

拡張ポート制御ライブラリ(8ビット出力)

概要

ピン数の少ないPIC(PIC12F683・PIC16F88など)では、製作するものによっては、もう少しピン数を増やしたい場合があります。

そこで、今回は、シリアル パラレル変換ICを利用して、ピン数を増やすことが可能な、“拡張ポート”を製作してみました。

<仕様>

- 出力専用のポートを、8ポート提供します。
- 制御信号は、3線式とします。
- 拡張ポートを制御するための制御ライブラリを提供します。

<PIC12F683に拡張ポートを接続した場合のポート数>

	PIC12F683単体	PIC12F683+拡張ポート
入出力ポート	5本	2本
入力ポート	1本	1本
出力ポート	-	8本
合計	6本	11本

<PIC16F88に拡張ポートを接続した場合のポート数>

	PIC16F88単体	PIC16F88+拡張ポート
入出力ポート	15本	12本
入力ポート	1本	1本
出力ポート	-	8本
合計	16本	21本

動作原理

シリアル パラレル変換ICには、新日本無線(JRC)の“NJU3711”を使用します・NJU3711を制御するためには、次の4本の信号が必要ですが・CLR端子を“H(Vdd)”に固定接続することにより、3本の制御信号で制御可能とします。

- シリアルデータ入力端子(DATA)
- クロック信号入力端子(CLK)
- ストローブ信号入力端子(STB)
- クリアー信号入力端子(CLR)

<拡張ポート制御ライブラリで提供する関数>

- ex_port_init();
NJU3711を初期化します。
PICの使用ポート(STB・CLK・DAT)を指定します。

- `ex_port_out(char output_data);`
8ビットのデータ(`output_data`)を出力します。
 - ビット“0” P1
 - ビット“1” P2
 - ビット“2” P3
 - ビット“3” P4
 - ビット“4” P5
 - ビット“5” P6
 - ビット“6” P7
 - ビット“7” P8
- `ex_port_out_bit(char pin, char sts);`
指定したビット(`pin(0~7)`)を、“0”または“1”の状態(`sts`)に設定します。

<動作確認用のプログラムの処理>

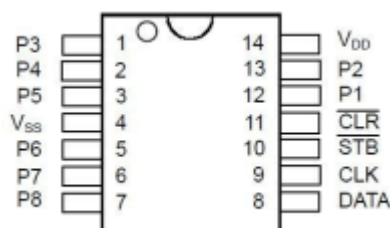
- 拡張ポートを初期化します。
- オープニングデモを行います。
 - 拡張ポートに接続したLEDを、点滅(全部点灯、全部消灯)させます。
 - 拡張ポートに接続したLEDを、順次点灯させます。
 - 拡張ポートに接続したLEDを、順次消灯させます。
- アナログデータ(CH1)を取り込み、値に応じて、拡張ポートに接続したLEDを点灯させます。



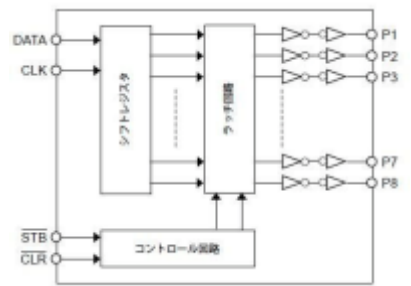
<NJU3711の概観>

<NJU3711の特長>

- 8ビットシリアル入力パラレル出力
- ヒステリシス入力(typ.0.5V)
- 動作電源電圧(5V±10%)
- 動作周波数(5MHz 以上)
- 出力電流(25mA)
- C-MOS構造
- 外形(DIP14/DMP14/SSOP14)



<NJU3711のピン配置>



<NJU3711のブロックダイアグラム>

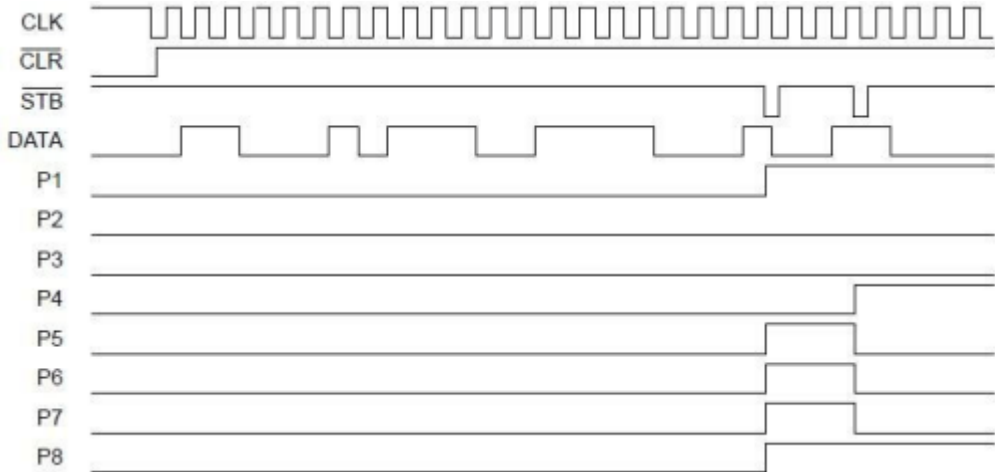
No.	記号	I/O	機能
1	P3	0	パラレル変換データ出力端子
2	P4	0	
3	P5	0	
4	V _{SS}	-	GND
5	P6	0	パラレル変換データ出力端子
6	P7	0	
7	P8	0	
8	DATA	I	シリアルデータ入力端子
9	CLK	I	クロック信号入力端子
10	STB	I	ストローブ信号入力端子
11	CLR	I	クリアー信号入力端子
12	P1	0	パラレル変換データ出力端子
13	P2	0	
14	V _{DD}	-	電源接続端子 (4.5〜5.5V)

<NJU3711の端子説明>

CLK	STB	CLR	動作内容
X	X	L	ラッチ回路の内容が全てリセットされ（シフトレジスタの内容は変化しません）、パラレル出力は全て“L”となります。
↑	H	H	DATA 端子のシリアルデータがシフトレジスタに取り込まれます。この時、ラッチ回路の内容は変化しません。
L	L	H	シフトレジスタの内容がラッチ回路に転送され、ラッチ回路の内容がパラレル出力から出力されます。
H			STB=“L”、CLR=“H” の状態で CLK が入力されると、シフトレジスタの内容がシフトし、これに従ってラッチ回路の内容も換わります。

<NJU3711の制御信号> (注1) X : Don't care

<NJU3711のタイムチャート>



ソースコード

動作確認用のプログラム

[ex_port.c](#)

```
//*****
*
/*
    <拡張ポート（出力専用）制御ライブラリ>

    拡張ポート用の□□には□□□□□□□□を使用します。
*/
//*****
*
//■■■関数宣言■■■
extern void    main();
extern void    opening_demonstration();
//*****
*
//■■■インクルード■■■
#include "ex_port_lib_nju3711.h"
//*****
*
//■■■マクロ定義■■■
//NJU3711
sbit    STB      at    GP5_bit;
sbit    CLK      at    GP4_bit;
sbit    DAT      at    GP2_bit;
sbit    STB_Direction  at    TRISI05_bit;
sbit    CLK_Direction  at    TRISI04_bit;
sbit    DAT_Direction  at    TRISI02_bit;
//*****
*
//■■■メイン関数■■■
void    main()
{
    OSCCON = 0b01110000;
    CMCON0 = 0b00000111;
    ANSEL  = 0b00000001;
    TRISIO = 0b00001011;
    //
    ex_port_Init();
    //
    opening_demonstration();
    //
    while (1) {
        switch (Adc_Read(0) / 114) {
            case 0:
                ex_port_out(0b11111111);
                break;
            case 1:
                ex_port_out(0b11111110);
                break;
            case 2:
                ex_port_out(0b11111100);
                break;
        }
    }
}
```

```
        case 3:
            ex_port_out(0b11111000);
            break;
        case 4:
            ex_port_out(0b11110000);
            break;
        case 5:
            ex_port_out(0b11100000);
            break;
        case 6:
            ex_port_out(0b11000000);
            break;
        case 7:
            ex_port_out(0b10000000);
            break;
        case 8:
            ex_port_out(0b00000000);
            break;
    }
    Delay_ms(100);
}

}

//*****
*
//■■■オープニングデモ関数■■■
void opening_demonstration()
{
    short cnt;
    //
    for (cnt = 0; cnt < 10; cnt++) {
        ex_port_out(0x00);
        Delay_ms(100);
        ex_port_out(0xFF);
        Delay_ms(100);
    }
    //
    for (cnt = 0; cnt < 8; cnt++) {
        ex_port_out_bit(cnt, 0);
        Delay_ms(100);
    }
    for (cnt = 0; cnt < 8; cnt++) {
        ex_port_out_bit(cnt, 1);
        Delay_ms(100);
    }
}

//*****
*
```

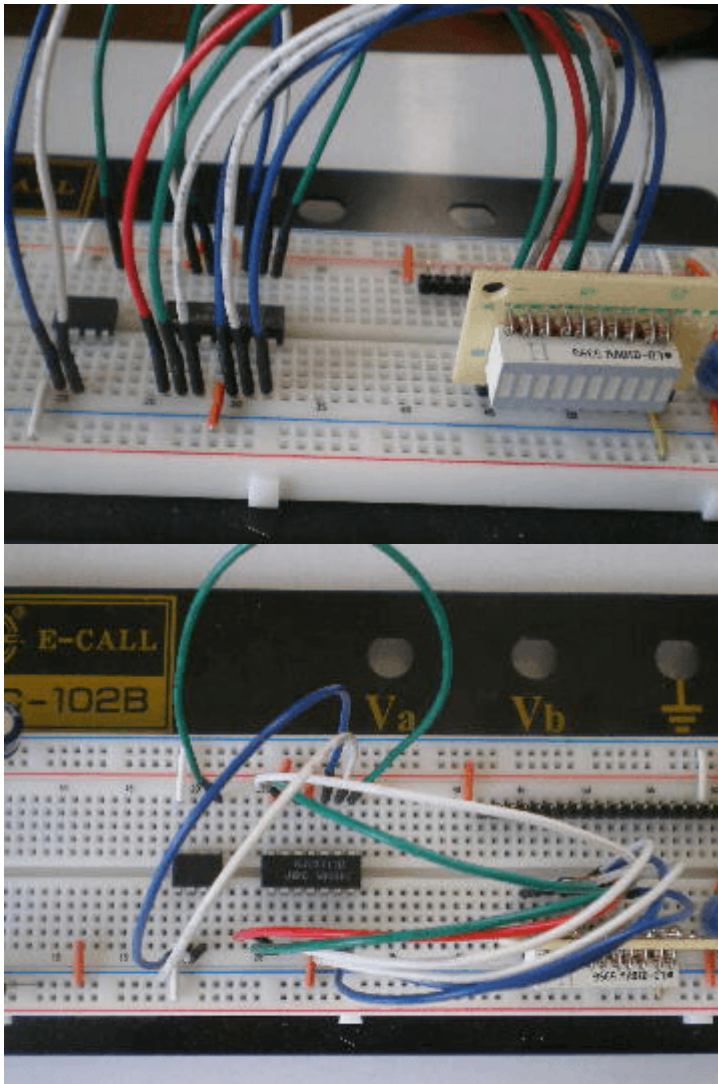
ex_port_lib_nju3711.c

```
//*****
*
/*
    <拡張ポート（出力専用）制御ライブラリ>

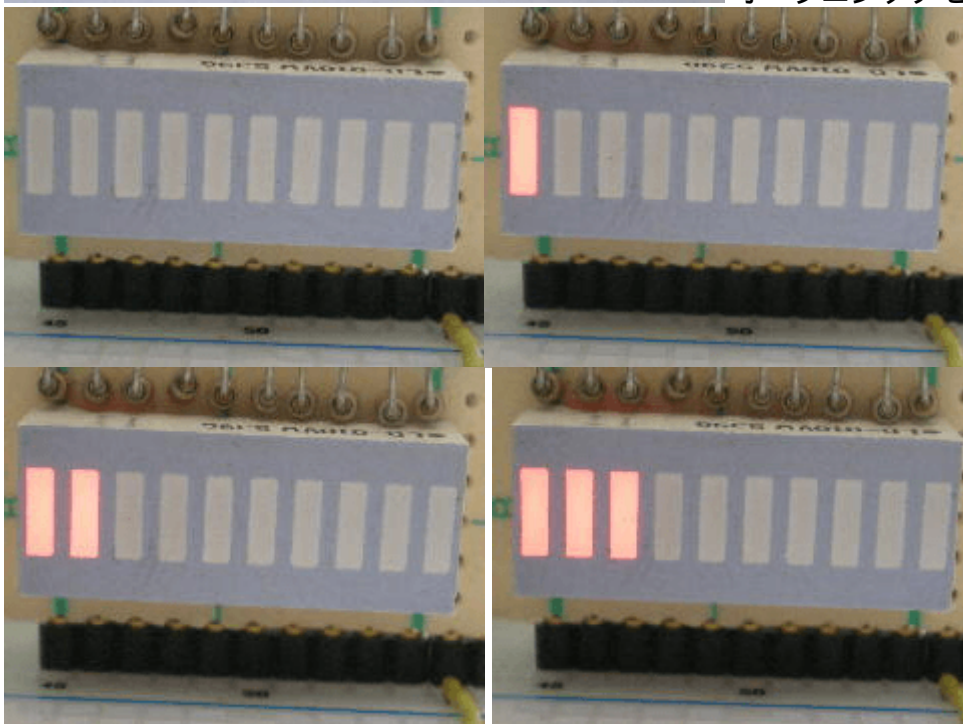
    拡張ポート用の□□には□□□□□□□□を使用します。
*/
//*****
*
//■■■ インクルード■■■
#include "ex_port_lib_nju3711.h"
//*****
*
//■■■ 拡張ポート初期化関数■■■
char    output_data_register;
void    ex_port_init()
{
    short    cnt;
    //
    STB_Direction = 0;
    CLK_Direction = 0;
    DAT_Direction = 0;
    //
    STB = 1;
    DAT = 0;
    CLK = 0;
    for (cnt = 0; cnt < 8; cnt++) {
        CLK = 1;
        CLK = 0;
    }
    STB = 0;
    STB = 1;
    output_data_register = 0;
}
//*****
*
//■■■ 拡張ポート出力関数■■■
void    ex_port_out(char output_data)
{
    short    cnt;
    //
    output_data_register = output_data;
    for (cnt = 0; cnt < 8; cnt++) {
        if ((output_data & 0b10000000) != 0) {
            DAT = 1;
        } else {
            DAT = 0;
        }
        CLK = 1;
    }
}
```

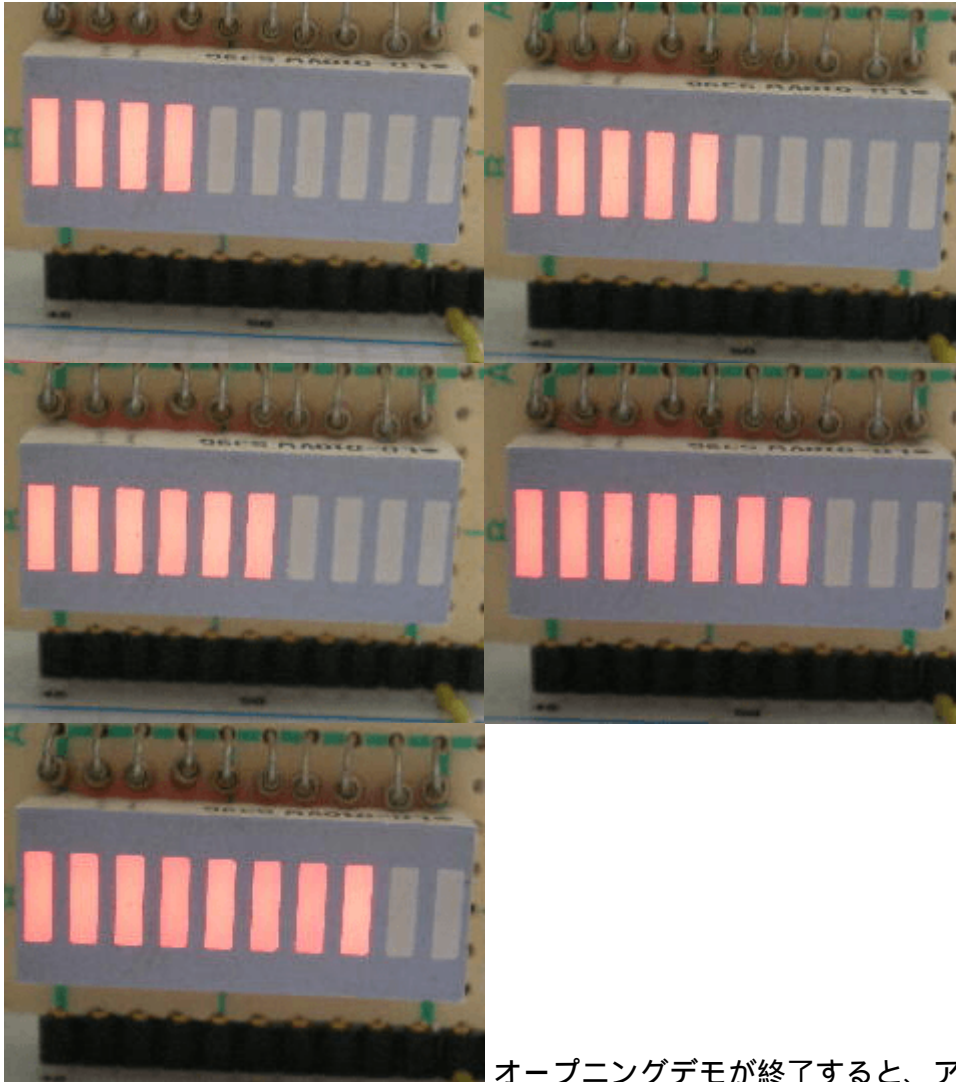
```
        CLK = 0;
        output_data = output_data << 1;
    }
    STB = 0;
    STB = 1;
}
//*****
*
//■■■ 拡張ポートビット出力関数 ■■■
void    ex_port_out_bit(char pin, char sts)
{
    switch (pin) {
    case 0:
        output_data_register.B0 = sts;
        break;
    case 1:
        output_data_register.B1 = sts;
        break;
    case 2:
        output_data_register.B2 = sts;
        break;
    case 3:
        output_data_register.B3 = sts;
        break;
    case 4:
        output_data_register.B4 = sts;
        break;
    case 5:
        output_data_register.B5 = sts;
        break;
    case 6:
        output_data_register.B6 = sts;
        break;
    case 7:
        output_data_register.B7 = sts;
        break;
    }
    ex_port_out(output_data_register);
}
//*****
*
```

動作確認



オープニングデモの様子です。





オープニングデモが終了すると、アナログ信号のレベルメータになります。

拡張ポート制御ライブラリの動作確認にはPIC12F683を使用しましたが、他のPICでも同様に動作します。

From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic12f683:33&rev=1588136635>

Last update: 2025/10/17 14:27

