DEVELOPMENT GUIDE

Beijing DWIN Technology Co., Ltd.

Updated English Version 2.1 at 2011.07.15

Copyright@ Beijing DWIN Technology Co.,Ltd.2003-2011. All Rights Reserved. ICP No.0533781

Development Guide

Content

Ⅰ Text Display

1.1 Text display (printf()function)

There are 2 commands for text display as the following table

Command	*Data*	*Text*	**Display color and display mode**
0x53	x+y+String	ASCII code, 8*8 dot matrix	Display color: set by the palette (Command 0x40) Display mode: foreground color and background color display
0X98	x+y+Lib_ID+C_mode+C_dots+color+Bcolor+String	Any code, any dot matrix	Display color: set by this command 0x98 itself Display mode: set by this command 0x98 itself

★ **(x,y): Coordicate of the left top corner of the first character.**
★ **Character spacing is set by Command 0x41. Word wrap at the end of one line.**
★ **ASCII character will be displayed as per half-angle.**

1.1.1 Text display of fixed content

The fixed content text is often used to make hint message interface. It is recommended to write complete command in the software and send a subprogram of sending some string. Relevant C code and ASM 51 code are as following:

```
//C code for sending string
// 0x CC 33 C3 3C is the frame end

void String (*Str)
{unsigned char d1,d2,d3,d4;
 d1=0x00;
for (;;)
{Txbyte(*Str);
//Send data of one byte through serial
port

d1=d2;
d2=d3;
d3=d4;
d4=*Str;
Str++;
if
(d1=0xcc&&d2==0x33&&d3==0xc3&&
d4=0x3c)
{break;}}
```

```
//ASM 51 code
STRING: CLR A
        MOVC    A, @A+DPTR
        MOV     SBUF, A
        JNB     TI
        INC      DPTR
        MOV      D1, D2
        MOV      D2, D3
        MOV      D3, D4
        MOV      D4, A
        MOV      A, D1
        CJNE      A, #0CCH, STRING
        MOV      A, D2
        CJNE      A,# 33H, STRING
        MOV      A, D3
        CJNE      A, #0C3H, STRING
        MOV      A, D4
        CJNE      A, #3CH,STRING
        RET
```

It is very convenient to use the above-mentioned String()function to display fixed content text. For example, if you want to display "Welcome to Synlink" of dot matrix at (20,100) on the screen, it even can be realized by assembly language as simply as the following code:

```
MOV       DPRT #STR1
LCALL     STRING

STR1   DW 0AA55H, 20, 100
          DB 'Welcome to Synlink',0CCH,33H,0C3H,3CH
```

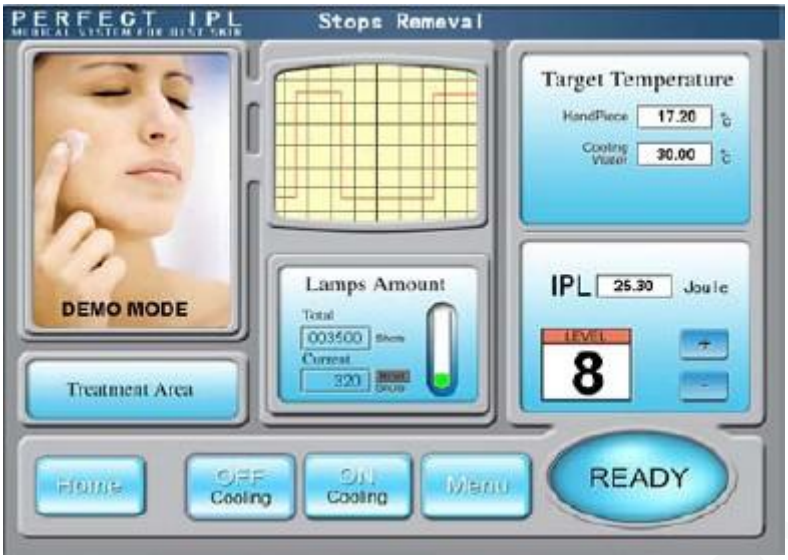
1.1.2 Overlay text display on the background picture

Sometimes, we need to display the text on the background picture without changing the background. There are two circumstances:

A. The background color of the text display position is pure color.
Use Command 0x42 (pick color of the appointed position as the background color) and then display text. In this way, background color of the text is the same with the color of the background picture. The time display "12:33:40" of the following picture is the example.

B. The background color of the text display position is not pure color.

Use Command 0x71 and 0x98 to realize the same effect. First, use Command 0x71 to cover the content that needs to be displayed with the original picture. Next, use Command 0x98 (display foreground color and don't display background color) to display text. Please refer to the following picture, from which you can see the text "DEMO MODE" is displayed on the background picture without changing the background.



1.1.3 Variable display

★ Data variable

Only after being transformed into ASCII text can the data be correctly displayed on the display module. For example, if you need to display data "127"(0x7F), 127 must be transformed into 0x31 0x32 0x37 before being sent to the display module. Relevant C code and ASM 51 code are as following:

```
// Data variable display
void Printn(int x, y, char n)
{ unsigned char a, b;
Txbyte(0xAA);
Txbyte(0x54);// Display ASCII character of 8*16 dot matrix
Txword(x);//Coordinate of the display position
Txword(y);
a=n/100
Txbyte(a+0x30);
b=(n-a*100)/10
Txbyte(b+0x30)
a=n-100*a-10*b
Txbyte(a+0x30);
TxE0F(); // Send CC 33 C3 3C
```

```
// Data variable display
PRINTN:PUSH     ACC
        MOV         DPTR   #0AA54H     //ASCII character of 8*16 dot matrix
        LCALL     TXWORD
        MOV       DPH, PSXH    //X coordinate
        MOV       DPL, PSXL
        LCALL      TXWORD
        MOV        DPH, PSYH   //y coordinate
        MOV        DPL, PSYL
        LCALL      TXWORD
        POP         ACC
        MOV        B, #100
        DIV          AB
        ADD         A, #30H
        LCALL       TXBYTE
        MOV         A, B
        MOV         B, #10
        DIV          AB
        ADD         A, #30H
        LCALL       TXBYTE
```

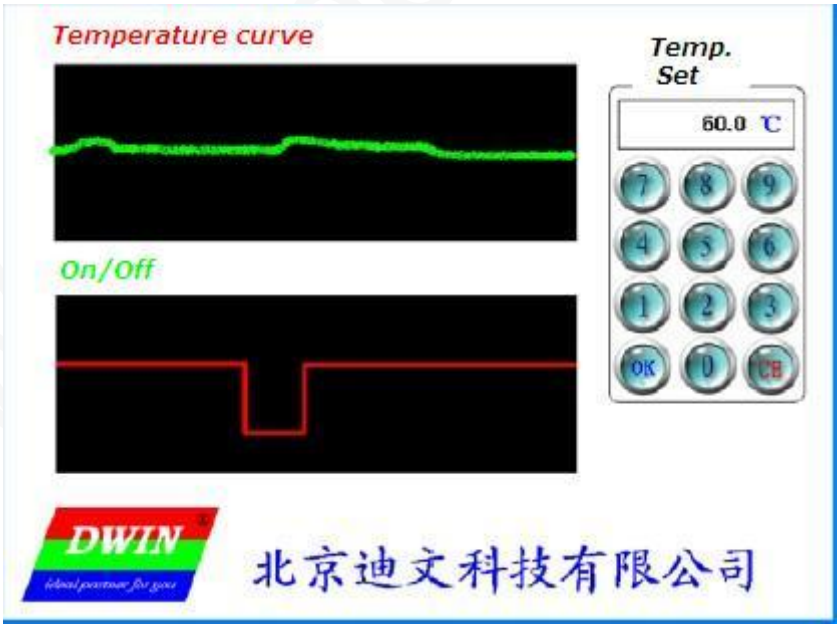
★ Text variable

In assembly program, it is very convenient to realize text variable display with STRING () function. While we can consider using a screen print function Prints () in C language to realize the text variable display.

```
// Prints (0,0 "Welcome to Synlink")
void prints (int x,int y,unsigned char *s)
{Txbyte(0xAA);
Txbyte (0x98);
Txword(x);
Txword(y);
while(*s)
        {Txbyte(*s);
        s++;}
TxE0F;}
```

1.2 Text input (scanf()function)

In most cases, we need to input data or text through keyboard or touch panel. In the following example, we make use of keyboard of touch panel to set temperature. Key "ok" means confirm and key "CE" means quit. Relevant C code is as follows:



```
// Keyboard input function, input format is XX.X
Scanfn()
{int n;
 char d[4],keycode;
 d[0]=0x30;
 d[1]=0x30;
 d[2]='.';
 d[3]=0x30;
 for (;;)
{if (kbhit())
{if(keycode=='E')
{break;}
If (keycode=='C')
{continue;}
d[0]=d[1];   // Move left once after every input, n=(n*10+(keycode-0x30)) is
                changed as the standard input method
d[1]=d[3];
d[3]=keycode
prints(682,110,*d);     // Display data
}}
N=d[0]*100+d[1]*10+d[3];    // Return data
```

Note: The above-mentioned example code is designed according to "anti-mistake input method" which is widely used in industry. When wrong input occurs, there is no need to quit. Re-inputting is ok.

Ⅱ Graphics display

2.1 Real-time dynamic curve display

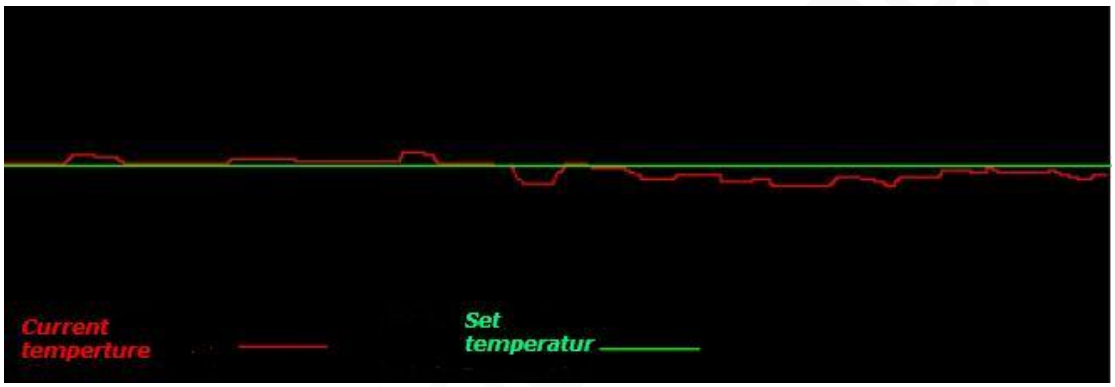
In actual application, we often need to display real-time variable curves. There are two kinds of curves according to width: one is "big curve", like respiratory wave, transient current and other curves with big width. The other is "small curve" with small width. The two curves can be displayed on our display module as the following description:

2.1.1 Real-time small curve display

Command 0x74 is especially for displaying small curve.

Command	*Data*	**Description**
0x74	(X+Y_S+Y_E+Bkcolor)+(Y1+Color1)+…+(Yi+Colori)	1. Y_S is the starting point of Y coordinate, Y_E is the end-point of Y coordinate. 2. Erase the vertical line from (X, Y_S) to (X, Y_E) with the appointed color (Bkcolor). 3. Set a point at (X,Yi) with the color (Colori). Many points at different coordinates can be set simultaneously. Note: The command will not change the palette property.

Note: Take sampling point as X-axis and different variable sampling result as different points of Y-axis. The small curve is realized by continuous upgrade of sampling output (x++). The following C code is for the picture.




```
// Example code for displaying temperature curve by Command 0x74
int x, t_now, t_set;
x=10;       // Initialize the position on the left starting border of the display window which is from
               (10,10) to (790,310)
for (;;)
{rdtmp();   // Read the current temperature as t_now
Txword(x); // Current (sampling) position of the curve
Txword(10); //(curve display window) coordinate of starting point of Y-axis
Txword(310); //(curve display window) coordinate of end-point of Y-axis
Txword(0x0000); //(curve display window) black as the background color
Txword(t_now); //   Position for the current temperature
Txword(0xf800); //red for displaying the current temperature
Twword(t_set); // Position for the set temperature
Txword(0x07e0);// green for displaying the set temperature
TxE0F();   // to send frame end CC 33 C3 3C
x++;   // the next position
```

```
if (x>790)
{x=10;}    //If it reaches the right border of the window, return the position to the left starting
             border.
delay(1);}  // Sampling delay 1mS
```

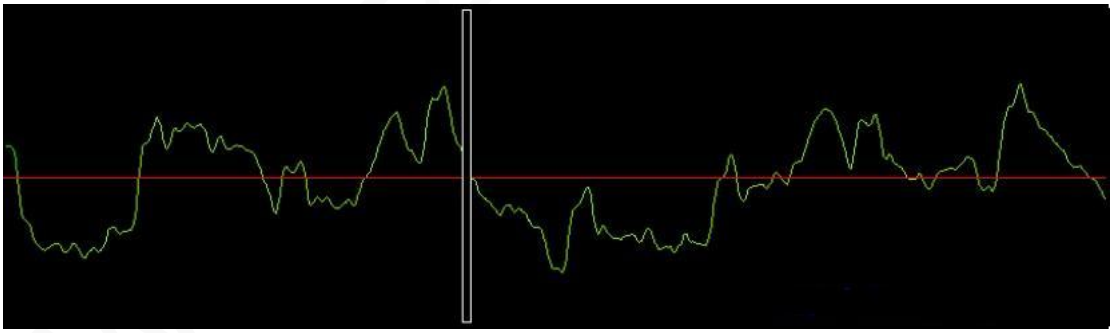
Note: Many curves can be displayed simultaneously if the data (position and color) for the other sampling points are added before TxE0F() function for the above-mentioned example.

2.1.2 Real-time big curve display

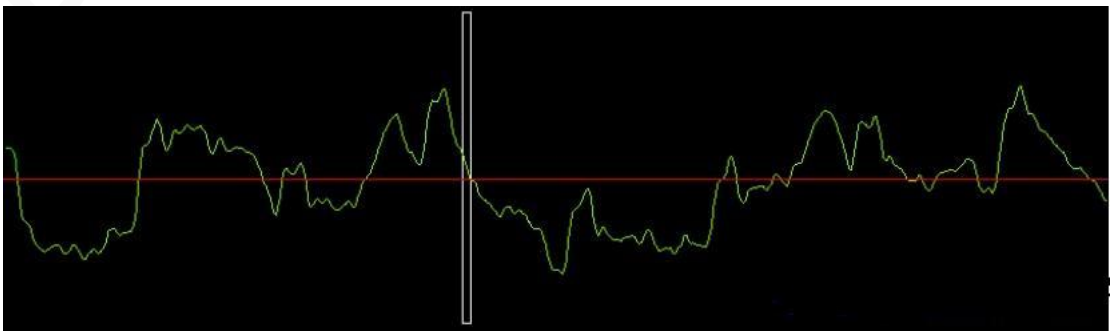
Due to the big width change, big curve display can't be realized by drawing point because the discrete points fail to make the real curve, especially when many curves display simultaneously. Commands 0x56 and 0x5A are used to realize big curve.

Command	*Data*	**Description**
0x56	(X0+Y0)+…+(Xi+Yi)	Connect the appointed points with a line
0x5A	$(X_S+Y_S+X_E+Y_E)_K$	Clear (fill with background color) the rectangle area.(X_S, Y_S) is the coordinate of the left top corner of the rectangle and (X_E+Y_E) is the coordinate of the right bottom of the rectangle.

Example: use command 0x5A clear the area that is going to display the curve.(Note: white lines are used to demonstration. Actually, they are invisible)



Then, connect two nearest sampling points with a line. Repeat the above-mentioned process. Big curve can be realized.



Relevant C code is as follows:

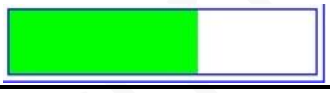
```
int x,v1_now, v1_old, v2_now, v2_old;
x=10;//Display window from (10,10) to (790,470)
v1_old=0; // Initialize v1
v2_old=0; // Initialize v2
for (;;)
{adpro(); // Read A/D sampling result to v1_now, v2_now
clrw(x,10,x+3,470);   //Use Command0x5A to clear the window in which there is going to
                        connection by lines. If the background color remains unchanged, use
                        Command 0x71
setcolor(0x07e0);      //Set the display color as green
line (x,v1_old,x+3,v1_now); //Connect the nearest two results of v1
setcolor(0xf800);     //Set the display color as red
line (x,v2_old,x+3,v2_now); //Connect the nearest two results of v2
x=x+3;              // The next coordinate
if (x>787)          // To judge whether X coordinate is beyond 787=790-3
{x=10;}
v1_old=v1_now;
v2_old=v2_now;
delay (10);}     //A/D time delay 10Ms
```

2.2 Progress bar display

Progress bar is a component in a graphical user interface used to convey the progress of a task. Simple progress bar can be realized by Command 0x59, 0x5A and 0X5B. First, use Command 0x59 to draw the outline sketch of the progress bar and fill the progress part by Command 0x5B according to the progress state. Then use Command 0x5A to clear the rest part. When there is any progress change, repeat Command 0x5B and 0X5A. The format of the three commands is as follows:

```
// The following code realized display of simple progress bar. The coordinate of the left top corner
of the progress bar is (x,y) and the width is 100*20
void status (int x, int y, unsigned char step)
// (x,y) is the coordinate of the internal box of the progress bar, step is the progress.
{setcolor (0x001f);
// Set the external box of the progress bar as blue.
   rectan(x-2, y-2, x+104, y+22);
// Use Command 0x59 to draw a rectangle as the external box whose thickness is 4 pixels.
rectan (x, y, x+102, y+20);
setcolor2(0x07e0,0xffff);
//Set fore ground color as green and back ground color as white.
```

```
fillw(x+1,y+1,x+1+step, y+19);
// Use Command 0x5B to fill the progress area with fore ground color.
clrw(x+2+step,y+1,x+101,y+19);}
//Use Command 0x5A to clear non-progress area with back ground color
```

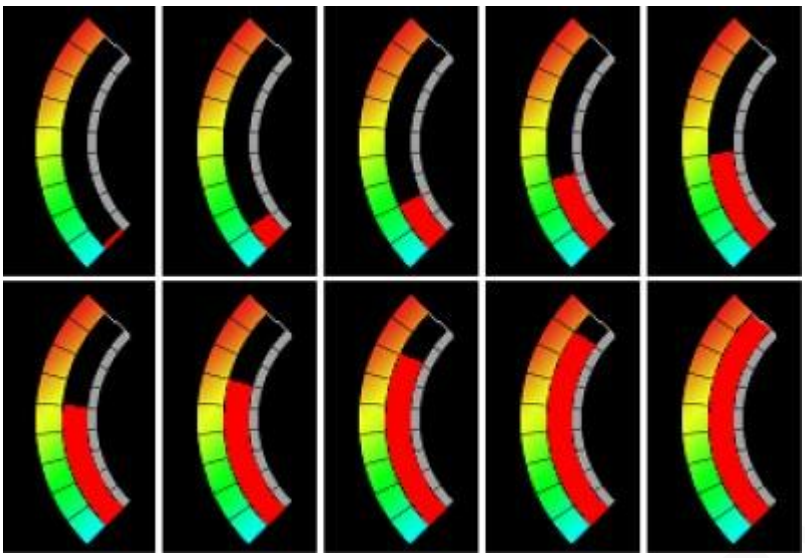
Command	*Data*	**Description**
0x59	$(X_S+Y_S+X_E+Y_E)_K$	Display rectangle box. (X_S, Y_S) is the left top corner and (X_E+Y_E) is the right bottom corner. 
0x5A	$(X_S+Y_S+X_E+Y_E)_K$	Clear the rectangle box.(Fill with background color) (X_S, Y_S) is the left top corner and (X_E+Y_E) is the right bottom corner. 
0X5B	$(X_S+Y_S+X_E+Y_E)_K$	Fill the rectangle box.(Fill with fore ground color) (X_S, Y_S) is the left top corner and (X_E+Y_E) is the right bottom corner. 

2.3 Simulated instrument panel display

In industrial application, the effect of simulated "instrument panel" is often required to make vivid display. As it is related to irregular area, the instrument panel can't be realized by filling the progress bar. We can make all possible display states be a lot of small icons. Then make the icons become one or more pictures before saving the picture(s) in the display module. Afterwards, use Command 0x71 to cut the specific "state" to the appointed display position according to different requirements for display state.

Command	*Data*	**Description**
0x71	PICNUM+X_S+Y_S+X_E+Y_E+X+Y	Cut the area (X_S,Y_S) (X_E,Y_E) of the picture whose index number is PICNUM and display the area at (x,y) on the current screen. (X_S,Y_S): Left top corner coordinate (X_E,Y_E):right bottom corner coordinate

This example illustrates how to realize simulated oil meter gauge. There are 10 different pointer states.



```
// Display simulated oil pressure gauge at (x,y)
oilmeter(int x, int y, unsigned char p)
{int x0,y0;
Txword(0XAA71);       // Command 0x71 for icon cut
Txbyte(0x0A);         // Save the picture at 10th of the flash
X0=(p%5)*108;         //Send x coordinate of the left top corner of the cut icon
Txword(x0);
Y0=(p/5)*240;         //Send y coordinate of the left top corner of the cut icon
Txword(y0);
Txword(x0+107);       //Send x coordinate of the right top corner of the cut icon
Txword(y0+239);       //Send y coordinate of the right top corner of the cut icon
Txword(x);            //Send x coordinate of the display position of the oil meter
Txword(y);            //Send y coordinate of the display position of the oil meter
TxE0F();}             // Send the frame end 0XCC 33 C3 3C
```

Very complex graphics interface can be realized by Command 0x71 and interface break-down & picture design. The interface can also realized by configuration file. Please refer to file 7.2 for more details.

2.4 Realize playback of historical curve using temporary buffer

In actual application, curve of historical record is always needed to be called, therefore, quick points setting (connecting line) function in temporary buffer to realize the playback. Temporary buffer is a RAM area of 40KW(80KB) in the display module. The user can save the data to be displayed in the temporary buffer. When the data is ready, high-speed and synchronous execution of commands (For example, the speed of points setting can reach 0.1Us/point, namely 107points/second) can be realized through calling simple commands.

Besides, as the data is saved in the memory, smooth translation display (Command 0XC101, 0C102) and curve zoom(Command 0Xc103) can be realized when the user changes the starting address of the memory visited by display commands.

Command description related to temporary buffer is as follows:

Command	*Data*	**Description**
0XC0	ADRH+ADRL+ Data0+…+Dat an	Write data into the temporary buffer which ranges from 0x0000-0x9FFF. ADRH:L is the starting address. There is 40Kword. Every address occupies 2 bytes.
0XC1	0x01+ ADRH+ADRL+ Pn_H+Pn_L	Set points in the temporary buffer. ADRH:L is the starting address for setting points. Pn_H:L is the number of setting points. Every point occupies 3 word. There are maximum 13653 points. The format of data in temporary buffer is Psx+Psy+Pixel_Color(Color, MSB)
	0X02+ ADRH+ADRL+ Ln_H+Ln_L	Connect points in the temporary buffer. ADRH:L is the starting address for setting points. Ln_H:L is the number of connecting points. Every line occupies 5word. There are maximum 8191 lines. The format of data in temporary buffer is Xs+Ys+Xe+Ye+Line_Color(Color, MSB)
	0x03+Address +X+Y+Line_N umberAddress +X+Y+Line_N umber+D_x+D is_x+Color	Address: starting address of temporary buffer. X:starting coordinate in X-axis Y:zero position(the lowest point) in Y-axis; The actual position = Y-Ly. Line_Number: The number of lines. Every line occupies 1 word. There are maximum 40960 lines. D_X: point distance. It ranges from 0x01 to 0xFF. The address of temporary address is D_X more for every more line. Address= Address+ D_x; Dis_x: increment of X coordinate. It ranges from 0x01 to 0xFF. X=x+Dis_x. Color: color of lines. This will not change the property of palette. The format of data for connecting lines in temporary buffer is: Ly(2 bytes),Ly is the height of the points.

The following c-codes are for realizing playback of the following three historical curves through temporary buffer.

```
int pa[1000],pb[1000],pc[1000];    // Historical data
int i,j,x;
long adr;
x=40;
for (i=0;i<25;i++);    // Send pa data to temporary buffer
{Txword(0xaac0);       // The command for writing in temporary buffer
Txword(i*120);         // Starting address in the temporary buffer, pa data area0x0000-0x1FFF;
                          Every point occupies 6 bytes.
for (j=0;j<40;j++)
{Txword(x);             //X coordinate
{Txword(pa[i*40+j]);     //Point data
Txword(0xf800);          //Color
X++
if(x>200)                //The window for historical record x=40-200
{(x=40;}}
TxE0F();}                //Send frame end
X=40;
for (i=0;i<25,i++)       //Send pb data to temporary buffer
{Txword(0xaac0);
{Txword(i*120+0x20000);
for(j=0;j<40;j++)
{Txword(x);
Txword(pb[i*40+j]);
Txword(0x07e0);
x++
if(x>200)
{(x=40;}}
TxE0F();}
x=40;
for(i=0;i<25;i++)            //Send pc data to temporary buffer
{Txword(0xaac0);
Txword(i*120+0x4000);
for(j=0;j<40;j++)
{Txword(x);
Txword(pc[i*40+j]);
Txword(0x001f);
x++;
if(x>200)
{(x=40;}}
TxE0F();}

//Check historical record through Command 0XC1
adr=0;                       //Starting address in the temporary buffer
while(KEYOK)
{if(Kcode==1)                //Key-down
{adr=adr+480;
```

```
if(adr＞3000)
{adr=0;}}
if(Kcode==0)                    //Key-up
{if(adr＜480)
{adr=480;
adr=adr-480;}
Piccut(10,40,300,200,400,40,300,200,400);   // Use Command 0x71 to cut and paste the
                                              window of historical curve, meanwhile, clear
                                              the former curve
for(i=0;i＜3;i++)
{Txword(0xaac1);                //Display historical curve
Txbyte(0x01);
Txword(adr+0x2000*i);           //Starting address in temporary buffer pa(i=0)
                                  pb(i=1) pc(i=2)
Txword(160);                    // Width of the window is 160 points
TxE0F();}
KEYOK=FALSE;}
```

2.5 How to design graphic interface similar to Windows style

Due to the following three features of the display module, we strongly recommend the customer to design "fashionable, direct" graphics interface to realize human machine interchange.

★ It is with standard 128MB flash which can be extended to 1GB.
★ Stable and reliable technology for simulated touch panel with excellent accuracy.
★ Flexible and simple serial port commands to realize 65K color display.

Therefore, to design HMI (Human Machine Interface) by our display module becomes much easier. The touch area can be set anywhere. Besides, big storage space for large-sized pictures without being compressed can make your interface become a group of interchange pages. Very simple code can be used to establish links among the display pages, which simplify the program and greatly lower the development difficulty. Pleas refer to the attached example code.

```
// Definition of the links list {current pictures, x0,y0,x1,y1, the pictures to be transferred}
int xdata mlink[][6]={{0,0,0,639,479,2},{0xffff,0,0,0,0,0}};
int i,x,y,n;      // x,y is the touch area
i=0;
while(TCHOK)
{TCHOK=FALSE;     //Clear touch mark
n=milink[i][0];
if(n==0xffff)     //0xffff is the end mark
{break;}
if(n==Pic_n)&&(x>mlink[i][1]&&(y>mlink[i][2]&&(x<mlink[i][3]&&(x<mlink[i][4]))
{Pic_n=mlink[i][5];     //Pick out the position of the next display picture interface
Picdisp(Pic_n);      // Display picture AA 70 Pic_n CC 33 C3 3C
Break;}
i++;}
```

The above C example finished touch links. mlink[][] array example is for a 640*480 display module. When you see the power-on interface (page 0), press any position (x0=0,y0=0,x1=639,y1=479), the second interface will be displayed. The user can add some other links in the array and save the relevant interfaces into the display module. Then complicated touch interface design can be finished. It is also very convenient to realize automatic touch interface transfer by making the configuration file. Please refer to file 7.1 for more details.
For the general touch products, the above simple code can help you finish more than half of the design. In case of future product upgrade or adjustment of the interface sequence, the user just need to change the definition of mlink[][] array. Actually, no professional engineer is required to do this. General graphic designer is enough.

2.6 Real-time upgrade of pictures

In some cases, we need real-time upgrade of some display content on the screen. Command 0x72 for sending data into video memory and Command 0XE2 saving the picture on the current screen into the module are required in this case.
Command format definition:
AA 72 <Address _H:M:L> <Pixel_data0……Pixel _data n> CC 33 C3 3C

AA	**Frame header**
72	Command for sending data into video memory
Address _H:M:L	Starting address of video memory
Pixel_data0……Pixel _data n	Data to be sent into the video memory

Note:
★ Calculation of video memory address
If horizontal pixel (X direction) is H and vertical pixel(Y direction) is V, the video memory address of the dot (x, y) is calculated like this: Adr=(y*H+x).

For example, the video memory address of (150,100) on the screen of display module whose resolution is 640×480 is 100*640+150=64150(0XFA96).

★ Pixel data
Our display module is of 65K color and every pixel dot consists of data of two bytes (16 bits). The display data is in conformity with color palette 5R6G5B and it is sent as MSB (Most Significant Bit). For example, if you need to display a red dot at (150,100), please send the command as follows: AA 72 00 FA 96 F8 00 CC 33 C3 3C.
★ Line width and line wrap
Command 0x72 can send maximum 248 bytes (124 continuous dots). Therefore, when the actual pixel dots are more than 124, the data shall be sent by a couple of frames separately. After line wrap, as the video memory address is not continuous, the coordinate of next video memory address shall be re-calculated before new data is sent.
★ Speed and time for download

The speed of serial port data communication is set by the user. The maximum value is 921600bps. For a H×V picture, the approximate time is
T=(H×V×2)/((B×0.1) ×0.9) second.
Take a picture of 128×160 as an example, the required time is 4 seconds at 115200bps and 0.5 second at 921600bps respectively.
Example: download a picture of 128×160 into display module whose resolution is 640×480 at (x, y) and display the picture at the top left corner.

Relevant C code is as follows:

```
unsigned char bmp[160][256];          // bitmap data
int i,j,k,n;
long adr;                             // video memory address
for (i=0,I<160;i++)                   // to send one line for every cycle
{adr=y*480+x;                         //  to calculate start address, 640×480 as display resolution
K=0;                                  // pointer position of bitmap data
for (;;)
{n=256-k;                             // the data to be sent by this frame, 256 is the byte value of one line of 128 pixel data
if (n==0)
{break;}                              // this line finished and to sent the next line
If (n>248)
{n=248;}                              // every frame contains no more than 248 byte
Txword(0xaa72);                       // 0x72, for direct video memory
Txadr(adr+k);                         // to send the start address
for(j=0;j<n;j++)
{Txbyte(bmp[i][k];                    // to send bitmap data
K++;}}}
```

The format of Command 0XE2 is:
AA E2 <Pic_ID> CC 33 C3 3C
★ <Pic_ID>: ID of the picture to be saved into the display module

Ⅲ Peripherals and auxiliary functions

To lower the complexity of user's system and make the application convenient, the display module integrated some peripherals and auxiliary functions. The basic peripherals include 4*4 (8*8 for some models) matrix keyboard interface, 4 wire touch screen, backlight brightness adjusting, user's database and auxiliary algorithm for MCU.

3.1 Keyboard interface

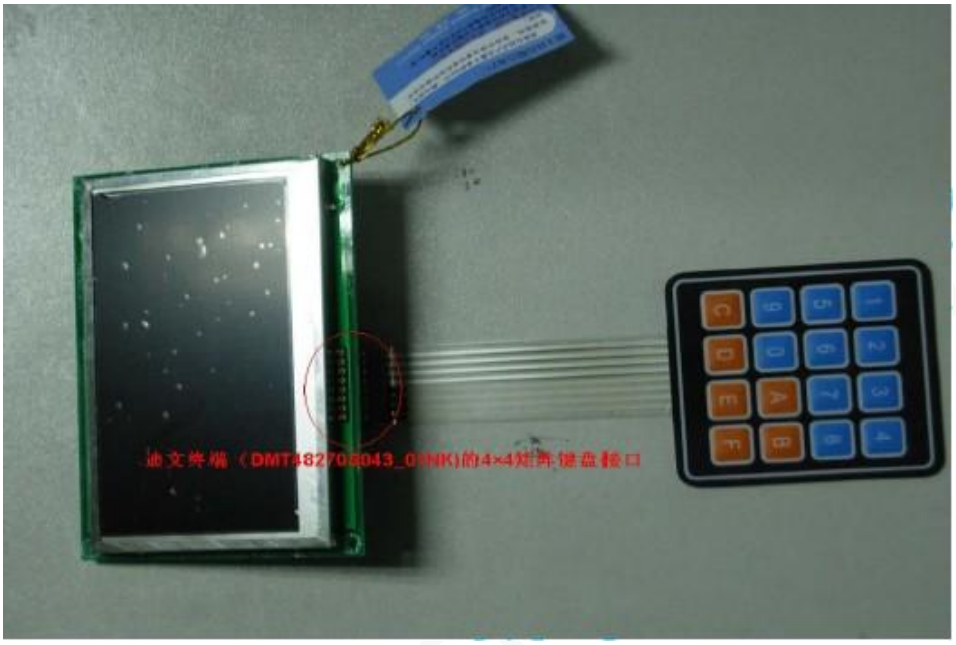
There are two Commands related to keyboard interface: 0x71 (for uploading the key code) and 0XE5(for setting key code).

Command	*Data*	**Description**
0x71	K_Code	Key code will be uploaded automatically when some key is pressed
0XE5	0x55+0XAA+0X5A+0XA5+K0+K63	Set keyboard interface.K0-K63 represent 64 key codes of a 8*8 matrix keyboard. For 4*4 keyboard, mapping is only available for part keys.

Keyboard scanning is finished automatically by the display module. When some key is pressed, corresponding key code will be uploaded. It is recommended to use serial port break to receive key code. Relevant C and ASM51 codes are as follows:

```
// C serial port break program; define sub-array Rx_key[3]. Put the correct key code at Key_code and mark Key_ok.
for (i=0;i<2,1++)
{Rx_key[i]=Rx_key[i+1];}    // Move the receive window for better judgment.
Rx_key[2]=SBUF;             //Put the data received in the end.
if ((Rx_key[0]==0XAA&(Rx_key[1]==0x71)&&(key_ok==FALSE))
  // Receive all codes and no key_code is not processed
{Key_ok=TRUE;     //Successful setting
K_code=Rx_key[2];}  // Save the key code
// ASM51 serial port break program
        MOV   RXKEY0,RXKEY1     //Move receive window
        MOV   RXKEY1, RXKEY2
        MOV   RXKEY2, SBUF
        CLR    RI
        JB      KEYOK,RXKEYE      // Don't receive new key code if there is some key code
                                     unprocessed.
        MOV   A, RXKEY0
        CJNE   A,#0AAH,RXKEYE    // To judge whether the receive is finished
        MOV   A, RXKEY1
        CJNE   A,#71H,RXKEYE
        MOV   KEYCODE,RXKEY2   //Successful setting
SETB    KEYOK
RXKEYE:NOP
```

Note: It is not convenient to directly use key code of hex for users' program; therefore, Command 0XE5 is set. The user can correspond key of keyboard with the key code uploaded to increase the readability of the program. Three steps are required to set the key code by Command 0XE5. Take the keyboard of the following picture as example. The key code of the keyboard corresponds ASCII character.(Key code 0x31 will be uploaded when "1" is pressed).



Step1: put all key codes in order

Send the following command to the display module:

AA E5 55 AA 5A A5 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F CC 33 C3 3C

Step 2: test key code

Press all keys of the keyboard and fill the key codes uploaded into the position of corresponding keys. Then you will have a key code form as the following:

KEY	0	1	2	3	4	5	6	7
Key code in order	0x00	0x01	0x02	0x03	0x08	0x09	0x0A	0X0B
The required key code	0X30	0X31	0X32	0X33	0X34	0X35	0X36	0X37
Key	8	9	A	B	C	D	E	F
Key code in order	0X10	0X11	0X12	0X13	0X18	0X19	0X1A	0X1B
The required key code	0X38	0X39	0X41	0X42	0X43	0X44	0X45	0X46

Step 3: get new key code setting form and download it into the display module.
According to the key codes from step 2, we use "the required key code" to replace "key code in order" and change the key codes that you don't need as 0xFF (or other key codes that are not used, for example 0x00 to improve anti-interference ability), then the key code setting list we need is available. Download it into the display module by Command 0XE5. Afterwards, the key code uploaded is 0x31 once you press key "1".
AA E5 55 AA 5A A5 30 31 32 33 FF FF FF FF 34 35 36 37 FF FF FF FF 38 39 41 42 FF FF FF FF 43 44 45 46 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF CC 33 C3 3C.

3.2 Touch Panel

3.2.1Uploading data format of Touch panel

Relevant commands related to touch panel are as follows:

COMMAND	DATA	DESCRIPTION
0x72	X+Y	Upload the coordinate of the touch area when stop pressing the touch panel.
0x73	X+Y	Upload the coordinate of the touch area when pressing the touch panel. Ten times for a second.
0XE4	0X55 0XAA 0X5A 0XA5	Touch panel calibration
0XE0	55 AA 5A A5 + TFT_ID+Bode_Set+Paral	Set working mode of the display module. TFT_ID/Bode_Set/ Paral: as per Command Description.

3.2.2 Recognition and process of Touch Screen Button

Touch screen button refers to an icon for showing a button on the display. When the users press the icon, your MCU can recognize it is the icon that is pressed and relevant operation will be carried out. The display module will upload the coordinate of the touch position when the users touch any part of the screen. Only when the coordinate is within the area for pre-set button can your MCU recognize it. The calculation method is as follows:
If x0 ﹤ x ﹤ x1 while y0 ﹤ y ﹤ y1, the touch area is valid.
(x0,y0) is the top left corner coordinate and (x1,y1) is the right bottom corner coordinate.

As shown in above picture, the coordinate of the valid button shall be 31 < x < 231 191 < y < 388.

In general, touch position is obtained by serial port break.

Take recognizing touch button of the above picture as an example. The relevant C codes and ASM51 are as follows:

```
// Serial port break service program, make the correct touch code as K_code and mark Key_ok by setting
int xdata kpos [1] [5]={{31,191,231,388,1}};   // the corresponding key code of the button area is 1. Add lines in case of more areas.
int x,y,i, K_code
unsigned char Rx_tch[5];
for (i=0; i < 5; i++)
{Rx_tch[i]=Rx_tch[i+1];}    //Move receive window for better judgment.
Rx_tch[5]=SBUF          //Put the received serial port data to the end
If (( Rx_tch[0]==0xAA)&&(Rx_tch[1])==0x72&&(key_ok==FALSE)
//Receive finished and no key code unprocessed.
{x=Rx_tch[2]*256+Rx_tch[3];       // Convert the received coordinate data into integer.
y=Rx_tch[4]*256+Rx_tch[5];
for (i=0;i < 1;i++)
{if ((x > kpos[i][0]&&(y > kpos[i][1]&&(x < kpos[i][2]&&(y > kpos[i][3]
// Judge whether the click is valid.
{K_code=kpos[4];                //Picl out the responding key code.
Key_code=TRUE                // Marking for successful setting receive.
Break;}}}
```

// Press touch panel of display module. Serial port break program. Read out the coordinate of touch area and recognize key code.

```
MOV   RXAA, RX72        //Move receive window
MOV   RX72,RXXH
MOV   RXXH,RXXL
MOV   RXXL,RXYH
MOV   RXYH,RXHL
MOV   RXYL,SBUF
CLR    RI
JB      KEYOK, RXTCHE      //No receipt of new key code if some key code is not processed
MOV   A,RXAA
CJNE   A,#0AAH,RXTCHE       // To judge whether the receipt is finished
MOV   A, RX72
CJNE   A,#72H,RXTCHE
MOV   DPTR,#KEYPOS       // After receipt is finished, pick out the coordinate and recognize it by checking the list
RXTCH1:MOV    A,#00H
          MOVC   A,@A+DPTR
          CJNE    A,#0FFH,RXTCH2
          LJMP    RXTCHE     //No effective key found when there is 0xFF
RXTCH2:MOV    B,A
          MOV    A,#01H
          MOVC   A,@A+DPTR
          CLR     C
          SUBB    A,RXXL
          MOV     A,B
          SUBB    A,RXXH
          JNC      RXTCH3    //x>x0
          MOV     A,#02H
          MOVC    A,@A+DPTR
          MOV     B,A
          MOV     A,#03H
          MOVC    A,@A+DPTR
          CLR      C
          SUFF     A,RXYL
          MOV      A,B
          SUBB     A,RXYH
```

```
        JNC     RXTCH3  //y>y0
        MOV     A,#04H
        MOVC    A,@A_DPTR
        MOV     B,A
        MOV     A,#05H
        MOVC    A,@A+DPTR
        CLR     C
        SUBB    A,RXXL
        MOV     A,B
        SUBB    A,RXXH
        JC      RXTCH3  //x<x1
        MOV     A,#06H
        MOVC    A,@A+DPTR
        MOV     B,A
        MOV     A,#07H
        MOVC    A,@A+DPTR
        CLR     C
        SUBB    A,RXYL
        MOV     A,B
        SUBB    A,RXYH
        JC      RXTCH3  //x<y<1
        MOV     A,#09H
        MOVC    A,@A+DPTR
        MOV     KCODE,A     //Pick out key code with low bit
        SETB    KEYOK
        LJMP    RXTCHE
RXTCH3  MOV     A,DPL       // Go the next recognizing position
        ADD     A,#10
        MOV     DPL,A
        CLR     A
        ADDC    A,DPH
        MOV     DPH,A
        LJMP    RXTCH1
RXTCHE:NOP
KEYPOS: DW31,191,231,388,1
        DB OFFH             //The end of touch panel area definition
```

Note: If the user needs the touch button can be recognized with some "action"(for example, the

touch button color is changed or the button becomes outstanding), Command 0x73, 0x72 and 0x71 shall be combined as follows:

When the user press touch panel (received Command 0x73), use Command 0x71 to transfer a position hint from a pressed button to the current button.

When the user release touch panel(received Command 0x72), use Command 0x71 to transfer a position hint from a unpressed button to the current button.

The touch panel is not pressed.

The touch panel is pressed. (Receive 0x73)

The touch panel is released after being pressed.(Receive 0x72)

3.2.3 Touch panel calibration

When the user find that the accuracy of the touch panel becomes lower, touch panel shall be calibrated.

Sending touch panel calibration command "AA E4 55 AA 5A A5 CC 33 C3 3C" to the display module through serial port. Then press three white spots in turn according to the on-screen prompt.

3.3 Access to 32MB users' memory

In order to simplify the user's system design and lower development difficulty, a 32MB users' memory area is available in picture memory to make the reading or writing operation through serial port in the memory very convenient. The users don't need to be concerned about the specific operation process (such as format and error correction). This memory is NAND flash with 10 year's life span, allowing 100,000 times' writing and erasing. Combined with DWIN's error controlling technology, it can fully meet the requirement for saving non-volatile for general measurement and control system

Command	*Data*	**Description**
0x90	Tx:0x55+0XAA +0X5A+0XA5+ ADRH:MH:ML: L+Data Rx:'OK'	Write data into user's memory. ADRH:MH:ML:L is the starting address and Data is the data to be saved. The maximum space for the data is 30MB(29.9735MB, 000000-01:DE:FF:FF) which overlays with the 32MB for pictures(the other 2MB is kept by the system); The internal memory is divided into two parts: a. address range: 0x01:DE:00:00-0X01:DE:FF:FF, for total 64KB random access memory. For every writing operation, perform" read back-change-write back". The data without being changed will be protected.Only M600(drive module) supports 64 KB random access memory. As temporary buffer is used for back-up, write random data momory will change the data of the rest 30KW of temporary buffer. b. address range: 0x00:00:00:00-001:DD:FF:FF sequence data memory, for 239 128KB datapage. When it comes to page heading (address=********1 00 00), the current page will be erased. There is no data back-up before erasing, which will not affect the data on the other pages. It fits for continuous and big data storage, such as non-paper record and audio recording. Physical media of the data base is NAND Flash whose life span is 10 years and it allows 100,000 times' writing and erasing.
0X91	Tx:ADRH:MH: ML:L:LENH:L Rx:ADRH:MH: ML:L+LENH:L+ Data	Read data from the appointed address. Len_H:L is the length of the data read out(0x0000 represents 65536). Data is the data read back. 64KB is the maximum length for every time. Display module will not respond to any command when Commands 0x90 and 0x91 are performed.

★ Structure division of 128MB data space

Physical address range	*0x00000000-0x01FFFFFF*	**0X02000000-0X07FFFFFF**
Space	32MB	96MB
Description	Memory for font library. (32 small font libraries of 128KB and 28 big font libraries of 1MB)	Memory for pictures.(The space for every picture shall be bigger than the calculated value for error-correction purpose). 153 full-screen pictures of 640x480 can be saved.

Note: 128MB data memory without user's memory.

PHYSICAL ADDRESS RANGE	0X00000000-0X01FFFFFF	0X02000000-0X05FFFFFF	0X06000000-0X07FDFFFF	0X07FE0000-0X07FFFFFF
Space	32MB	64MB	31.875MB	128KB
Description	Memory for font library	Memory for pictures. 102 full-screen pictures of 640x480 can be saved.	0X00000000-0X01DDFFFF 29.875 MB Sequence data memory	0x01DE0000-0X01DEFFFF 64 KB Random data memory
			User's memory(The actual physical storage space is bigger than the space available for error-correction purpose)	

Note: 128MB data memory with user's memory included.

★ The process of reading user's memory

When receiving the command for reading user's memory, the display module calculates the actual physical range of the memory according to the address and read out data from the memory. After error processing, the data will be sent to the serial port.

Note: The maximum length of the data that can be read out once is 64KB(The corresponding length is 0x0000).

★ The process of writing user's memory

When receiving the command for writing user's memory, the display module will judge it is random memory or sequence memory according to the address. The following steps shall be followed in case of random memory:

a. Read the data in 64KB random memory back to temporary buffer(RAM read back back-up);
b. Erase random data memory;

c. Write the data into the corresponding position of temporary buffer(change);
d. Perform error-correction on the data in the temporary buffer and write the data into random data memory (write-back)
e. The serial port responds "OK", writing operation is finished.

All the above-mentioned steps will be finished by the display module automatically. No interference from the users is required.

Note: Temporary buffer is not available for drive module M100, which therefore doesn't support 64KB random database.

For random data memory, the user can conveniently and randomly change the data of any position in the memory because the display module always backs up the original data.

Writing operation shall be performed as follows in case of random data memory:

a. Judge whether the current address that needs to be written at is the staring address(address:** *0 00 00) of a 128KB section. If yes, perform step b, if no, perform c;
b. Erase a 128KB section;
c. Write one-byte data into the appointed address;
d. Pointer of the address adds 1 to judge whether the writing data is complete. If yes, perform step e next. If no, perform step a;
e. The serial port responds "OK", writing operation is finished.

All the above-mentioned steps will be finished by the display module automatically. No interference from the user is required.

For sequence data memory, the display module doesn't back up the original data. The data will not overlay the written data. The written data can be changed unless it is erased. This method only fits for saving historical record data with sequence.

3.4 Data-processing of the measuring data

In many cases, filter processing for data is required to get rid of interference. "Mean-value filter" is simple and effective algorithm, so it is used widely. In general, there are three steps

★ Collect groups of data
★ Remove the largest value and smallest value.
★ Accumulate and average the rest data

For some simple 8-bit MCU, it is not convenient to perform Mean-value filter processing on typestyle data (such as 12bit A/D result). The biggest difficulty is to remove the largest value and smallest value, especially when there are a couple of largest values and smallest values involved. Due to it is relating to sorting processing and big algorithm, the response will be slow.

In our display module, sorting processing for typestyle is built in. After users send the data into the display module via serial port, the display module will send back the data of ascending order to the user, realizing math coprocessor function of CPU. Please refer to the relevant command as follows:

COMMAND	DATA	DESCRIPTION
0XB0	Tx:0x03+Data_Pack0	Sorting Data_Pack0 is the 2-byte array to be sorted. Data is sent by MSB. Data_Pack1 is the data of ascending sorting.
	Rx:0x03+Data_Pack1	

Example:
Supposing the original data order: 123,100,127,128,119,122,125,130,147,133
The order after sorting in display module: 100,119,122,123,125,127,128,130,133,147

The use removes three values at two sides (3 largest values and 3 smallest values):100,119,122,123,125,127,128,130,133,147
The user averages the rest 4 values and gets the filter result: 125

Of course, the simpler method is to pick out the data in the middle of the sorting as filter result. The following is example C code for filter processing on real-time A/D result. This example can also be regarded as the solution for improving A/D accuracy through sampling(based on the result of counting the average, the speed for sampling is 4 times quicker and A/D accuracy is 1bit higher)

```
// Perform sorting processing for every 32 times A/D result, picking sum of 16 values in the middle as A/D result
// 12 bit A/D, output is 16 bit result(sum of 16 times and the actual accuracy is 14 bit)
unsigned int ad_result, ad_out;       //A/D return result
unsigned char i, *adin;
Txword(0xaab0);                 // Data sorting AA B0 03 Data_Pack CC 33 C3 3C
Txbyte(0x03);
for (i=0;i<32;i++)
{Rd_ad()};                      //A/D transfer, save the result at ad_result
Txword(ad_result);}             //Send A/D result to the display module
TxE0F();}                       // Frame end
Rxstr(*adin);                   // Receive sorting result from display module
Ad_out=0;
for (i=16;i<32;i++)             // Accumulate the 16 values in the middle calculate
{ad_out=ad_out+(*(adin+2*i+3))*256+*(adin+2*i+4);}
```

Ⅳ. Simplify design with configuration file

4.1 Automatic touch interface transfer

For the display module with touch panel, configuration file can be downloaded into the display module in advance to decrease code workload of the users. Then the mode of touch interface transfer is set as auto-transfer to realize user's intervention-free function.

Development process:

Step 1: Design the interfaces having the same resolution with the display module and download them into the display module. For example, if the display module is DMT64480S057_11WT, the resolution of the designed interfaces shall be 640*480 dot matrix.

Step 2: Make the configuration file.

The configuration file is a binary file consisting of maximum 8,192 touch commands. Each touch command contains 16 bytes. The definition is as follows:

STARTING ADDRESS	DATA LENGTH (BYTE)	DEFINITION	DESCRIPTION
0x00	2	Pic_Now	Number of the picture on the current screen. If the high byte of Pic_Now is 0XFF, it means the end of touch commands.
0x02	4	xs,ys	Left top coordinate of valid touch area.
0x06	4	xe,ye	Right bottom coordinate of valid touch area.
0x0A	2	Pic_Next	Number of the next picture after valid touch area is pressed. If the high byte of Pic_Next is 0XFF, it means no touch interface transfer.
0X0C	2	Pic_Cut	Number of the picture with animation. If the high byte of Pic_Cut is 0XFF, it means no picture with animation.
0X0E	2	Touch_Code	The touch code uploaded after effective touch area is pressed. If the high byte of Touch_Code is 0XFF, it means no upload of touch code. If the high byte of Touch_Code is 0XFE, it means the uploading data is indexed into 0x1A configuration file. Then the low byte of Touch_Code is the index ID. For 0x1A configuration file, every index length is fixed as 256 bytes and the first byte is the valid length of the index.

The process of making configuration file is actually the process of designing the interfaces and arranging interface transfer. The configuration file can be complied by UltraEdit or some compiling

system, such as C51 or ASM51.

Take ASM51 as an example to illustrate how to compile configuration file.

Example of touch interface transfer for DMT80480S070_02WT:

Press the button 

of picture numbered 19. The picture numbered 20 will be displayed and touch code (0x0031) will be uploaded.

When some button is pressed, the button effect will be the same as that of picture numbered 2. Blue characters belong to valid touch area.

```
ORG 0000H
DW 19,654,195,792,292,20,2,1     //A touch command
DW 0FFFFH                        // All touch commands finished
END
```

This is the example for one touch button and one touch interface transfer. The user can add corresponding commands in case of more touch interface transfers.

Change the finished configuration file *.ASM into *.hex file by "ASM51.exe" or some other compiler.
Change the *.hex file into *.bin file by the software "hexbin.exe".

Step 3 Download the *.bin file into the display module.
Use Command 0XF2 to download the*.bin file into display module at 0X1E.

Step 4 Set the mode of touch interface transfer as automatic transfer
Use Command 0XE0 and set Para1.5,0x20 as 1. Then when being pressed, the display module will no longer upload the coordinate. Instead, it will perform touch interface transfer automatically and upload the touch code defined by the user.

Step 5 Repeat Step 2 and Step 3 for a couple of times to test whether the test interface transfer is accurate.

4.2 Call different icons for display

Command 0x71 for icon cut can make the user cut one area of any picture saved in the display module and display the area at appointed position on the current
screen. In actual applications, it is not convenient enough for the users. Therefore, Command 0x99 and 0x1D are added to make calling icons like calling text.

The definition of the Command 0x99 is

command	Data	Description
0x99	$(x,y,Icon\_ID)_0$+……+$(x,y,Icon\_ID)_n$	（x,y）is the top left coordinate of icon that will be displayed; Icon_ID is the index ID in Icon library.

Configuration file of icon library is a binary file consisting of maximum 13,107 icon definition. ICON_ID ranges from 0000H to 3332H. It starts from 0. For example:
DW 01,00,00,100,100 ICON_ID address is 0000H.
DW 02,00,00,100,100 ICON_ID address is 0001H.

Each icon definition contains 10 bytes as follows:

Starting address	length（Byte）	Definition	Description
0x00	2	Pic_ID	Picture ID of the saved Icon
0x02	4	x_s,y_s	The left top coordinate of icon area
0x06	4	x_e,y_e	The right bottom coordinate of icon area

When receiving command 0x99, the display module shall carry out it as follows:
★ Cut Pic_ID（Xs,Ys）（Xe,Ye）from position Icon_ID×10 in configured file 0x1D
★ Display the cut icon at （X,Y）on the current screen.
★ Display the next icon.