b[2][5]	b[2][6]
b[3][0]	b[3][1]	b[3][2]	b[3][3]	b[3][4]	b[3][5]	b[3][6]

5．XC8　C言語プログラム

XC8 C 言語プログラムの作成例を示す．XC8 コンパイラは，ANSI 準拠のため、普通のC言語の文法と同じように、演算，繰返し，配列、関数，ポインタ、構造体等を用いたプログラミングが可能である．MPLAB IDE で XC8 コンパイラを使う場合，C プログラムを拡張子.c で作成し，ProjectWizard で以下のように C コンパイラを使う設定にする．

Active Toolsuite：　Microchip XC8 ToolSuite
ToolsuiteContents：Microchip MPLAB XC8 Compiler

XC8コンパイラ用の記述[5)6)]についての説明である．

(1) 制御関数のヘッダファイル

レジスタやビットの名前が定義されている．
#include　“pic.h”

(2) 内部発振回路

4MHzの指定．#define _XTAL_FREQ 4000000

(3) コンフィグレーション・ビットの設定

PICC Lite v9.80を境に変更されているので2つの記述例を示す．

PICC Lite　v9.80まで
__CONFIG(INTIO & WDTDIS & PWRTDIS & BORDIS & LVPDIS & MCLRDIS);

PICC Lite　v9.81以降　XC8v1.01で確認済み
__CONFIG(FOSC_INTOSCIO & CP_OFF & WDTE_OFF & PWRTE_ON & BOREN_OFF & LVP_OFF & MCLRE_OFF);

(4) トライステート・ポート

このレジスタのビットを「1」にすると入力ピンになり，「0」にすると出力ピンになる．TRISA=0x00，TRISB=0x00

(5) 遅延時間

delay関数はpic.hファイルのBuilt-in delay routineに，ミリ秒単位とマイクロ秒単位が定義されている．先頭にアンダーバーを2個つけて1msの場合は__delay_ms(1);，1μsの場合は__delay_us(1);と記述する．数値を変更すると希望の遅延時間が得られる．

リスト-4が文字表示プログラム例である．無限ループwhile(1)でONCTと繰り返し連続表示させているため4重ループである．

リスト-4　文字表示プログラム例

```
#include "pic.h"                    //ヘッダファイル
#define _XTAL_FREQ 4000000          //内部発振回路 4MHz

//コンフィグレーション・ビットの設定
__CONFIG(FOSC_INTOSCIO & CP_OFF & WDTE_OFF &
PWRTE_ON & BOREN_OFF & LVP_OFF & MCLRE_OFF);

main(){
                            // PIC の初期化
    CMCON = 0x07;   // コンパレータ OFF
    TRISA = 0x00;       // PORTA すべてを出力設定
    TRISB = 0x00;       // PORTB すべてを出力設定

    int i,j,n;
    //ポート A データ
    int a[7]={0x01,0x02,0x04,0x08,0x10,0x40,0x80};
    //ポート B データ
    int b[4][7]={
            {0x81,0x00,0x7E,0x7E,0x7E,0x00,0x81},  //O
            {0x00,0x00,0xF9,0xE7,0x9F,0x00,0x00},   //N
            {0x81,0x00,0x7E,0x7E,0x7E,0x3C,0xBD}, //C
            {0xFE,0xFE,0xFE,0x00,0x00,0xFE,0xFE}, //T
    };
    while(1) {                      //無限ループ
        for(i=0;i<4;i++){           //行の移動：i
            for(n=143;n>0;n--){     //繰り返し回数:ｎ=143
                for(j=0;j<7;j++){   //列の移動:j
                    PORTA=a[j];     //ポート A へ出力
                    PORTB=b[i][j];  //ポート B へ出力
                    __delay_ms(1);  //1ms のウェイト
                }
            }
        }
    }
    return 0;
}
```

表-7　アセンブラとC言語の比較

	アセンブラ	C言語
ソースファイル	3531バイト	935バイト
文字数	453	148
実行型ファイル	1156バイト	2052バイト

６．考察

(1) 表-7が「ONCT」と文字表示をするアセンブラとC言語での違いを比較したものである．ソースファイルと文字数では，C言語の方が約1/3の記述で済み，実行型のファイルサイズは2倍ほど大きい．このことから，C言語は短時間でプログラム作成が可能といえる．

(2) ポートの入出力設定TRISA，TRISBは，バンクの切換えが自動で行われるので，値を設定するだけでよい．

(3) 遅延時間の記述では、C言語はdelay関数の1行で簡潔に済み、アセンブラでは，1サイクル実行時間から繰返しで遅延時間を作成するため，計算が必要となる．

(4) C言語は自作の関数や標準ライブラリ関数で利用方法を考え，アセンブラは命令を１つひとつ記述するため，詳細に仕組みを理解でき細かな設定には向いている．

(5) printfでの確認は有効な手段である．格納された配列データをfor文で繰り返し取り出す動作を把握でき,表示で確認できたので確実性が増した．

７．おわりに

本報告では，C言語のfor文の繰返しによる多重ループやPIC専用のdelay関数を使い、効率のよい単純明快なプログラムが完成した．C言語が人にやさしい高水準言語と呼ばれる理由が確かめられた．

参考文献

1) 山﨑勝翁，手島規博 : PIC16F648Aによる文字表示回路の製作，大分工業高等専門学校紀要　第45号
2) C言語ではじめるPICマイコン　ーフリーのCコンパイラではじめようー　中尾真治　オーム社
3) [新版] 明解C言語 入門編　柴田望洋　ソフトバンククリエイティブ
4) [新版] 明解C言語 中級編　柴田望洋　ソフトバンククリエイティブ
5) MICROCHIP HI-TECH C User's GUIDE
6) MPLAB XC8 C Compiler User's Guide

（2013.9.30受付）