Data Processing and Character Display Based on STM32

Xiaoyan Wang[1,a,*], Wenhao Xie[1,b]

[1] School of Science, Xi'an ShiYou university, Xi'an, 710065, China

[a] 18512262@qq.com, [b] 1609632928@qq.com

*corresponding author

Keywords: STM32, Character display, data structure

Abstract: HMI is passed between people and computers, a medium of exchange and dialogue interface information, indispensable in industrial automation systems. Based on the STM32 processor, it encapsulates line, characters and other display functions, to achieve the data processing and LCD graphic display, the program can support Chinese characters, ASCII character display that supports multi-page display, configuration page information through a DB, support cursor and simple graphical display. The program is simple and convenient, and has better practicability.

1. Introduction

Industrial HMI is in industrial applications interface, is an important tool used to control the industrial field personnel monitoring device controller, connected to the PLC, inverter, governor, instrumentation and other industrial equipment, and written by mouse, keyboard, touch screen input methods to address data resources within the industrial equipment, while a variety of ways to display real-time data and historical data to achieve information interaction of man and machine[1,2].

Body frame character is through the use of man-machine interface configuration software on the PC to write the configuration screen, the configuration screen data in a format via serial download to the embedded controller, the controller will save the downloaded data to the flash memory, read from the flash memory when needed relevant data, and to reproduce the configuration on the display unit screen[3].

2. Design purpose

The data from the PLC to Tx_Disp(Structure) finishing three parts, one for the string output StringBufer; A saved parameter information (Including i and a) UpdateTable entered last control character information FlickerTable. Timer 1 will be based on the description of the modified value within FlickerTable StringBuffer, the value of Timer 2 will StringBuffer sent to LCD, the screen is refreshed regularly. When you open a cursor, that is, when a user enters the modified state, the program determines the location of each input field based on the description UpdateTable within a certain region of the cursor in the input jitter. Note: Enter the two situations: A D-type data input, each character can be modified independently of its domain, The idea of the program are: to get the current cursor position; Remove the response from the StringBuffer in ascii code, according to the keys after the string and then modify the domain records domain UpdateTable taken in; Another case is the character list, the domain is the index of the list changes, so you have to call the modified output function updates the value of StringBuffer[4]. As the picture shows:

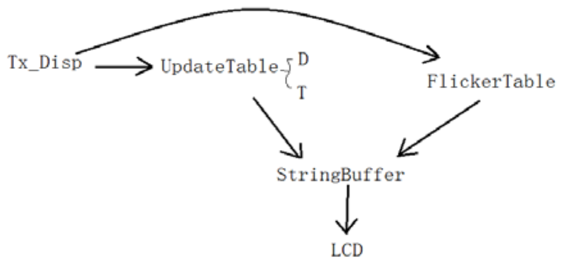

Figure 1: The overall structure

3. The structure definition

```
// Data from the plc
    struct Tx_Disp_Struct{
              uchar   offsetr1;        // Ordinate offset
              uchar   offsetr2;
              uchar   offsetr3;
              uchar   offsetr4;
              uchar   Row[maxRowNum][maxRowLength];
    // Configuration information about rowNum line, which has rowLength characters, including
    constant characters and ASC, and the beginning of the format string %%
              uint            Var[maxVarNum];
  // The first varNum a format string corresponding Var, Var each of indefinite length
                          }Tx_Disp;
     // Data from the plc
    // Box Description
    struct G_Arr_Struct{
              uint x1; // Graphics left corner abscissa unit: [pixel]
              uint y1; // Graphics ordinate the upper left corner, unit: [Page]
              uint x2; // Lower right corner of the graphic horizontal axis, unit: [pixel]
              uint y2; // Graphic bottom right corner of the vertical axis, unit: [Page]
    }G_Arr;
      // Box graphic display
      struct Grph_Disp_Struct{
              uchar   GNum;                              // Graphics Box Number: 0-4
              struct G_Arr_Struct    G_Arr[maxBoxNum];
```

```
            }Grph_Disp;


        // Enter the area to be described, that is, "Enter to update the list UpdateTable"
struct UpdateTable_Struct{
                uchar   Row_Type;
          // 4 represents the current row area where the high and low of four type int variables
          indicating the current or the character list
                uchar   Column;
        // High four starting column, the low four end columns altogether 16 optional
                uchar   VarIndex;                    // Variables Location plc structure
                uint    Var;                         // The current variable Var
                char    format[5];                   // Format string
                uint    r1;
                uint    r2;
          }UpdateTable[maxVarNum];
```

4. Text Table Interface

Single list data structure is as follows:

```
    List0
      {int  EleNum; // The total number of list elements
       int  TextLen;// Each text length [bytes], ranging from 1 to 16
       String Text01[TextLen] ;// The index number = Text 1
       …
       String Text?[ TextLen] ;/ / Ref =? Text}
```

The total length of a single list of data is variable. In this tentative list of individual total length over 320 bytes (equivalent to 20 index * 16 bytes). The interface between the application layer and the underlying structure of a single list for the interface. When the need to use multiple lists, controlled by the application layer in turn will send one of these lists to the bottom. Up to more than ten list. The underlying order in which they will in turn be sent to these lists Automatic numbered: 0 to the 9th.
For example, from PC Incoming:

as well as:

```
Tx_Disp.Var[0] = 1235;
```

```
 Tx_Disp.Var[1] = 4578;
 Tx_Disp.Var[2] = 5689;
 Tx_Disp.Var[3] = 1;
 Tx_Disp.Var[4] = 2;
 Tx_Disp.Var[5] = 2;
 Tx_Disp.Var[6] = 0;
Tx_Disp.Var[7] = 2;
```

You will get:Screen output： such as 12.35 and can be modified to accept cursor; elsewhere flashing, frame and so on.

5. Main Function Description

void FlickerTableToDisplayBuffer();

C type of the variable from FlickerTable filled StringBuffer. If the C type corresponds to the variable Var is 1, then the call to fill the flashing characters, and then the next time you fill the space, and so on.Thus, at the timing when this function call results showed flashes; if C corresponding to the type of variable to 2, then fill the space, namely to hide the effects.

void GetCurrentUpdateTableIndex();

According to the current cursor position to give the line "Enter to update the list UpdateTable" where (UpdateTable index).Specifically: the first progressive scan UpdateTable; if the current cursor position within the scope of a variable in UpdateTable corresponding position (on the same line, the cursor between the input area starting column and ending columns), UpdateTable extracted from the current type, if the D type, save the address of the current character; If the T type, then save the text of the list of index numbers, characters, and the starting position of the text starting position. Finally, save where UpdateTable line number (index) and exit.

6. Conclusion

STM has rich peripheral interfaces, which can be very convenient for data processing, data communication, improve the flexibility of the system design, make the system easy to upgrade and expand. This system development cycle is short, the operation is stable and reliable performance, easy to use, the system USES C language program design, its very good portability, modular thinking can be flexible extend other functions in the program, can satisfy the demands of different display.

References

[1] Han Jinghai, Wang Rui Cortext - M3 development technology and the implementation [M]. Xi 'an:Xian university of electronic science and technology press, 2013

[2] Wu Junpeng, Zhang Guoyin, Yao Aihong etc, Based on ARM embedded system design of experiment and practice course [M]. Beijing: tsinghua university press, 2011

[3] Liu Jinxing, Li Hongwen. Liquid crystal display control module based on microprocessor [J], LCD to display , 2011 26（1）

[4] Li Ke. Based on the Cortex_M3 character of man-machine interface design and implementation [Master degree theses of master of], wuhan: huazhong university of science and technology, 2013