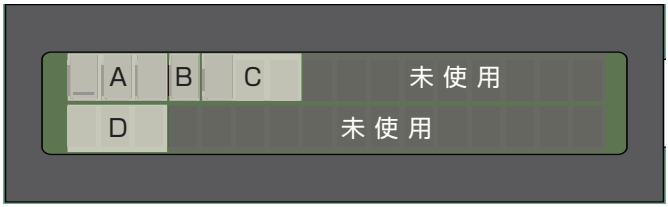
09-1 動作説明

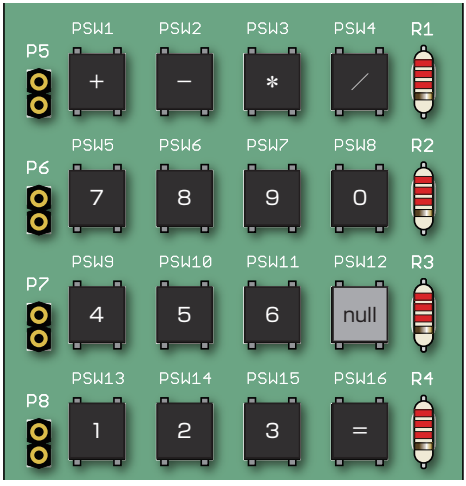
LCD とドットマトリクス SW を利用して**四則演算の電卓**

LCD の表示文字配置

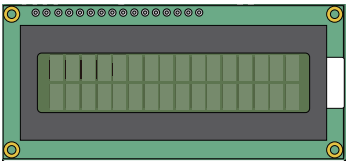


A　1 つ目の数を表示（1 ～ 3 文字）
B　演算を表示（1 文字）
C　2 つ目の数を表示（1 ～ 3 文字）
D　計算結果を表示（1 ～ 3 文字）

ドットマトリクス SW の割り当て

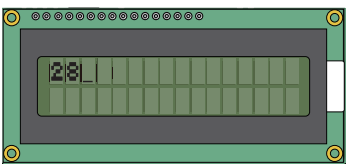


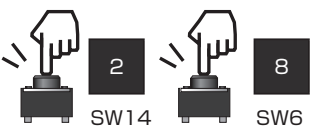
①電源を入れると，カーソルが表示される



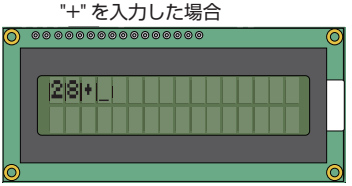
②ドットマトリクス SW を用いて計算式を入力

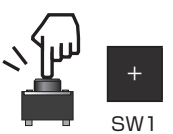
1 つ目の数を入力（数は 0 ～ 255 まで）
2 → 8 と入力した場合



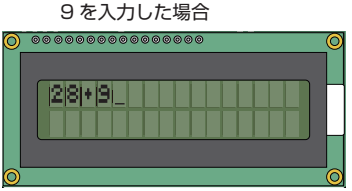


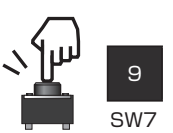
演算を入力
"+" を入力した場合



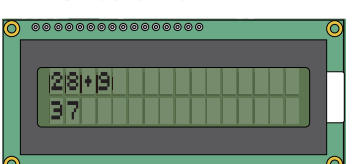


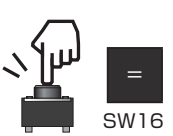
2 つ目の数を入力（数は 0 ～ 255 まで）
9 を入力した場合





③ "=" を入力し計算結果を表示

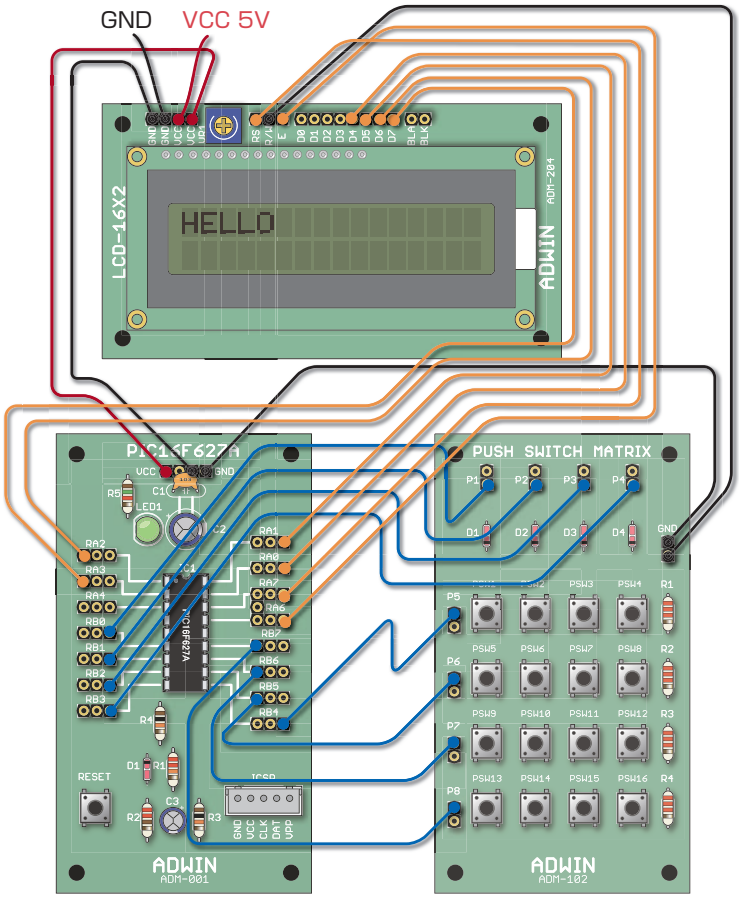




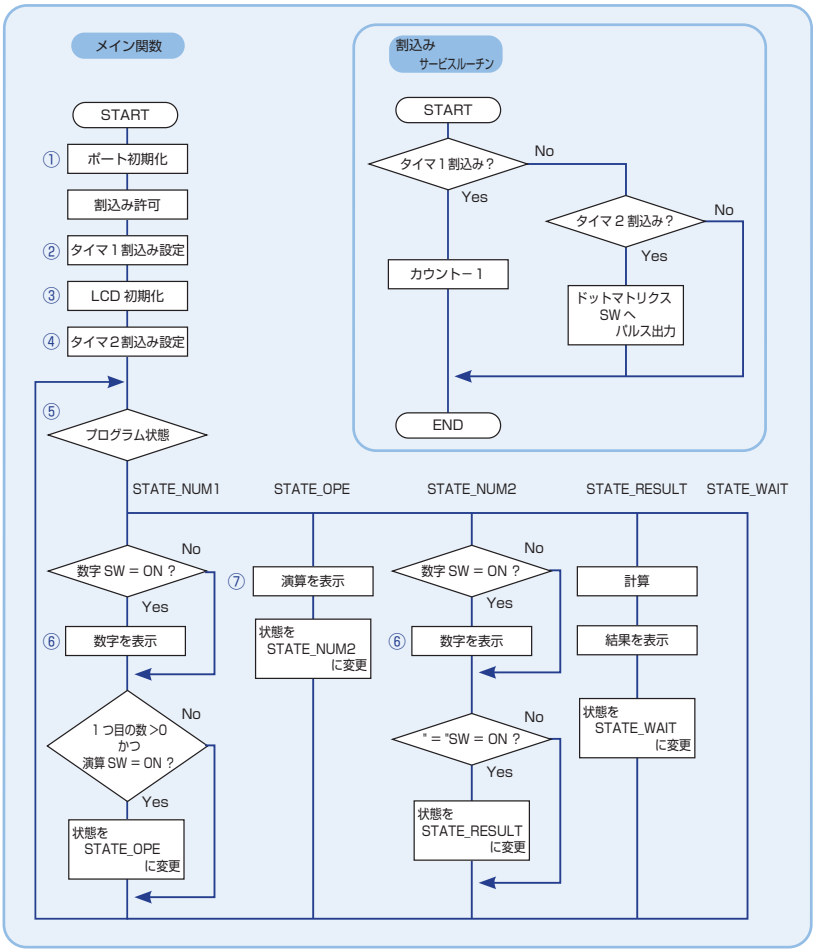
※入力できる数と計算結果で
表示できるは 0 ～ 255 まで

09-2 配線

使用ボードNo. ADM-001・ADM-102・ADM-204



09-3 概略フローチャート



09-4 概略フローチャート解説

① LCD と接続したピンは出力ピンに設定
ドットマトリクス SW の P1 ～ P4 に接続したピンは出力に設定
ドットマトリクス SW の P5 ～ P8 に接続したピンは入力に設定

② 待ち時間用にタイマ 1 割込みを使用（16 ビットタイマ）
＜設定内容＞
・プリスケーラ：1
・クロック源　：内部クロック
・カウント周期：1 μ秒
・タイマ初期値：64535　1 m秒おきに割り込みを発生させるため
※ 1m 秒おきに割り込み発生

③ LCD の初期化や表示は，ADM-202 の仕様書参照

④ ドットマトリクスの P1 ～ P4 に順次パルスを送るために使用
（詳細は ADM-102 の仕様書参照）
＜設定内容＞
・プリスケーラ：1
※ 256 μ秒おきに割込み

⑤ プログラムを 5 つの状態に分け遷移させる

遷移順	状態名	内容
1	STATE_NUM1	1 つ目の数を入力
2	STATE_OPE	演算を入力
3	STATE_NUM2	2 つ目の数を入力
4	STATE_RESULT	計算し，結果を表示
5	STATE_WAIT	永久ループに入り終了

⑥ どの数字が入力されたか検出し，数を表示する
ドットマトリクス SW の検出方法については，ADM-102 の仕様書参照

⑦ どの演算が入力されたか検出し，演算を表示する

09-5 ソースコード

掲載ソースは弊社 HP で公開中のエレモサンプルソースからの抜粋です。
（URL : https://www.adwin.com/image/support/ADM_SampleSource.zip）

ソースファイル
ソースは容量が大きくなったため，分割

main.c	メインプログラム
lcd.c	LCD 用の関数 待ち時間関数
lcd.h	LCD 用のマクロ定義や関数のプロトタイプ宣言 待ち時間関数用のプロトタイプ宣言
timer.c	タイマ割込み用の関数
timer.h	タイマ割込み用の関数のプロトタイプ宣言

主な関数

inputNum 関数	数を入力する関数
inputOpe 関数	演算を表示する関数
result 関数	計算し，結果を表示する関数

主な変数，配列

g_ope	入力した演算を格納
a_num[2]	入力した数を格納。 1 つ目は a_num[0] に， 2 つ目は a_num[1] に 格納する
ans	計算結果を格納
state	プログラムの状態を格納 プログラムの各状態はマクロ定義する

以降，ソースの抜粋

プログラム状態マクロ

```
#define    STATE_NUM1      0      // プログラム状態 1
#define    STATE_NUM2      1      // プログラム状態 2
#define    STATE_OPE       2      // プログラム状態 3
#define    STATE_RESULT    3      // プログラム状態 4
#define    STATE_WAIT      4      // プログラム状態 5
```

プログラムの状態による処理の分岐

```
while(1){
    switch ( state ){
        case STATE_NUM1:{
        /* 処理 */
        }
        case STATE_OPE:{
        /* 処理 */
        }
        case STATE_NUM2:{
        /* 処理 */
        }
        case STATE_RESULT:{
        /* 処理 */
        }
        case STATE_WAIT:{
        /* 処理 */
        }
    }
}
```

割込みサービスルーチン

```
static void interrupt isr(void){
    /* タイマ 1 割込み処理 待ち時間関数用 */
    if (TMR1IF == 1 ){
        g_cnt--;
        TMR1IF = 0;
        TMR1L = 0b00010111;
        TMR1H = 0b11111100;
    }

    /* タイマ 2 割込み処理 ドットマトリクス SW へ順次パルスを出力 */
    else if ( TMR2IF == 1 ){
        if ( RB3 == 1 )  {PORTB = 0x01;}
        else             {PORTB <<= 1; }
        TMR2IF = 0;
    }
}
```

どの SW が押されたか検出

```
/* どの SW が押されたか検出 */
if ( RB7 ){
    if ( RB0 )       {wk = 1;}  // 数字 1 が押された場合
    else if ( RB1 )  {wk = 2;}  // 数字 2 が押された場合
    else if ( RB2 )  {wk = 3;}  // 数字 3 が押された場合
}
if ( RB6 ){
    if ( RB0 )       {wk = 4;}  // 数字 4 が押された場合
    else if ( RB1 )  {wk = 5;}  // 数字 5 が押された場合
    else if ( RB2 )  {wk = 6;}  // 数字 6 が押された場合
}
if ( RB5 ){
    if ( RB0 )       {wk = 7;}  // 数字 7 が押された場合
    else if ( RB1 )  {wk = 8;}  // 数字 8 が押された場合
    else if ( RB2 )  {wk = 9;}  // 数字 9 が押された場合
    else if ( RB3 )  {wk = 0;}  // 数字 0 が押された場合
}
```

計算

```
switch ( g_ope ){
  case 0:  ans = a_num[0] + a_num[1];  break; // 加算の場合の処理
  case 1:  ans = a_num[0] - a_num[1];  break; // 減算の場合の処理
  case 2:  ans = a_num[0] * a_num[1];  break; // 乗算の場合の処理
  case 3:  ans = a_num[0] / a_num[1];  break; // 除算の場合の処理
}
```

計算結果の表示

```
/* 計算結果を表示 */
if ( ans > 99 )  { lcd_putc(a_val[ans/100] ); }
// 計算結果が 3 桁以上の場合，3 桁目を表示
if ( ans > 9 )   { lcd_putc(a_val[(ans/10)%10] ); }
// 計算結果が 2 桁以上の場合，2 桁目を表示
lcd_putc(a_val[ans%10] ); // 1 桁目を表示
```