AVR340：汎用入出力を用いたLCDの直接駆動

要点

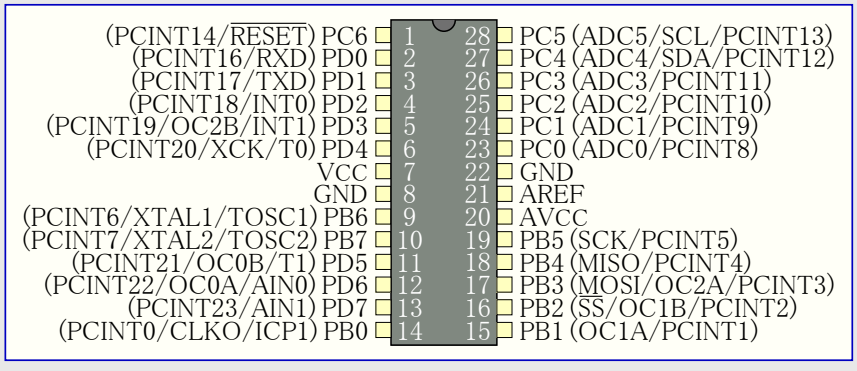
- 4Kﾊﾞｲﾄ自己ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾌﾟﾛｸﾞﾗﾑ用ﾌﾗｯｼｭ ﾒﾓﾘ
- 512ﾊﾞｲﾄのSRAM、256ﾊﾞｲﾄのEEPROM
- 8ﾁｬﾈﾙの10ﾋﾞｯﾄA/D変換器(TQFP/MLF)
- ﾃﾞﾊﾞｯｸﾞWIREﾁｯﾌﾟ上ﾃﾞﾊﾞｯｸﾞ ｼｽﾃﾑ
- 20MHzで最大20MIPSの単位処理量
- 外囲器に依存する、23～28の入出力線
- PDIP(下図)、TQFP、QFN/MLF外囲器が利用可能

1. 序説

多くの製品がﾏｲｸﾛ ｺﾝﾄﾛｰﾗ(MCU)へｲﾝﾀｰﾌｪｰｽされる液晶表示器(LCD:Liquid Crystal Dis-play)を必要とします。この応用記述は多重化されたLCDの操作を記述します。また、LCDを操作するCﾌﾟﾛｸﾞﾗﾑだけでなく、LCDによって必要とされる電気的波形や接続も検討されます。その結果は多くの製品に対する開始点と素晴らしく安価な組み合わせです。

動作するために、多重化されたLCDが内部的により複雑とは言え、それらは最安価の表示器で、全ての硝子LCD(硝子は基板上の付加LCD駆動部ﾁｯﾌﾟがないことの参照)での最低I/Oﾋﾟﾝ数を必要とします。

図1-1. ATmega48 PDIP外囲器

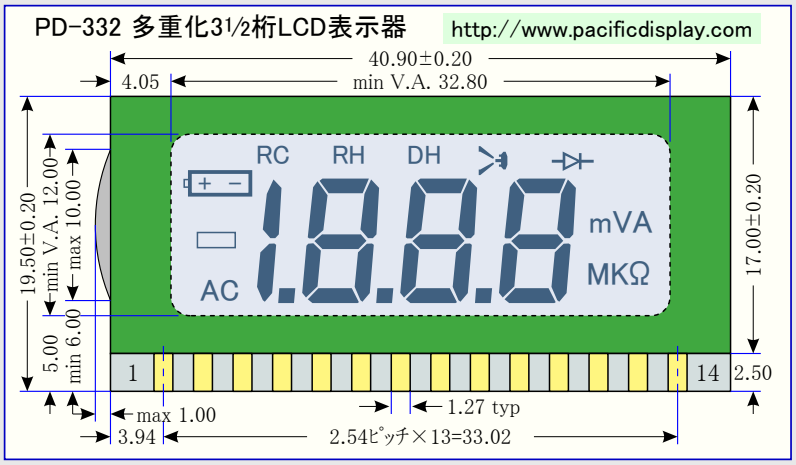


2. LCDの詳細

この内で記述されるLCDは(LCD上にｲﾝﾀｰﾌｪｰｽICがない)硝子だけのﾃﾞﾊﾞｲｽです。LCDは4つの共通(COM)入力と8つのｾｸﾞﾒﾝﾄ入力を持ち、ATmega48で12本の入出力を必要とします。ATmega48では未だ11～16本の追加入出力があります。

LCDは交流(AC)信号、代表的には方形波で供給されなければなりません。静的な直流(DC)信号で供給された場合、LCDは損傷を受け、ｾｸﾞﾒﾝﾄの反転不能極板を引き起こすでしょう。図2-1.はLCD外形の例を示します。

図2-1. LCD外形



8ﾋﾞｯﾄ AVR® マイクロ コントローラ

応用記述

本書は一般の方々の便宜のため有志により作成されたもので、ATMEL社とは無関係であることを御承知ください。しおりのはじめにでの内容にご注意ください。

Rev. 8103A-09/07, 8103AJ2-12/13

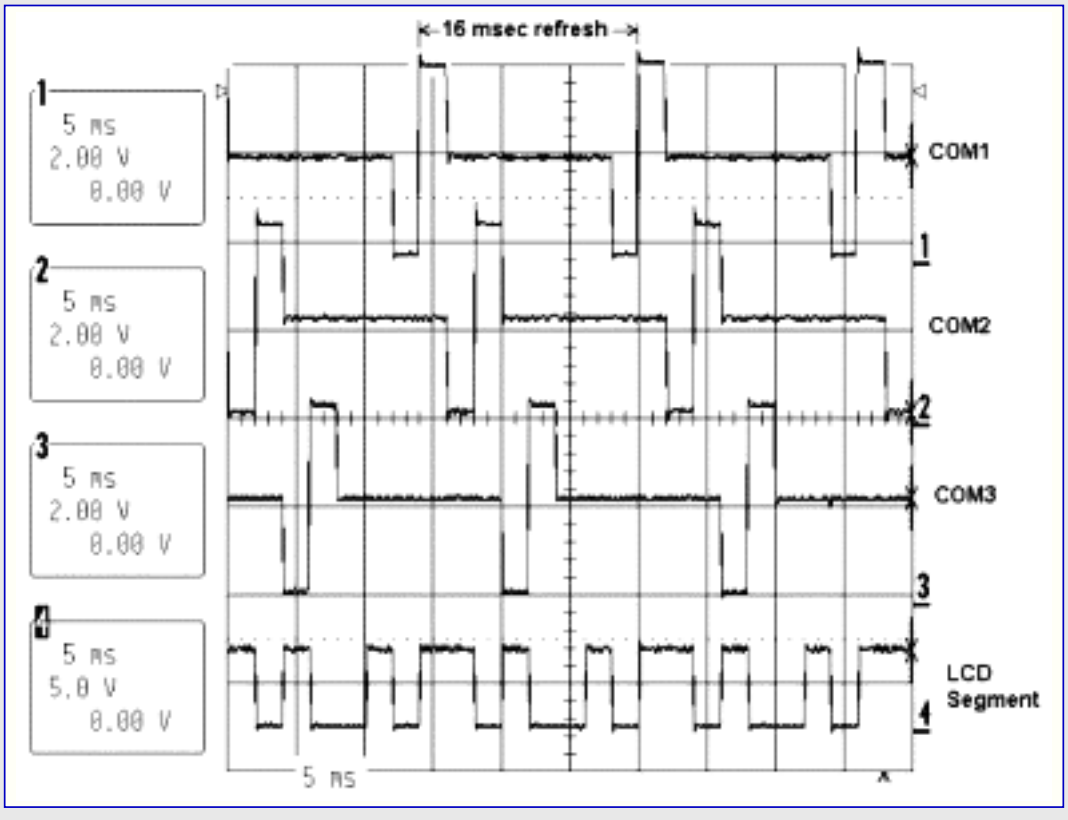
LCDｾｸﾞﾒﾝﾄはCOM入力とｾｸﾞﾒﾝﾄ入力間の電圧差が代表的にAC3.5Vの場合で可視になります。

共通(COM)入力: この例では4つのCOM入力が各々1/2VCCにﾊﾞｲｱｽされます。これは選択したｾｸﾞﾒﾝﾄを可視にするために行われます。**図2-2.**は4つのCOMの内の3つの波形を示します。4つのCOM全ては各々1つのAC周期を生成するように順次許可されます。4つ全てのCOMに対する総周期時間は概ね16m秒で、そしてそれは約60Hzの速度でLCDを再活性します。これはちらつきが感じられない充分な速さで、且つOFFｾｸﾞﾒﾝﾄの残像を防ぐのに充分な遅さです。

このｵｼﾛｽｺｰﾌﾟ画像は各COM信号が1/2VCCから2ms間VCCに、そして2ms間GNDへ、それから1/2VCCへ戻ることを示します。これは「1/2VCC駆動」として参照されます。より複雑なLCDは度々「1/3駆動」を使用しますが、これはこの応用記述の範囲外です。

LCD駆動の付加的な情報については www.atmel.com/avr で利用可能な「**STK502使用者の手引き**」をご覧ください。
各COM周期は4ms間活性で、その後に1/2VCCへ復帰して不活性状態です。これは4つのCOM全てに関して総時間に加算し、16msになります。

図2-2. ｵｼﾛｽｺｰﾌﾟ画像



ｾｸﾞﾒﾝﾄを励行するため、そのｾｸﾞﾒﾝﾄへの波形はそれのCOM波形と180°位相がずれていなければなりません。従ってｾｸﾞﾒﾝﾄとCOMの信号間の電圧差は代表的にAC 5Vで、そしてそれは再活性周期の300～400ms後にｾｸﾞﾒﾝﾄを可視にします。

ｾｸﾞﾒﾝﾄを非励行(OFF切り替え)にするには、そのｾｸﾞﾒﾝﾄのCOMが活性である間、COMとｾｸﾞﾒﾝﾄの波形が互いに同位相であるべきで、それは2.5Vではありません。故に全ｾｸﾞﾒﾝﾄOFFのLCDは各COM入力で同位相のｾｸﾞﾒﾝﾄ入力を持ちます(4ﾁｬﾈﾙ利用できないので、**図2-2.**からCOM3が省略されました)。

3. 回路説明

図3-1.の回路はATmega48とPD-383 LCD間の簡単なｲﾝﾀｰﾌｪｰｽを示します。COM信号上のﾌﾟﾙｱｯﾌﾟ/ﾌﾟﾙﾀﾞｳﾝ抵抗に注意してください。これらの抵抗はCOM信号が不活性状態になる時に必要な1/2VCCを供給します。

LCDに於いて、8つのｾｸﾞﾒﾝﾄ接続はこの応用記述で1A,1B,2A,2B～として再記号化され、Cﾌﾟﾛｸﾞﾗﾑのｾｸﾞﾒﾝﾄ表内のﾃﾞｰﾀと一致します。

図3-1. 接続概要

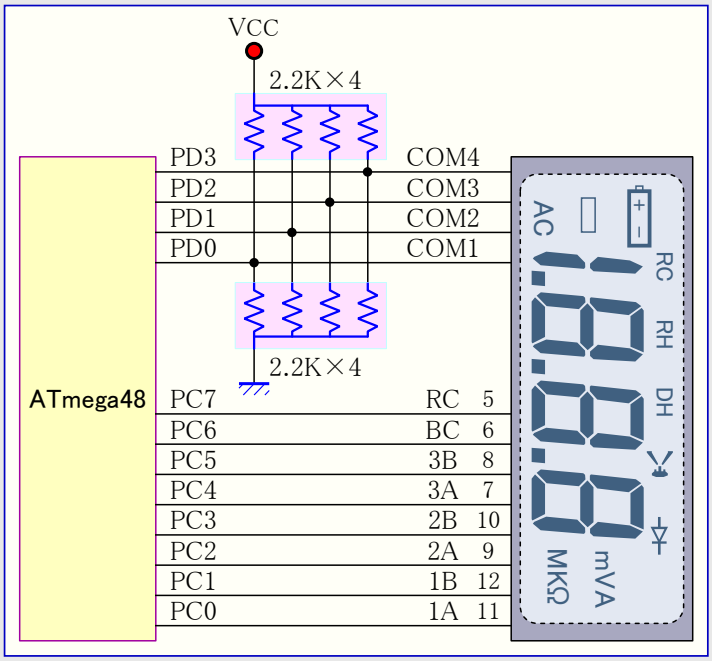
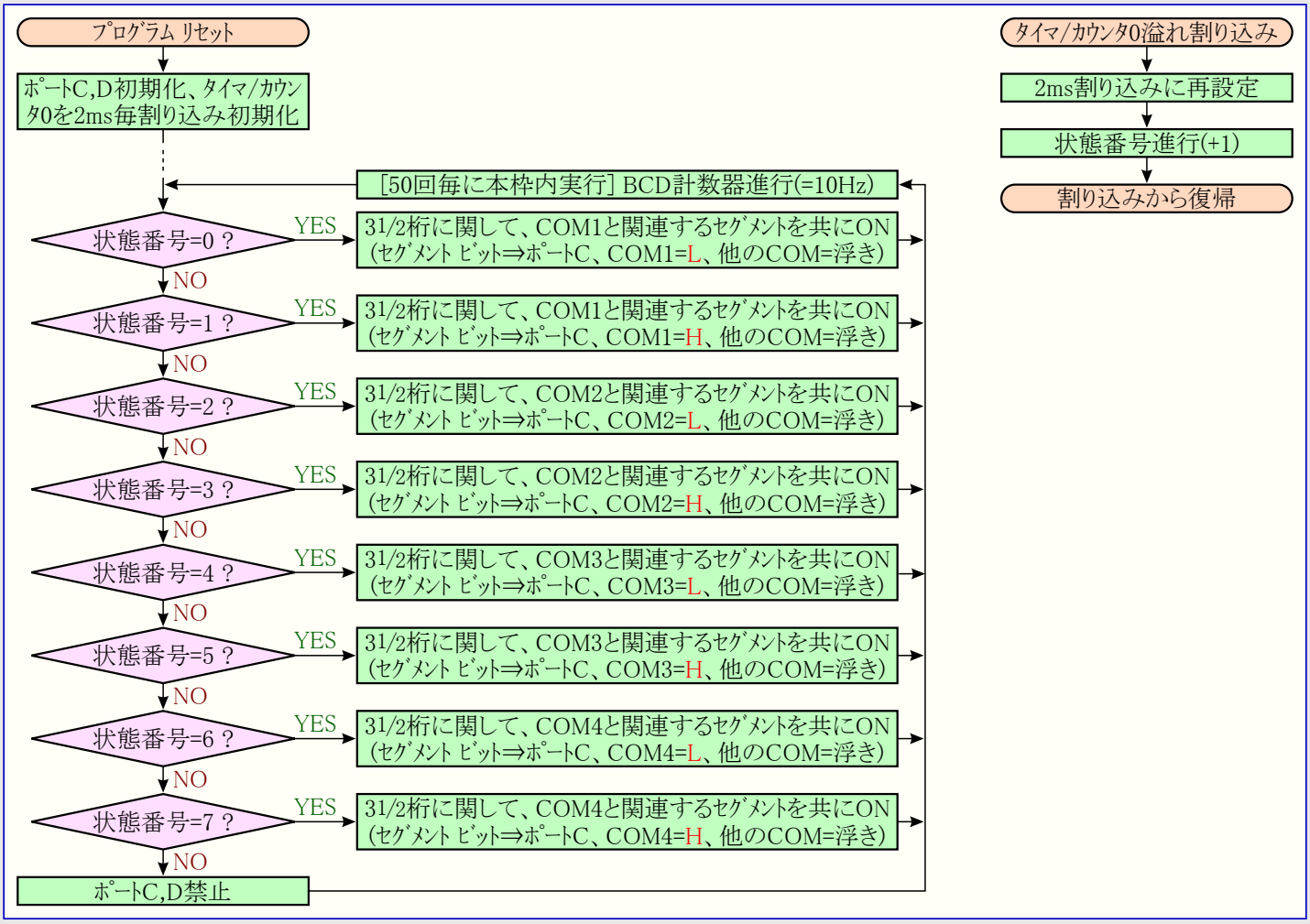


図3-2. ﾌﾟﾛｸﾞﾗﾑ流れ図



4. 参照表操作

参照表(LUT)は文字0～9に対してどのｾｸﾞﾒﾝﾄを活性にするのかを記述したﾊﾞｲﾄの10ﾊﾞｲﾄ長の一覧表です。より多くの文字を定義することができ、新規文字毎に1ﾊﾞｲﾄを必要とします。

基本的な操作は以下です。

- 主ﾌﾟﾛｸﾞﾗﾑに於いて、4つのRAM位置が表示されるべき4つの文字を含みます。これらのRAM位置は0～9の2進数を含み、後でASCII文字に変換できます。
- 各文字位置に関して、COMAとCOMBの入力があります。4つのCOM入力が活性の時に、COMAとCOMBの入力は各文字のLUT値が共に論理和(OR)されたC言語命令の結果です。
- 例として右端位置に"5"を表示するにはLUTの最終列を調べてください。

1. 2ms毎に割り込みが起き、それがｱｸｾｽされるべきSwitch命令文を引き起こします。
2. COM1が活性(case 0)の時に、ﾌﾟﾛｸﾞﾗﾑは"5"の行でLUTをｱｸｾｽして右端2ﾋﾞｯﾄ以外の全てを遮蔽します。これらはAVRのRAM位置"segs_out"変数に加算されます。
3. 同様に、COM1が活性の間、ﾌﾟﾛｸﾞﾗﾑはsegs_out変数に残り3つの表示文字の先頭2ﾋﾞｯﾄを論理和(OR)するためにLUTを更に3回ｱｸｾｽします。
4. 合計2×4=8ﾋﾞｯﾄがsegs_out変数に論理和(OR)され、そしてそれらはPORTC出力ﾎﾟｰﾄに書かれます。
5. 2ms後にSwitch命令文case 1がｱｸｾｽされます。この回は全ﾋﾞｯﾄの極性を反転するためにsegs_outﾃﾞｰﾀが$FFと排他的論理和(XOR)され、これはLCDによって交流(AC)波形が必要とされるために必要とされます。これらの排他的論理和(XOR)された(補数)結果がPORTCに送られます。

- 更に2ms経過後、上の手順が再び起こり、今回はCOM2が活性の間です。Switch命令文case 2が実行されます。今回、各表示文字に対するﾃﾞｰﾀはLUTからｱｸｾｽされ、segs_outで組み立てられ、そしてPORTDでCOM2が活性信号の時にPORTCへ送ります。
- 手順はCOM3とCOM4の信号に関しても全く同じにされます。4つのCOM手順全体は16msかかり、それはSwitch命令文内の8つのcaseの各々に対して2msです。

ﾌﾟﾛｸﾞﾗﾑ流れの更なる詳細については図3-2.を調べてください。

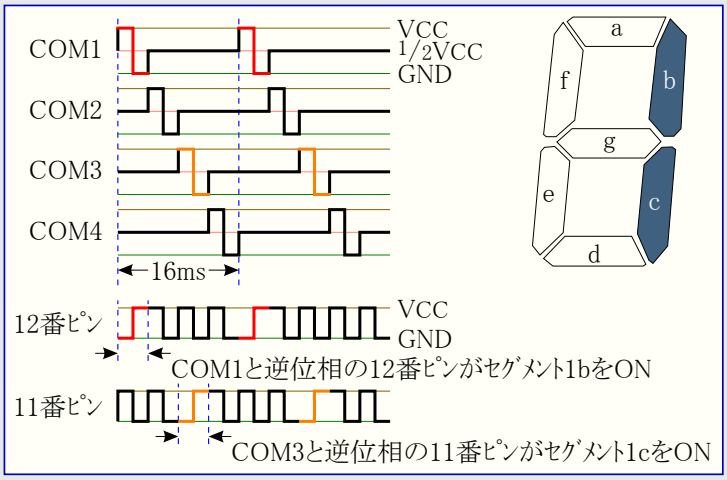
表4-1. 参照表(LUT)

文字	ｾｸﾞﾒﾝﾄ (1=活性,0=不活性)								16進数
	DP (COM4B)	d (COM4A)	c (COM3B)	e (COM4A)	g (COM2B)	f (COM2A)	b (COM1B)	a (COM1A)	
0	0	1	1	1	0	1	1	1	77
1	0	0	1	0	0	0	1	1	22
2	1	1	0	1	1	0	1	1	DB
3	1	0	0	1	0	1	1	1	97
4	0	0	1	0	1	1	1	0	2E
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	0	7C
7	0	0	1	0	0	0	1	1	23
8	1	1	1	1	1	1	1	1	FF
9	0	0	1	0	1	1	1	1	2F

5. COM1～COM4信号の構成図

図5-1.は2つのLCDｾｸﾞﾒﾝﾄに対して交流(AC)波形がどう生成されるかを示します。COM信号は6章で示されるCﾌﾟﾛｸﾞﾗﾑによって生成されます。LCDｾｸﾞﾒﾝﾄを活性にするにはLCDの望むAまたはBの入力ﾋﾟﾝに逆極性波形が印加されなければなりません。4つの文字位置各々に対してAとBの入力があります。

図5-1. 位相詳細と交流(AC)波形



6. 応用ｺｰﾄﾞ

```
// ﾌｧｲﾙ名: LCX_App_Note_5_3_05.c (Pacific DisplaysのPPD-332 3と1/2桁LCD動作のための実演Cｺｰﾄﾞ)
// http://www.hpinfotech.roから入手可能なCodeVision AVR 1.24.4a評価版でｺﾝﾊﾟｲﾙ
// このﾌﾟﾛｸﾞﾗﾑはLCDに進行する4つの10進桁を表示します。LCDは4つのCOMと8つのｾｸﾞﾒﾝﾄ(計12)の接続を持ちます。
// COM1～COM4はﾎﾟｰﾄDで出力します。
// ﾎﾟｰﾄCで出力するｾｸﾞﾒﾝﾄはｾｸﾞﾒﾝﾄ表配列のAとBの値に合わせるために1A,1B,2A,2B,3A,3Bと呼ばれます。
// PD-332は3桁各々に対して桁毎にAとBと記された2つのｾｸﾞﾒﾝﾄの同様のLCD配線を持ちます。

#include <mega48.h>
unsigned char segs_out = 0;
unsigned char state_counter = 8;
unsigned char output_change = 0;
unsigned char LCD_d_1       =0;     // LCD桁値(計数器は"000"から開始)
unsigned char LCD_d_2       =0;
unsigned char LCD_d_3       =0;
unsigned char LCD_d_4       =0;
unsigned char Pt_1_sec      =0;     // 0.1秒計数器

// 以下原型定義
void initialization(void);

#define debug 0
#define LCD_Driver 1

// Pacific DisplaysのPD-332 3と1/2桁4本COM線LCD用参照表(LUT)
// 次表は表示文字0～9に対して10の参照点を持ちます。16進値はLCD入力A,Bに対するCOM1～COM4です。
const unsigned char segment_table[] ={0x77, 0x22, 0xDB, 0x97, 0x2E, 0x6D, 0x7C, 0x23, 0xFF, 0x2F}; // 表示値0～9

// * ﾀｲﾏ/ｶｳﾝﾀ0溢れ割り込み処理ﾙｰﾁﾝ * //
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// ﾀｲﾏ/ｶｳﾝﾀ0値再設定
  TCNT0=5;            // ﾀｲﾏ/ｶｳﾝﾀ0周期=8MHz÷64前置分周=8μs, 2ms=8μs*250, 255-250=5 [2005年5月4日]
  state_counter++;
  output_change = 1; // これはﾒｲﾝ繰り返し用のﾌﾗｸﾞです。
  if (state_counter > 7) state_counter = 0;
} // * 割り込み処理ﾙｰﾁﾝ終了 *

// * ここからﾒｲﾝ開始 *
void main(void) {
// * 初期化関数呼び出し(下で定義される関数をご覧ください。) *
  initialization();
#asm("sei")    // * 全体割り込み許可 *
```

```
// * 次の無限While繰り返しはLCD再活性用のSwitch命令文を含みます。 *
#if LCD_Driver
while (1)
{
  if(output_change){
    output_change = 0;
// 以下の状態番号(state_counter)が各々H/L出力でポートD経由の4つのCOM出力波形を生成します。
    switch (state_counter) {
    case 0: {
      segs_out = (segment_table[LCD_d_1]& 0x03);                    // 1の桁A,Bビット値取得
      segs_out = segs_out | ((segment_table[LCD_d_2]& 0x03)*4);     // 10の桁A,Bビット値取得
      segs_out = segs_out | ((segment_table[LCD_d_3]& 0x03)*16);    // 100の桁A,Bビット値取得
      segs_out = segs_out | ((segment_table[LCD_d_4]& 0x03)*64);    // 1000の桁A,Bビット値取得
      DDRD = 0;
      PORTD = 0x00;
      PORTC = segs_out;
      DDRC  = 0xFF;                 // 常時出力有効
      DDRD  = 0x01;                 // COM1=Low設定
    }
    break;
    case 1: {
      PORTD = 0x01;
      PORTC = segs_out ^ 0xFF;      // 逆相セグメント出力
      DDRC  = 0xFF;                 // 常時出力有効
      DDRD  = 0x01;                 // COM1=High設定
    }
    break;
    case 2: {
      segs_out = (segment_table[LCD_d_1]& 0x0C)/4;                  // 1の桁A,Bビット値取得
      segs_out = segs_out | (segment_table[LCD_d_2]& 0x0C);         // 10の桁A,Bビット値取得
      segs_out = segs_out | ((segment_table[LCD_d_3]& 0x0C)*4);     // 100の桁A,Bビット値取得
      segs_out = segs_out | ((segment_table[LCD_d_4]& 0x0C)*16);    // 1000の桁A,Bビット値取得
      DDRD  = 0;
      PORTD = 0x00;
      PORTC = segs_out;
      DDRC  = 0xFF;                 // 常時出力有効
      DDRD  = 0x02;                 // COM2=Low設定
    }
    break;
    case 3: {
      PORTD = 0x02;
      PORTC = segs_out ^ 0xFF;      // 逆相セグメント出力
      DDRC  =0xFF;
      DDRD  = 0x02;                 // COM2=High設定
    }
    break;
    case 4: {
      segs_out = (segment_table[LCD_d_1]& 0x30)/16;                 // 1の桁A,Bビット値取得
      segs_out = segs_out | ((segment_table[LCD_d_2]& 0x30)/4);     // 10の桁A,Bビット値取得
      segs_out = segs_out | (segment_table[LCD_d_3]& 0x30);         // 100の桁A,Bビット値取得
      segs_out = segs_out | ((segment_table[LCD_d_4]& 0x30)*4)      // 1000の桁A,Bビット値取得
      DDRD = 0;
      PORTD = 0x00;
      PORTC = segs_out;
      DDRC  = 0xFF;
      DDRD  = 0x04;                 // COM3=Low設定
    }
    break;
```

```
    case 5: {
      PORTD = 0x04;
      PORTC = segs_out ^ 0xFF;      // 逆相ｾｸﾞﾒﾝﾄ出力
      DDRC  = 0xFF;
      DDRD  = 0x04;                 // COM3=High設定
    }
    break;
    case 6: {
      segs_out = (segment_table[LCD_d_1]& 0xC0)/64;                    // 1の桁A,Bﾋﾞｯﾄ値取得
      segs_out = segs_out | ((segment_table[LCD_d_2]& 0xC0)/16);       // 10の桁A,Bﾋﾞｯﾄ値取得
      segs_out = segs_out | ((segment_table[LCD_d_3]& 0xC0)/4);        // 100の桁A,Bﾋﾞｯﾄ値取得
      segs_out = segs_out | (segment_table[LCD_d_4]& 0xC0);            // 1000の桁A,Bﾋﾞｯﾄ値取得
      DDRD = 0;
      PORTD = 0x00;
//    PORTC = 0x00;                 // LCD_d_3 ^ 0xFF;    // 逆相ｾｸﾞﾒﾝﾄ出力
      PORTC = segs_out;
      DDRC  = 0xFF;
      DDRD  = 0x08;                 // COM4=Low設定
    }
    break;
    case 7: {
      PORTD = 0x08;
      PORTC = 0x55;
//    PORTC = 0xFF;                 // LCD_d_3;
      PORTC = segs_out ^ 0xFF;      // 逆相ｾｸﾞﾒﾝﾄ出力
      DDRC  = 0xFF;
      DDRD  = 0x08;                 // COM4=High設定
    }
    break;
    default:
      DDRC = 0;
      DDRD = 0;                     // COM1～COM4浮き(入力)
    }
// 0.1s計測のために計数器進行(+1)
    Pt_1_sec++;
    if (Pt_1_sec >=50){      // 0.1s
      Pt_1_sec = 0;
      LCD_d_1++;             // LCD桁用3と1/2桁ﾘﾌﾟﾙｷｬﾘｰBCD計数器
      if (LCD_d_1 >=10){
        LCD_d_1 = 0;
        LCD_d_2++;}
      if (LCD_d_2 >=10){
        LCD_d_2 = 0;
        LCD_d_3++;}
      if (LCD_d_3 >=10){
        LCD_d_3 = 0;
        LCD_d_4++;}
     } // 0.1s計数終了
  } // * Switch命令文終了 *
}
#endif // * 無限繰り返し終了 *
} // * ﾒｲﾝ終了 *
```

```
// * ここに初期化関数定義 *
void initialization(void) {
// あなたの局所変数をここで宣言してください。
// ｼｽﾃﾑ ｸﾛｯｸ前置分周=1
  CLKPR=0x80;
  CLKPR=0x00;
//DDRC=0x7F;           // 7ｾｸﾞﾒﾝﾄ出力
// ﾀｲﾏ/ｶｳﾝﾀ0初期化
  TCCR0A=0x00;
  TCCR0B=0x03;         // =8MHz/64 (2005年3月22日)
  TCNT0=0xC1;
// 外部割り込み初期化
// INT0: OFF
// INT1: OFF
// PCINT0～7のﾋﾟﾝ変化割り込み: OFF
// PCINT8～14のﾋﾟﾝ変化割り込み: OFF
// PCINT16～23のﾋﾟﾝ変化割り込み: OFF
  EICRA=0x00;
  EIMSK=0x00;
  PCICR=0x00;
// ﾀｲﾏ/ｶｳﾝﾀ0割り込み初期化
  TIMSK0=0x01;
// ﾀｲﾏ/ｶｳﾝﾀ1割り込み初期化
  TIMSK1=0x00;
// ﾀｲﾏ/ｶｳﾝﾀ2割り込み初期化
  TIMSK2=0x00;
// ｱﾅﾛｸﾞ比較器初期化
// ｱﾅﾛｸﾞ比較器: OFF
// ｱﾅﾛｸﾞ比較器によるﾀｲﾏ/ｶｳﾝﾀ1捕獲入力: OFF
} // * 初期化関数終了 *
```

7. 参照

http://www.atmel.com/products/AVR/に於けるﾃﾞｰﾀｼｰﾄと更なる詳細

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製品窓口

ｳｪﾌﾞｻｲﾄ
www.atmel.com

技術支援
avr@atmel.com

販売窓口
www.atmel.com/contacts

文献請求
www.atmel.com/literature

本応用記述はATMELのAVR340応用記述(doc8103.pdf Rev.8103A-09/07)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。