

16 進表示用 7 セグメント LED デコーダの作製

中山 勝

筑波大学システム情報系技術室

〒305-8573 茨城県つくば市天王台 1-1-1

概要

情報科学類 2 年次に実施されている実験に、論理回路実験がある。この実験では、サンハヤト製の IC トレーナーを使用し、組み合わせ回路や順序回路について学習する。IC トレーナーには、表示器として 7 セグメント LED が実装されているが、使用されているデコーダが 10 進デコーダであるため、10～15 は数として表示されない。そこで、マイクロチップテクノロジー社製 PIC (Peripheral Interface Controller) をプログラミングすることで、16 進数を表示できるデコーダを作製したので報告する。

キーワード：PIC、7 セグメント LED、デコーダ

1. はじめに

IC トレーナーとは、ブレットボードの周りに、電源・トグルスイッチ・プッシュスイッチ・LED・7 セグメント LED 等を備えた、論理回路を学ぶための実習装置である(図 1)。

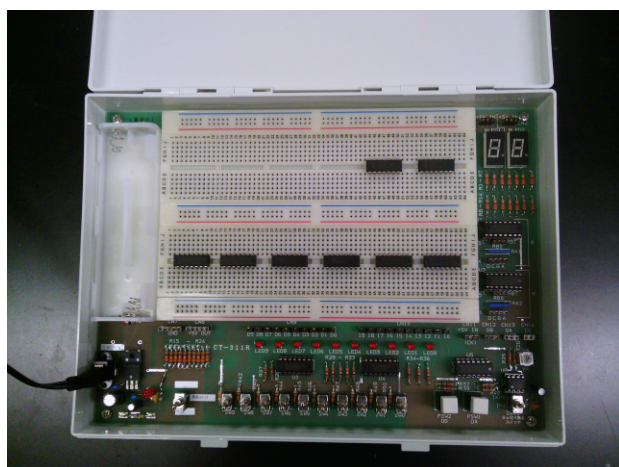


図 1. IC トレーナー

この装置を使用して学生が取り組む論理回路設計の中に、4 減算できる回路を持つ 4 bit バイナリカウンタがある。このとき、7 セグメント LED を表示器として使用するのだが、デコーダ IC が 74LS47(10 進デコーダ)であるため、その表示が図 2 の様になってしまう。この表示では、10～15 の数値は、非常に読み取りづらい。それを解消するために、10 以上の数値は、A(10) b(11) c(12) d(13) E(14) F(15)と表示できるデコーダを作製した。

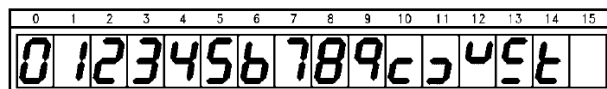


図 2. 74LS47 による 7 セグメント LED の表示

2. PIC とは

PIC とは、マイクロチップテクノロジー社が製造しているマイクロコントローラ(制御用 IC)製品群の総称である。開発環境としては、やはりマイクロチップテクノロジー社が提供している、MPLAB IDE X¹があり、アセンブラや C 言語で、プログラムを記述することができる。使用する PIC には、DIP 14 ピンパッケージの 8 bit マイコンである、PIC16F1823²を選択した。したがってコンパイラは、MPLAB[®] XC8 を使用した。

また、PIC にプログラムを書きこむためのライターには、PICkit3(図 3)を使用した。PICkit3 は、コンピュータと USB 接続し、PIC とはジャンパー線で接続する。



図 3. PICkit3

PICkit3 と PIC16F1823 の接続を、表 1 に示す。

表 1. 接続表

PICkit3	信号名	PIC16F1823
1	$\overline{\text{MCLR}}/\text{V}_{\text{pp}}$	4
2	V_{cc}	1
3	V_{ss}	14
4	PGD(ICSPDAT)	13
5	PGC(ICSPCLK)	12
6	未接続	-

¹ <https://www.microchip.com/mplab>

3. 74LS47 と PIC16F1823

74LS47 は、入力された 4 ビット 2 進数の値を、7 セグメント LED(図 4)の a から g の内、対応するセグメントを点灯させることで、数値として表示させる機能を持っている。

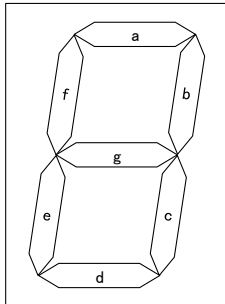


図 4. 7 セグメント LED

一方、PIC16F1823(図 5)は全 14 ピンの内、電源を除く 12 ピンをデジタル I/O として設定できる。そこで、RA3 - RA0(LSB)を入力、RA4、RC5 - RC0 を a - g に割り当ててことにした。

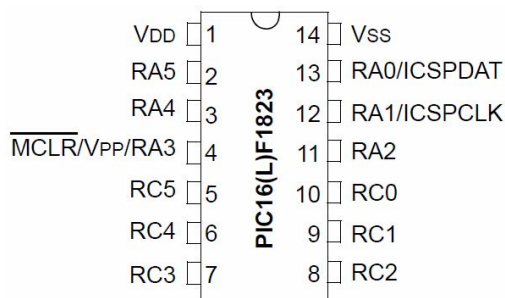


図 5. PIC16F1823 のピン配置

4. プログラミング

表 2 は、C 言語で記述した、制御用プログラムである。

表 2. 制御用プログラム

```
void main()
{
    ANSELA = 0;
    ANSELG = 0;

    TRISA = 0b00001111;
    TRISC = 0b00000000;

    PORTAbits.RA4 = 1;
    PORTAbits.RA5 = 0;    //unused
    PORTC = 0b11111111;

    while(1){
        switch(PORTA & 0x0F){
            case 0x00:
```

```
                PORTAbits.RA4 = 1;
                PORTC = 0b11000000;
                break;
            case 0x01:
                PORTAbits.RA4 = 1;
                PORTC = 0b11111001;
                break;
            case 0x02:
                PORTAbits.RA4 = 0;
                PORTC = 0b11100100;
                break;
            case 0x03:
                PORTAbits.RA4 = 0;
                PORTC = 0b11110000;
                break;
            case 0x04:
                PORTAbits.RA4 = 0;
                PORTC = 0b11011001;
                break;
            case 0x05:
                PORTAbits.RA4 = 0;
                PORTC = 0b11010010;
                break;
            case 0x06:
                PORTAbits.RA4 = 0;
                PORTC = 0b11000010;
                break;
            case 0x07:
                PORTAbits.RA4 = 1;
                PORTC = 0b11111000;
                break;
            case 0x08:
                PORTAbits.RA4 = 0;
                PORTC = 0b11000000;
                break;
            case 0x09:
                PORTAbits.RA4 = 0;
                PORTC = 0b11010000;
                break;
            case 0x0A:
                PORTAbits.RA4 = 0;
                PORTC = 0b11001000;
                break;
            case 0x0B:
                PORTAbits.RA4 = 0;
                PORTC = 0b11000011;
                break;
            case 0x0C:
                PORTAbits.RA4 = 0;
                PORTC = 0b11100111;
                break;
            case 0x0D:
                PORTAbits.RA4 = 0;
                PORTC = 0b11100001;
                break;
            case 0x0E:
                PORTAbits.RA4 = 0;
                PORTC = 0b11000110;
                break;
            case 0x0F:
                PORTAbits.RA4 = 0;
                PORTC = 0b11001110;
                break;
            default:
                PORTAbits.RA4 = 1;
                PORTC = 0b11111111;
                break;
        }
    }
}
```

まず、すべての I/O をデジタルに設定するため、アナログセレクトを 0 にする。その後、トライステートレジスタを入力に割り当てるポートは 1 を、それ以外は 0 (出力ポート)にする。そして、出力に設定したポートの初期化をしている。1 を設定しているのは、IC トレーナーに使用されている 7 セグメント LED が、アノードコモンであることによる(全消灯)。あとは、while 文で無限ループを作り、

switch-case 文によりその時の入力に応じて、7セグメント LED に表示するための出力を決めている。

5. 基板の作製

74LS47 は、IC トレーナー上にある 16 ピンソケットに挿入されている。そのピン配置を、図 6 に示す。



図 6. 74LS47 のピン配置

74LS47 に代わり、PIC16F1823 を使用するためには、74LS47 が実装されている、16 ピンの IC ソケットに挿入できるようにする必要がある。そのため、14 ピンの PIC16F1823 を、74LS47 の電源及び信号線の配置に変換する基板を作製した。図 7 はその回路図である。

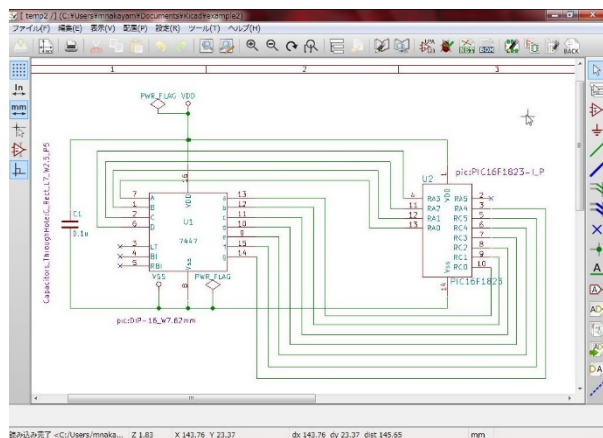


図 7. 変換基板の回路図

基板設計には、フリーウェアである KiCad を使用した。部品の配置は、PIC16F1823 と 16 ピンソケットに挿入するための、細ピンヘッダを実装する部分をまたぐ形にした。また配線は、外部ツールである FreeRouting を使い、自動配線した。基板は、スルーホール両面基板で、PIC とバイパスコンデンサを取り付ける表面を図 8 に、細ピンヘッダを取り付ける裏面を、図 9 に示す。

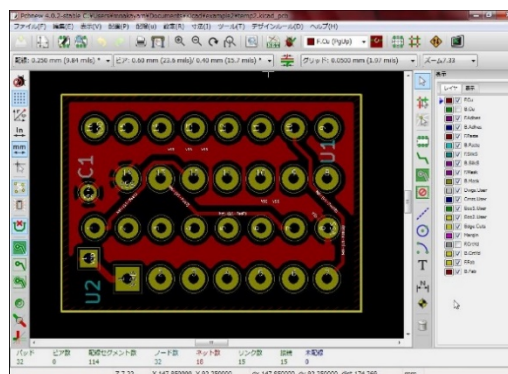


図 8. 基板の表面

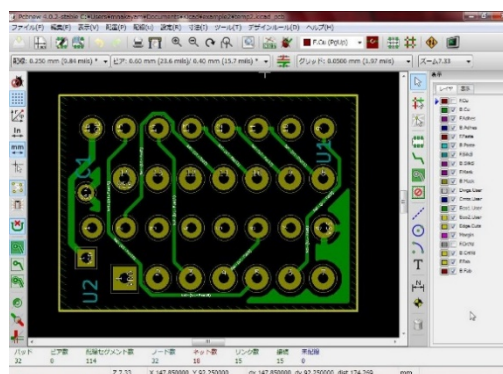


図 9. 基板の裏面

6. まとめ

完成した基板に部品を実装した状態の写真を、図 10 に示す。

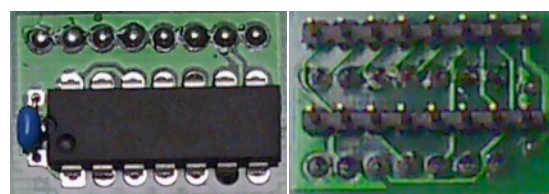


図 10. 完成品(表、裏)

図 11 は、IC トレーナーに実装された 74LS47 のかわりに、今回作製したデコーダを使用し、7 セグメント LED に 10～15 を表示させたものである。



図 11. 16 進表示

なお、従来の 6 の表示は b(11)と同じなので、6 はセグメント a も点灯させ、図 12 のような表示とした。

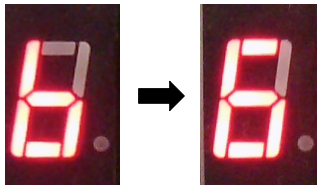


図 12. 6 の表示

その結果、読み取りが難しかった 10～15 の値も、読み取り易くなったと考える。

謝辞

今回、私の提案を採用してくださった、システム情報系和田耕一教授、庄野和宏准教授、富安洋史講師に感謝いたします。

参考文献

- [1] BCD-TO-SEVEN-SEGMENT DECODERS/DRIVERS, TEXAS INSTRUMENTS.
<http://www.tij.co.jp/jp/lit/ds/symlink/sn74ls47.pdf>
- [2] PIC12(L)F1822/PIC16(L)F1823 データシート, Microchip Technology Inc.
http://ww1.microchip.com/downloads/jp/devicedoc/41413c_jp.pdf

Fabrication of 7-segment LED decoder for hexadecimal display

Masaru Nakayama

Technical Service Office for Systems and Information Engineering, University of Tsukuba,
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573 Japan

Keywords: PIC, 7-segmentLED, decoder