

Implementing LCD Control with C in the HT67F40

D/N: HA0294E

Introduction

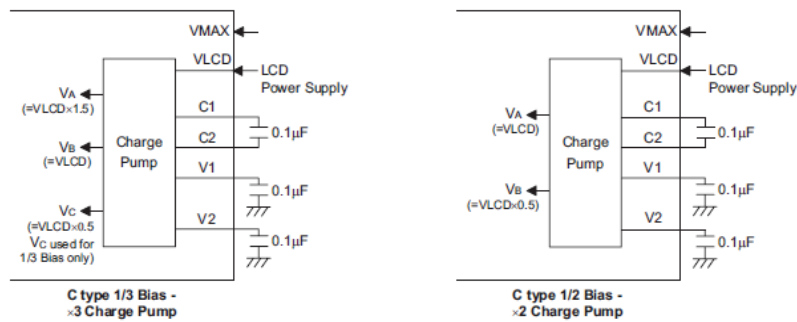
The HT67Fx0 MCUs all include an internal R/C-Type LCD providing 1/2 or 1/3 Bias, 4-COM drive capability. The following example uses the HT67F40 to introduce an LCD Driver with 1/3 Bias and 1/4 Duty.

Operating Principle

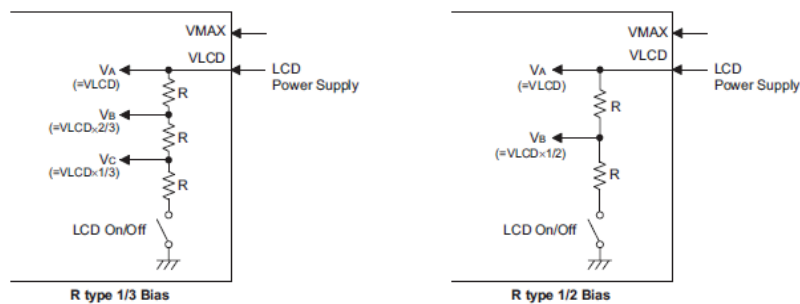
According to LCD driving principles, only an AC voltage can be applied to LCD pixels. The LCD display contrast is determined by the voltage of the COM voltage minus the SEG voltage. When the difference is larger than the LCD saturation voltage, the pixels will be illuminated, otherwise they will be off. The LCD type MCUs automatically generate LCD drive signals using their internal LCD drive circuitry, so the devices can drive LCD panels.

Drive selections are shown in the following table:

Part No.	Duty	Drive No.	Bias	Bias Type	Wave Type
HT67F40	1/2	33×2	1/2 or 1/3	C or R	A or B
	1/3	33×3			
	1/4	32×4			

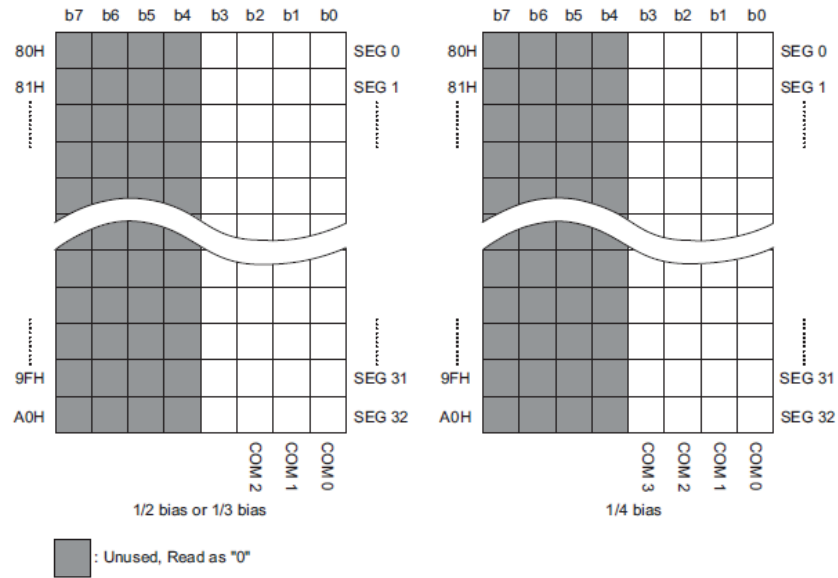


C Type Bias Voltage Levels

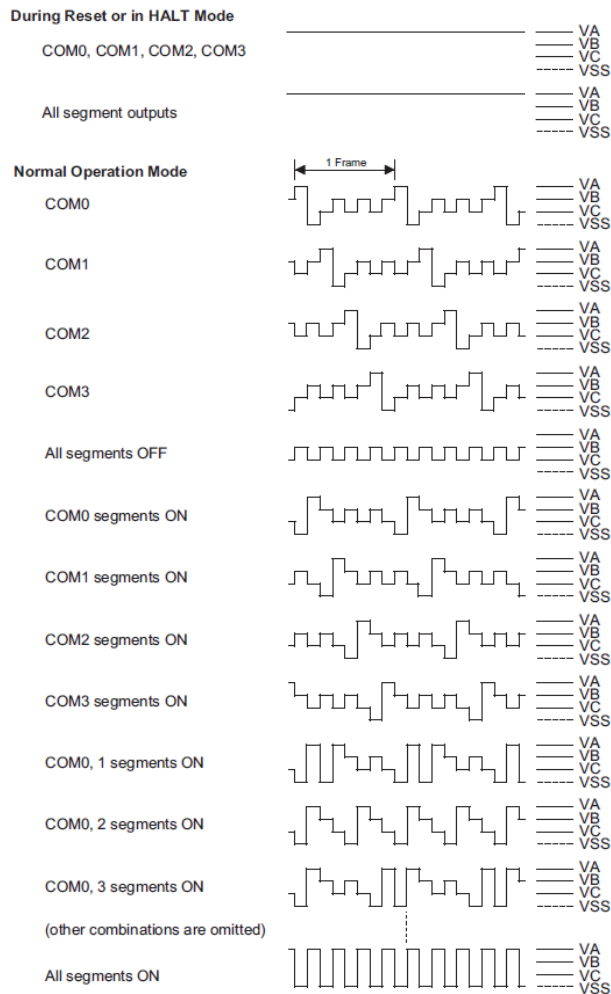


R Type Bias Voltage Levels

LCD Memory Map

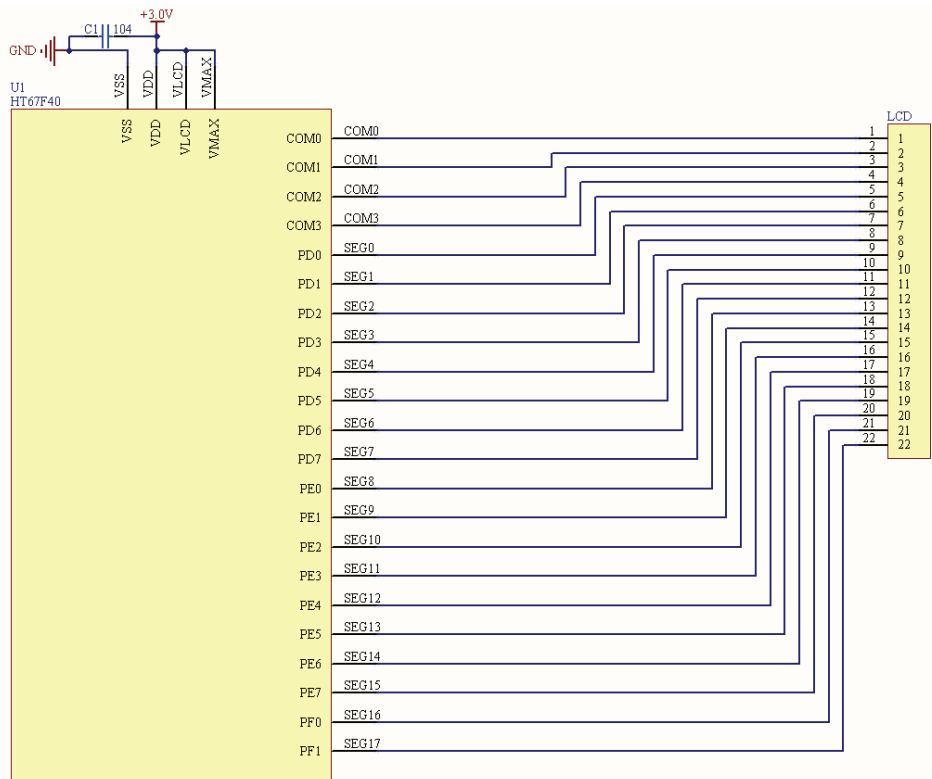


LCD Waveform Timing Diagrams

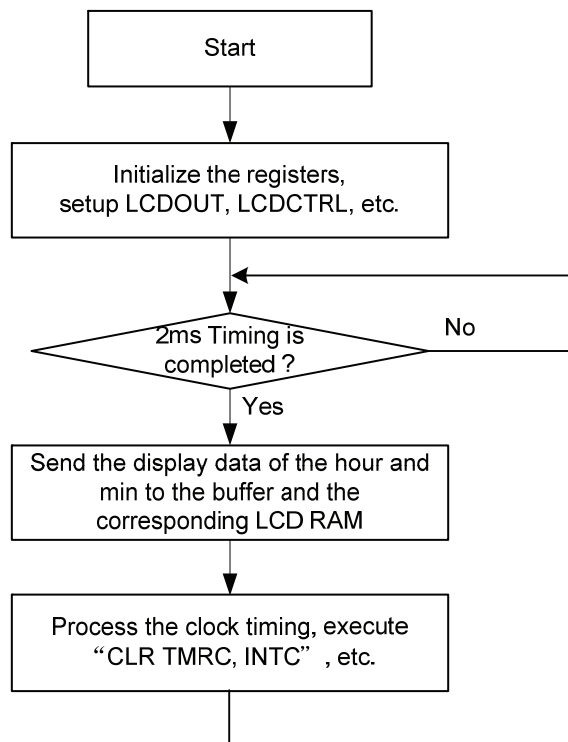


LCD Driver Output – Type A - 1/4 Duty, 1/3 Bias
 Note: For 1/3 R type bias, the VA=VLCD, VB=VLCDx2/3 and VC=VLCDx1/3.
 For 1/3 C type bias, the VA=VLCDx1.5, VB=VLCD and VC=VLCDx1/2.

Application Circuit



S/W Flowchart



Program Description

This program example uses a 4×18 segment LCD and displays cyclic real time clock data from 00:00 to 23:59, using 9 bytes of display buffer RAM (DISP8~DISP16). Refer to the software-defined section for details. Users may decide which COM and SEG are used according to actual requirements. By setting one bit of the LCD RAM to high, its corresponding LCD segment will be illuminated. In a contrary, clear the bit to zero will extinguish the LCD segment.

Program Example

```
//The following are the header files definition

//lcd.h
#ifndef _holtek_h_
#define _holtek_h_

#define u8    unsigned char    /* unsigned 8 bit type definition */
#define s8    signed char      /* signed 8 bit type definition */
#define u16   unsigned int     /* unsigned 16 bit type definition */
#define s16   signed int       /* signed 16 bit type definition */
#define u32   unsigned long    /* unsigned 32 bit type definition */
#define s32   signed long      /* signed 32 bit type definition */

#define nop _nop()
#define clrwdt1 _clrwdt1()
#define clrwdt2 _clrwdt2()

//RAM DEFINE
extern u8    DISP8;           //display buffer
extern u8    DISP9;
extern u8    DISP10;
extern u8    DISP11;
extern u8    DISP12;
extern u8    DISP13;
extern u8    DISP14;
extern u8    DISP15;
extern u8    DISP16;

#define F3_S  DISP8|=0x1 //0
#define F3_C  DISP8&=0xfe
#define G3_S  DISP8|=0x2 //1
#define G3_C  DISP8&=0xfd
#define E3_S  DISP8|=0x4 //2
#define E3_C  DISP8&=0xfb

#define B3_S  DISP9|=0x1 //0
#define B3_C  DISP9&=0xfe
#define C3_S  DISP9|=0x2 //1
#define C3_C  DISP9&=0xfd
#define D3_S  DISP9|=0x4 //2
#define D3_C  DISP9&=0xfb
#define A3_S  DISP9|=0x8 //3
#define A3_C  DISP9&=0xf7

#define F4_S  DISP10|=0x1 //0
#define F4_C  DISP10&=0xfe
#define G4_S  DISP10|=0x2 //1
```

```

#define G4_C DISP10&=0xfd
#define E4_S DISP10|=0x4 //2
#define E4_C DISP10&=0xfb

#define B4_S DISP11|=0x1 //0
#define B4_C DISP11&=0xfe
#define C4_S DISP11|=0x2 //1
#define C4_C DISP11&=0xfd
#define D4_S DISP11|=0x4 //2
#define D4_C DISP11&=0xfb
#define A4_S DISP11|=0x8 //3
#define A4_C DISP11&=0xf7

#define COL_S DISP12|=0x1 //0
#define COL_C DISP12&=0xfe

#define F5_S DISP13|=0x1 //0
#define F5_C DISP13&=0xfe
#define G5_S DISP13|=0x2 //1
#define G5_C DISP13&=0xfd
#define E5_S DISP13|=0x4 //2
#define E5_C DISP13&=0xfb

#define B5_S DISP14|=0x1 //0
#define B5_C DISP14&=0xfe
#define C5_S DISP14|=0x2 //1
#define C5_C DISP14&=0xfd
#define D5_S DISP14|=0x4 //2
#define D5_C DISP14&=0xfb
#define A5_S DISP14|=0x8 //3
#define A5_C DISP14&=0xf7

#define F6_S DISP15|=0x1 //0
#define F6_C DISP15&=0xfe
#define G6_S DISP15|=0x2 //1
#define G6_C DISP15&=0xfd
#define E6_S DISP15|=0x4 //2
#define E6_C DISP15&=0xfb

#define B6_S DISP16|=0x1 //0
#define B6_C DISP16&=0xfe
#define C6_S DISP16|=0x2 //1
#define C6_C DISP16&=0xfd
#define D6_S DISP16|=0x4 //2
#define D6_C DISP16&=0xfb
#define A6_S DISP16|=0x8 //3
#define A6_C DISP16&=0xf7

void Chip_Init();
void DIS_P();
void dis_p3();
void dis_p4();
void dis_p5();
void dis_p6();
void timer_p();

#endif

```

// The following are main program contents

```

//lcd.c
//function: edit for segment0~17 + COM0~3 to drive LCD
//MCU:HT67F40
//option:
//SysVolt: 3.0V
//OSC: HIRC
//WDT: enable
//SysFreq: 4M
//Vlcd: 3.0V
//PB0/RES: I/O pin
//Others select by user

#include "HT67F40.h"
#include "lcd.h"

#pragma vector TM_ISR @0x14 //1ms timer
const u8 DigitCode[16] = {0x28,0x7e,0xa4,0x64,0x72,0x61,0x21,0x7c,0x20,0x60,0x30,
                          0x23,0xa9,0x26,0xa1,0xb1}; //0~f

#pragma rambank0 //define the following for RAM
u8 DISP8; //display buffer
u8 DISP9;
u8 DISP10;
u8 DISP11;
u8 DISP12;
u8 DISP13;
u8 DISP14;
u8 DISP15;
u8 DISP16;

u8 time2ms;
u8 SENDDATA;
u8 STATE;
u8 t500ms;
u16 t1s;
u8 hour;
u8 min;
u8 index;
u16 time1s_dis;
u16 time1s;
u8 tmin;

bit time2msflag;
bit t500msflag;
bit t1sflag;
bit poweronflag;

//-----main-----
void main() //main program
{
    Chip_Init();

    for(_mp0 = 0x80; _mp0 < 0xff; _mp0++) //bank0
    {
        _iar0=0;
    }

    _dmbp0=1;
    for(_mp1 = 0x80; _mp1 < 0xa0; _mp1++) //bank1
    {

```

```

        _iar1=0;
    }
    _dmbp0=0;

poweronflag=1;

while(1)
{
    clrwdt1;
    clrwdt2;
    if(time2msflag)
    {
        time2msflag=0;
        DIS_P();                //send the display data to the buffer

        _dmbp0=1;                //send the display buffer to the LCD RAM
        _mpl = 0x88;
        _iar1= DISP8;
        _mpl = 0x89;
        _iar1= DISP9;
        _mpl = 0x8a;
        _iar1= DISP10;
        _mpl = 0x8b;
        _iar1= DISP11;
        _mpl = 0x8c;
        _iar1= DISP12;
        _mpl = 0x8d;
        _iar1= DISP13;
        _mpl = 0x8e;
        _iar1= DISP14;
        _mpl = 0x8f;
        _iar1= DISP15;
        _mpl = 0x90;
        _iar1= DISP16;
        _dmbp0=0;

        timer_p();                //clock and data processing
    }
}
//-----
void Chip_Init()                //initialization procedure
{
    _cp0c=0;
    _cp1c=0;
    _acer1=0;

    _lcdout0=0;                //seg0~7
    _lcdout1=0;                //seg8~15
    _lcdout2=0b11111100;        //seg16~17
    _lcdctrl=0b01001111;        //1/3 bias, 1/4 duty

    _t0ae=1;
    _ade=0;
    _mf0e=1;
    _tm0c=0;
    _tm0a1=0b11101000;        //1ms
    _tm0ah=0b00000011;
    _tm0c1=0b11000001;
    _t0on=1;
}

```

```

    _emi=1;
}
//-----
void timer_p()                //clock and data processing
{
    if(tlsflag)
    {
        t1sflag=0;
        if(++tmin>=60)
        {
            tmin=0;
            if(++min>=60)
            {
                min=0;
                if(++hour>=24)
                    hour=0;
            }
        }
    }
}
//-----
void DIS_P()                  //display data processing
{
    if (STATE)
    {
        index=hour/10;
        SENDDATA = ~DigitCode[index];
        dis_p3();

        index=hour%10;
        SENDDATA = ~DigitCode[index];
        dis_p4();

        index=min/10;
        SENDDATA = ~DigitCode[index];
        dis_p5();

        index=min%10;
        SENDDATA = ~DigitCode[index];
        dis_p6();

        if(!t500msflag)
            COL_C;
        else
            COL_S;
    }
    else
    {
        if(poweronflag)
        {
            F3_S;
            G3_S;
            E3_S;
            DISP9=0x0f;
            F4_S;
            G4_S;
            E4_S;
            DISP11=0x0f;
            COL_S;
            F5_S;
            G5_S;
        }
    }
}

```

```

        E5_S;
        DISP14=0x0F;
        F6_S;
        G6_S;
        E6_S;
        DISP16=0x0F;
    }
    else
    {
        DISP8=0;
        DISP9=0;
        DISP10=0;
        DISP11=0;
        DISP12=0;
        DISP13=0;
        DISP14=0;
        DISP15=0;
        DISP16=0;
        STATE=1;
    }
}
}
//-----
void dis_p3()                                //display the high bit of the hour
{
    if(SENDDATA & 0x80)
        C3_S;
    else
        C3_C;

    if(SENDDATA & 0x40)
        E3_S;
    else
        E3_C;

    if(SENDDATA & 0x10)
        D3_S;
    else
        D3_C;

    if(SENDDATA & 0x08)
        G3_S;
    else
        G3_C;

    if(SENDDATA & 0x04)
        F3_S;
    else
        F3_C;

    if(SENDDATA & 0x02)
        A3_S;
    else
        A3_C;

    if(SENDDATA & 0x01)
        B3_S;
    else
        B3_C;
}
//-----

```

```

void dis_p4()                                // display the low bit of the hour
{
    if(SENDDATA & 0x80)
        C4_S;
    else
        C4_C;

    if(SENDDATA & 0x40)
        E4_S;
    else
        E4_C;

    if(SENDDATA & 0x10)
        D4_S;
    else
        D4_C;

    if(SENDDATA & 0x08)
        G4_S;
    else
        G4_C;

    if(SENDDATA & 0x04)
        F4_S;
    else
        F4_C;

    if(SENDDATA & 0x02)
        A4_S;
    else
        A4_C;

    if(SENDDATA & 0x01)
        B4_S;
    else
        B4_C;
}
//-----
void dis_p5()                                // display the high bit of the min
{
    if(SENDDATA & 0x80)
        C5_S;
    else
        C5_C;

    if(SENDDATA & 0x40)
        E5_S;
    else
        E5_C;

    if(SENDDATA & 0x10)
        D5_S;
    else
        D5_C;

    if(SENDDATA & 0x08)
        G5_S;
    else
        G5_C;

    if(SENDDATA & 0x04)

```

```

        F5_S;
    else
        F5_C;

    if (SENDDATA & 0x02)
        A5_S;
    else
        A5_C;

    if (SENDDATA & 0x01)
        B5_S;
    else
        B5_C;
}
//-----
void dis_p6()                // display the low bit of the min
{
    if (SENDDATA & 0x80)
        C6_S;
    else
        C6_C;

    if (SENDDATA & 0x40)
        E6_S;
    else
        E6_C;

    if (SENDDATA & 0x10)
        D6_S;
    else
        D6_C;

    if (SENDDATA & 0x08)
        G6_S;
    else
        G6_C;

    if (SENDDATA & 0x04)
        F6_S;
    else
        F6_C;

    if (SENDDATA & 0x02)
        A6_S;
    else
        A6_C;

    if (SENDDATA & 0x01)
        B6_S;
    else
        B6_C;
}
//-----
//1ms timer program
//-----
void TM_ISR()
{
    _t0af=0;

    if(++time2ms>=2)
    {

```

```
        time2ms=0;
        time2msflag=1;

    if(poweronflag)
    {
        if(++time1s_dis>=1000)    //power-on and all pixels illuminated for 2S
        {
            time1s_dis=0;
            poweronflag=0;
        }
    }
    else
    {
        if(++t500ms>=250)
        {
            t500ms=0;
            t500msflag = ~t500msflag; //second flash

            if(++t1s>=2)
            {
                t1s=0;
                t1sflag=1;
            }
        }
    }
}
```

Conclusion

This program example has shown how a display pattern can be generated on a 4×18 LCD. Users can use this example to apply to other selected LCDs with some modification.

Attachment



HT67F40 LCD display program(C language).rar