Summary

1. DESCRIPTION

In this project, you will learn how to work with a 14-segment alphanumeric display using two-cascaded Shift Register.

This solution is widely used to economize microcontroller pins. If you choose not to use the Shift Register then you will need 17 pins available.

When we work with electronics projects, we usually do not have this amount of pins available, so one of the solutions is the use of drivers or Shift Register to serialize the sending to display.

2. HARDWARE

2.1. Display

The display used in this project has 4 digits and 14 segments, and segments DP1 and DP2. This type of display is widely used in home appliances and can be easily purchased on internet. As there are many manufacturers, may be a difference in the distribution of the output pins from one manufacturer to another, but the operating mode is the same for all. Just adapt your circuit or modify the software.

- The display has 18 pins. Pin 6 is not used. To turn ON and OFF the digits, pins 1,2,3,4 and 5 are used.
- Drive the segments A, B, C, D, E, F, G1, G2, H, J, M, N, pins 17,15,16,11,18,7,12,13, 14,8,10 and 9 respectively.
- Drive the segments K1, K2, K3, K4, L1, L2, L3, L4, DP1 and DP2, the pins 17,15,16,11,18,7,12,13,14 and 8 are used.

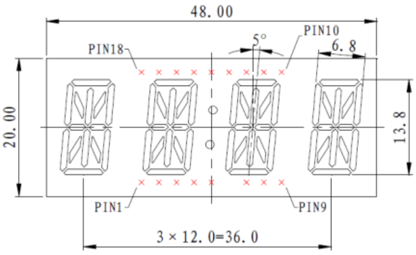


Figure 1. Display dimensions.

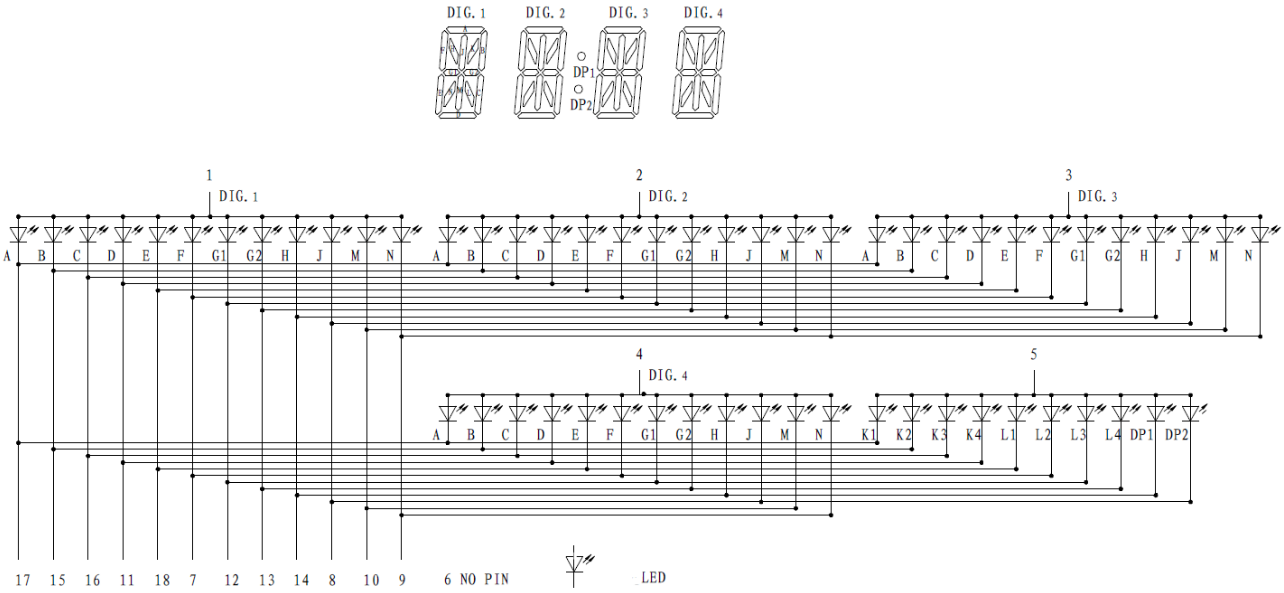


Figure 2. Display scheme.

The display is of the Common Anode type. It means that it needs external power and cannot be driven directly by the microcontroller. (Figure 3).

- "DIG1", "DIG2", "DIG3", "DIG4" are pins that (enable / disable)
- Letters 'A', 'B', 'C' ... 'M' are responsible for triggering their respective segment.

As an example, let us say that you would like to write the number "0" on digit 2 of the display. So you need to enable segments 'A', 'B', 'C', 'D', 'E', 'F' and also enable pin 2 on the display.

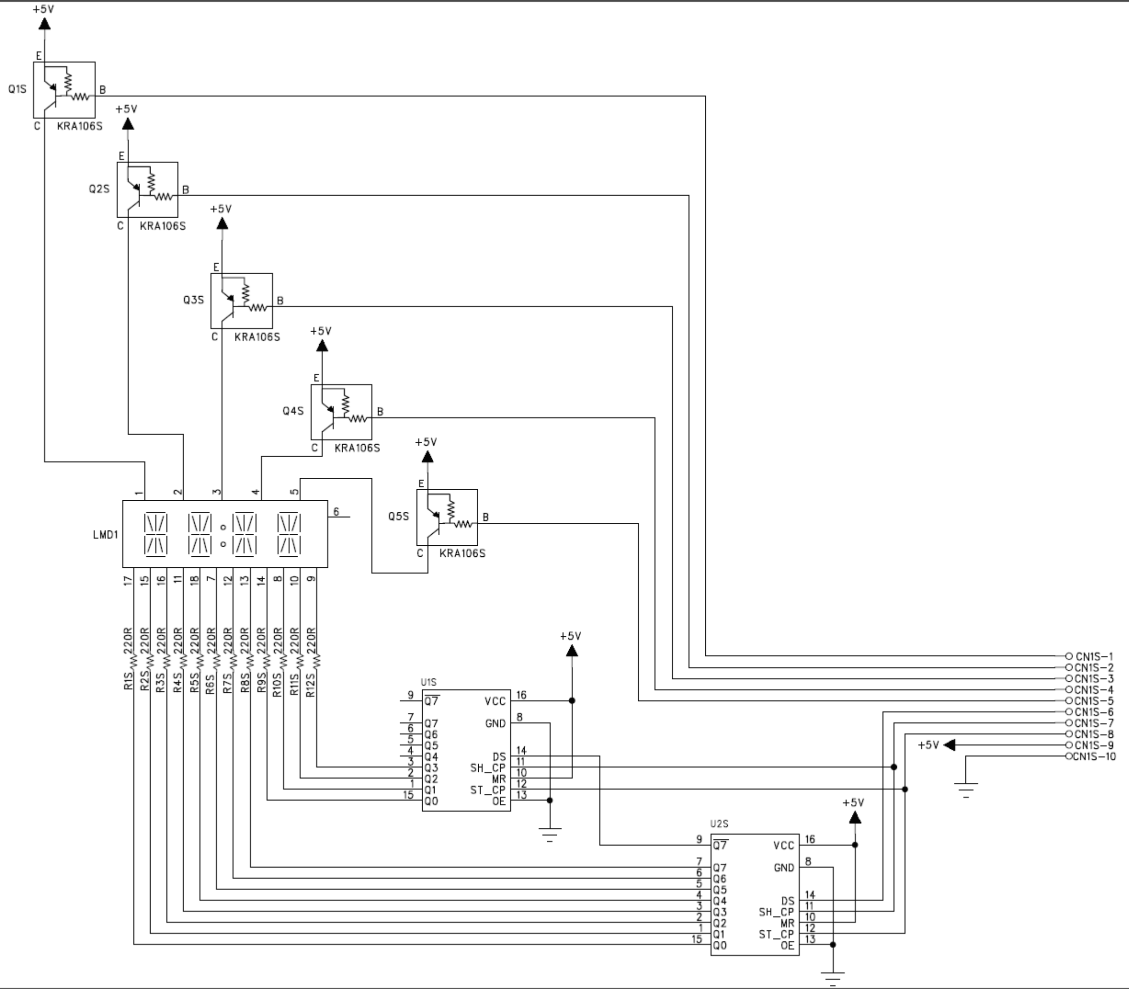


Figure 3. Complete schematic.

No problem if you do not find the transistor with the same manufacturer code. Any PNP type transistor that supports "If" current of at least 40mA should work in this circuit.

It is also possible to use a display of the Common Cathode type, in which case the transistors Q1S, Q2S, Q3S, Q4S and Q5S can be removed from the circuit and the microcontroller's output pins will power supply the display.

3. SHIFT REGISTER 74HC595

Shift register controller used is the 74HC595 family, with 8 bits of serial input and 8 bits of parallel output.

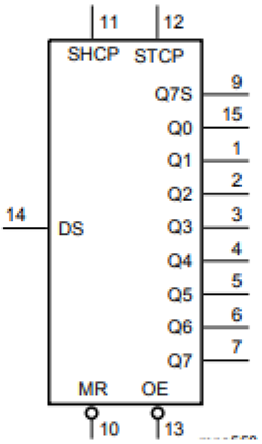
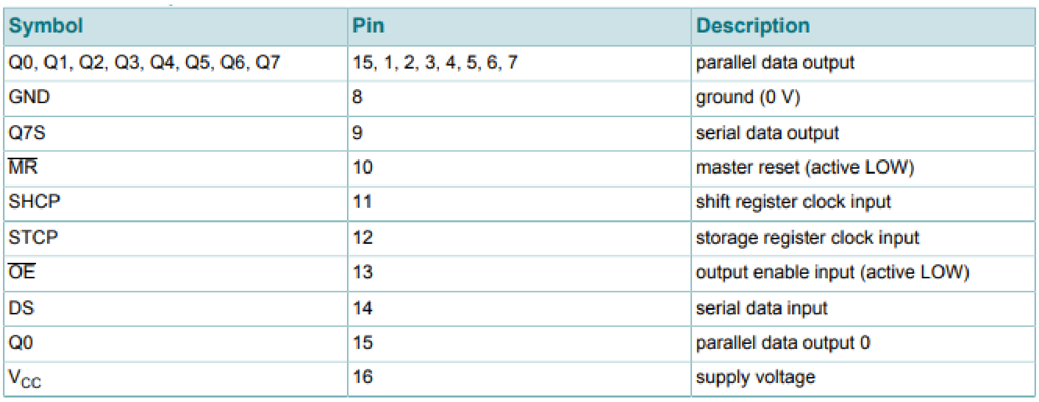


Figure 4. 74HC595.



Symbol	Pin	Description
Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7	15, 1, 2, 3, 4, 5, 6, 7	parallel data output
GND	8	ground (0 V)
Q7S	9	serial data output
$\overline{\text{MR}}$	10	master reset (active LOW)
SHCP	11	shift register clock input
STCP	12	storage register clock input
$\overline{\text{OE}}$	13	output enable input (active LOW)
DS	14	serial data input
Q0	15	parallel data output 0
V_{CC}	16	supply voltage

Table 1. Shift Register pin-out description.

The flow chart below shows the operating mode of the component for our application, 8 bits of serial input to 8 bits of parallel output.

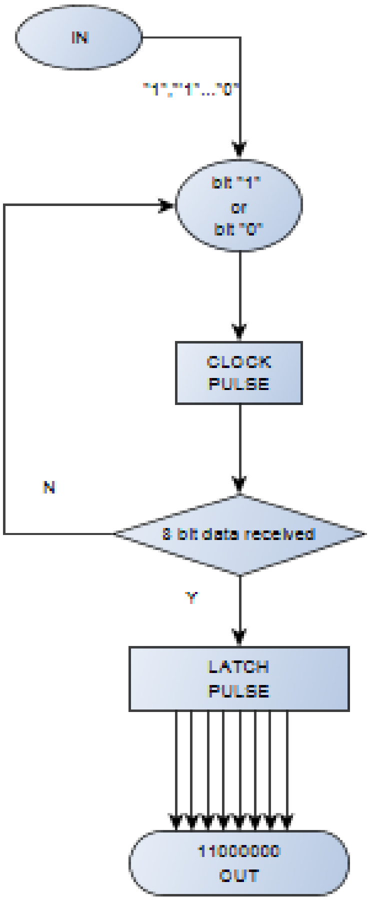
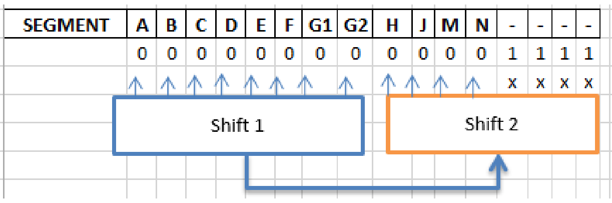


Figure 5. Fluxogram input serial output 8bits parallel.

When a data (bit) is sent to the DS pin, we must carry out a clock (SHCP) from the logic low to high, by doing this the data is stored in component’s internal buffer. When the storage of the 8 bits in the buffer is finished, we send a pulse on the latch pin (STCP). Thus, we will have 8 bits in the parallel output of the component.

In this project we will work with 12 segments (segment 'A' to segment 'N'). As commented before the shift register used is limited to 8 bits. To solve this problem we put two components connected in "cascade".



SEGMENT	A	B	C	D	E	F	G1	G2	H	J	M	N	-	-	-	-
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
													x	x	x	x

Figure 6.Shift Registers output configuration segments.

Above picture is possible verify that the last 4 bits of the controller are not used. On hardware, these 4 bits are not connected and on software to facilitate software implementation, opted to work with variables of 16 bits.

// pins configured in the software. Library "**driver_shift_register**":

- SHCP -> **CLOCK**
- DS -> **DATA**
- STCP -> **LATCH**

// settings made on the ***hardware***:

- MR -> Reset pin(Default state = active LOW)
- E -> Enable / disable pin (Default state = active LOW)
- Q0: Q7 -> Data output pins.
- DS -> Serial Data Input
- Q7S -> Serial Data Output

When in "cascade" we must connect pin Q7S of the first controller to pin DS of the second controller.

4. FIRMWARE

Firmware is divide in 4 main files:

+ display_handler
+ driver_shift_register
+ main
+ timer

4.1. Display_Handler

In this file the routines are implemented:

• LedsMultiplexer ():
// multiplex the display digits;

• Display_Numbers ():
// displays numbers from 0 to 9;

• Display_Alphanumeric ():
// displays characters from 'A' to 'Z'.

```
                         /*display_handler.c*/
/*********************************************************************
**********
******************************* Includes
***************************************
**********************************************************************
*********/
#include "display_handler.h"
#include "definitions.h"
/*********************************************************************
**********
******************************* Definitions
************************************
**********************************************************************
*********/
// digits
typedef struct{
           uint16_t dig1;
           uint16_t dig2;
           uint16_t dig3;
           uint16_t dig4;
           uint16_t dig5;
}DigiTypedef;

/*********************************************************************
**********
**************************** Local variables
***********************************
**********************************************************************
*********/
// struct variable declaration
DigiTypedef Disp;


/*********************************************************************
**********
***************************** Public Functions
*********************************
**********************************************************************
*********/

/*--------------------------------------------------------------------
--------
                DIGITS POSITION
----------------------------------------------------------------------
-----

       -----   -----               ----      ----
      /    /   /    /     O       /    /    /    /
     -----    -----               -----     -----
   /    /   /    /      O      /    /    /    /
  -----    ------             -----     -----
 DIG1     DIG2       DIG5     DIG3     DIG4


 DIG1 - DIG2 - DIG 3 - DIG4: data digits
 DIG 5 - alphanumeric segments and colon
----------------------------------------------------------------------
------*/
```

```
/*********************************************************************
**********

* Function Name: LedsMultiplexer
* Description  : This function multiplex display digits 1,2,3,4 and 5
call this function inside timer interrupt of 1ms
* Arguments    : None
* Return Value : None
**********************************************************************
*********/
void LedsMultiplexer(void){
           // switch state
           static uint8_t disp_stm = 0;
           //
           switch( disp_stm ){
           case 0:
                      // Turn OFF display digit 5
                      DIG5 = 1;
                      // Send to Shift Register
                      send_data_to_shift_register( Disp.dig1 );
                      // Turn ON display digit 1
                      DIG1 = 0;
                      // Change switch state to next position
                      disp_stm++;
                      break;

           case 1:
                      // Turn OFF display digit 1
                      DIG1 = 1;
                      // Send data to shift register
                      send_data_to_shift_register( Disp.dig2 );
                      // Turn ON display digit 2
                      DIG2 = 0;
                      // Change switch state to next position
                      disp_stm++;
                      break;

           case 2:
                      // Turn OFF display digit 2
                      DIG2 = 1;
                      // Send data to shit register
                      send_data_to_shift_register( Disp.dig3 );
                      // Turn ON display digit 3
                      DIG3 = 0;
                      // Change switch state to next position
                      disp_stm++;
                      break;

           case 3:
                      // Turn OFF display digit 3
                      DIG3 = 1;
                      // Send data to shift register
                      send_data_to_shift_register( Disp.dig4 );
                      // Turn ON display digit 4
                      DIG4 = 0;
                      // Change state to next position
                      disp_stm++;
                      break;
           case 4:
                      // Turn OFF display digit 4
                      DIG4 = 1;
                      // Send data to shift register
                      send_data_to_shift_register( Disp.dig5 );
                      // turn ON display digt 5
                      DIG5 = 0;
                      // change switch state to first position (back to
init)
                      disp_stm = 0;
                      break;
           }
}

/*********************************************************************
**********
* Function Name: Display_Numbers
* Description  : This function increment data shown on display digits
* Arguments    : None
* Return Value : None
**********************************************************************
*********/
void Display_Numbers(void) {
           static unsigned char i = 0;
           // Lookup Table
           uint16_t const anode_14seg_table[]={
           //-----------------------------------
           //  NUMBERS
           //-----------------------------------
           //- HEX --  Number --   TABLE POSITION
                0X03FF, //   0   |   0
                0xFF9F, //   1   |   1
                0x24FF, //   2   |   2
                0x0CFF, //   3   |   3
                0x98FF, //   4   |   4
                0x48FF, //   5   |   5
                0x40FF, //   6   |   6
                0x1FFF, //   7   |   7
                0x00FF, //   8   |   8
                0x08FF, //   9   |   9
           };

           // once reach number 9
           if( i > 9 ){
                      // back counting from 0
                      i = 0;
           }

                      // Display digit 1
                      Disp.dig1 = anode_14seg_table[i++];
                      // Display digit 2
                      Disp.dig2 = anode_14seg_table[i++];
                      // Display digit 3
                      Disp.dig3 = D_OFF;
                      // Display digit 4
                      Disp.dig4 = D_OFF;
                      // Display digit 5
                      Disp.dig5 = D_OFF;
}
```

```
/*********************************************************************
**********
* Function Name: Display_Alphanumeric
* Description  : This function increment data shown on display digits
* Arguments    : uint16_t D1, uint16_t D2, uint16_t D3, uint16_t D4,
uint16_t D5
* Return Value : None
**********************************************************************
*********/
void Display_Alphanumeric(uint16_t D1, uint16_t D2, uint16_t D3,
uint16_t D4, uint16_t D5){
           Disp.dig1 = D1;
           Disp.dig2 = D2;
           Disp.dig3 = D3;
           Disp.dig4 = D4;
           Disp.dig5 = D5;
}
/* EOF */
```

Function *Display_Alphanumeric()* parameters D1, D2, D3 and D4 receive the characters that will be displayed. Parameter D5 is used to indicate which if segment 'L' or segment 'K' will be used.

For example:

To display the word "AREA" on the display, we will place the letter ‘A' on digit 1, letter 'R' on digit 2, letter 'E' on digit 3 and the letter ‘A' on digit 4. Except letter R, the rest can be displayed using only 8 main segments ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g1’, ‘g2’.

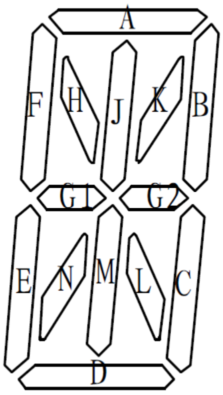


Figure 7. Segments distribution on each digit.

To write the letter R we use the segment ‘L’'. Since we want to show the letter 'R' on display 2nd digit, then we need to send a, b, c, e, f, g1, g2, and segment 'L2'.

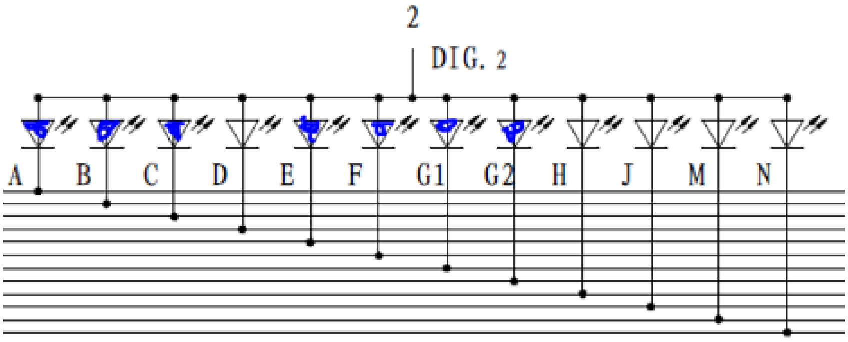


Figure 8. Digit 2 segments used to display letter ‘R’.

Display pin number 5 controls segment ‘L2’.

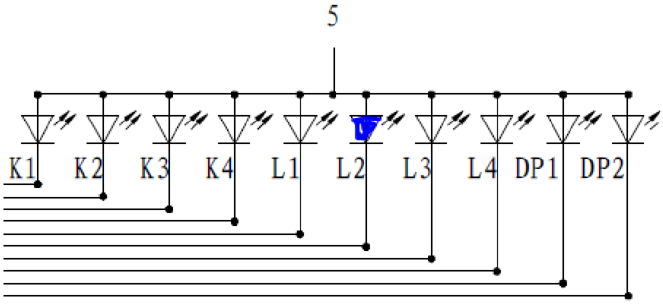


Figure 9. Alphanumerical segment L2.

```
                        /* display_handler.h */

#ifndef DISPLAY_HANDLER_H_
#define DISPLAY_HANDLER_H_

/*********************************************************************
*********/
/******************************* Definitions
**********************************/
/*********************************************************************
*********/
// MCU OUTPUT
// insert here the pins according to your microcontroller
// remeber to define as OUTPUT
//                                                   +---------
//                                                   |        |
#define DIG1               (P12_bit.no0) // ----> |1|2|:|3|4|   |
#define DIG2               (P4_bit.no1)  // ---------+   | |    |
#define DIG3               (P1_bit.no0)  // -------------+ |    |
#define DIG4               (P14_bit.no7) // ---------------+    |
#define DIG5               (P14_bit.no6) //---------------------+

/*
 * /* Overview of display 14-seg
 *
 * EXTERNAL              INTERNAL
            A
           _____         H   j   k
        F |\ | /| B       H  j  k
          | \|/ |            j
            - -           G1  G2
        E | /|\ | C          m
          |/_|_\|         n  m  l
                         n   m   l
             D
/////////////////////////////////////////////////////////////
 common anode segments will be ON when '0' and OFF when '1'
 example letter A :
            _____
           |      |
           |      |
             - -
           |      |
           |      |

a | b | c | d | e | f | g1 | g2 | h | J | m | n | - - - -
0   0   0   1   0   0   0    0    1   1   1   1   1 1 1 1
*/
```

```
#define LETR_A                   (0x10FF)

#define LETR_B                   (0x0E9F)
#define LETR_C                   (0x63FF)
#define LETR_D                   (0x0F9F)
#define LETR_E                   (0x60FF)
#define LETR_F                   (0x70FF)
#define LETR_G                   (0x42FF)
#define LETR_H                   (0x90FF)
#define LETR_I                   (0x6F9F)
#define LETR_J                   (0x87FF)
#define LETR_K                   (0xFF9F)
#define LETR_L                   (0xE3FF)
#define LETR_M                   (0x937F)
#define LETR_N                   (0x937F)
#define LETR_O                   (0x03FF)
#define LETR_P                   (0x30FF)
#define LETR_Q                   (0x03FF)
#define LETR_R                   (0x30FF)
#define LETR_S                   (0x48FF)
#define LETR_T                   (0x7F9F)
#define LETR_U                   (0x83FF)
#define LETR_V                   (0xF3EF)
#define LETR_X                   (0xFF6F)
#define LETR_W                   (0x93EF)
#define LETR_Y                   (0xFF5F)
#define LETR_Z                   (0x6FEF)

#define SEG_K1                   (0x7FFF)
#define SEG_K2                   (0xBFFF)
#define SEG_K3                   (0xDFFF)

#define SEG_L1                   (0xF7FF)
#define SEG_L2                   (0xFBFF)
#define SEG_L3                   (0xFDFF)
#define SEG_L4                   (0xFEFF)

#define D_OFF                    (0xFFFF)  // Display OFF
#define COLON                    (0xFF3F)  // :

/*********************************************************************
*********/
/**************************** Public Functions
********************************/
/*********************************************************************
*********/
void LedsMultiplexer(void);
void Display_Numbers(void);
void Display_Alphanumeric(uint16_t, uint16_t, uint16_t, uint16_t,
uint16_t);

#endif /* DISPLAY_HANDLER_H_ */
```

4.2. Driver_Shift_Register

This file contains the control functions of the Shift Register

- send_data_to_shift_register ():

  // this function receives 16 bits instruction and separates by bit

- Store_Data ():

  // this function stores each bit in shift register buffer;

```
                      /*driver_shift_register.c*/
/**********************************************************************
********
************************** Includes
*****************************************
***********************************************************************
*******/
#include "driver_shift_register.h"

/**********************************************************************
*********
************************** Global Functions
***********************************
***********************************************************************
********/

/**********************************************************************
**********
* Function Name: send_data_to_shift_register
* Description  : This function get data then serialize it to be sent
out to shift
* register
* Arguments    : None
* Return Value : None
***********************************************************************
*********/
void send_data_to_shift_register(uint16_t Data)
{
   uint16_t Buffer  =  0x0000;               // unsigned int 0x00;
   uint16_t Mask    =  0x0001;             // unsigned int
0b0000000000000001
   uint8_t bit      =  0;      // static

// Separates each bit and stores it in the Buffer
        for ( bit = 0 ; bit <= 15 ; bit++){
              Buffer = Data & Mask;
              Store_Data(Buffer);
              Mask = Mask << 1;
        }

        //  LATCH  PULSE - this will inform the shift register end of
information
        LATCH = 0;
        LATCH = 1;
        // now shift register will send 16bit package to display
}

/**********************************************************************
*********
************************** Local Functions
***********************************
***********************************************************************
********/

/**********************************************************************
**********
* Function Name: Store_Data
* Description  : This function send bit to shift register DATA pin
* register
* Arguments    : None
* Return Value : None
```

```
**********************************************************************
*********/
void Store_Data(uint16_t bit_value){  // Bit Test
           // if bit is 0
    if (bit_value == 0)
           // send 0 to shift register DATA pin
      DATA = 0;
    else
           // send 1 to shift register DATA pin
      DATA = 1;

    //  Clock Pulse to store data into register
// CLOCK PULSE
   CLOCK = 0;
   CLOCK = 1;
}/* End of File */
```

/* driver_shiftt_register.h */

```
#ifndef DRIVER_SHIFT_REGISTER_H_
#define DRIVER_SHIFT_REGISTER_H_
/*
######################################################################
#########
Cascade Shift Register Connections

    74HC595
    ------------
 --| Q7'        |--
 --| Q7         |--
 --| Q6         |--
 --| Q5         |--
 --| Q4      DS |--*********
 --| Q3   SH_CP |-- ++++++ * +++++++++++++++++++++++++++++++++++++++
 --| Q2      MR |-- 5V     *                                        +
 --| Q1   ST_CP |-- ------ * ---------------------------------      +
 --| Q0      OE |-- GND    *                                  |     +
    ------------           *                                  |     +
                           *           -------------          |     +
                           **********-| Q7'          |--      |     +
                                     -| Q7           |--      |     +
                                     -| Q6           |--      |     +
                                     -| Q5           |--      |     +
                                     -| Q4        DS |-- DATA |     +
                                     -| Q3     SH_CP |-- +++++++++++
CLOCK
                                     -| Q2        MR |-- 5V   |
                                     -| Q1     ST_CP |-- ----- LATCH
                                     -| Q0        OE_|-- GND
-------------------------              --------------
Signal | Pin of MCU RL78
-------------------------
 DATA   | P15 - No.23
 CLOCK | P14 - No.24
 LATCH | P13 - No.25
-------------------------
######################################################################
##########
*/
/*********************************************************************
**********
```

```
************************** Macro Definitions
***********************************
**********************************************************************
*********/
// definition of MCU PINS connected to shift register
// set as OUTPUT

// define DATA as microcontroller pin P1.5
#define DATA          (P1_bit.no5)
// define CLOCK as microcontroller pin P1.4
#define CLOCK         (P1_bit.no4)
// define LATCH as microcontroller pin P1.3
#define LATCH         (P1_bit.no3)

/*********************************************************************
**********
************************** Global Functions
************************************
**********************************************************************
*********/
void send_data_to_shift_register(uint16_t Value);
void Store_Data(uint16_t Buffer);

#endif /* DRIVER_SHIFT_REGISTER_H_ */
```

4.4. Main

Functions implemented in other files will be called here

- **Delay ():**

// makes a delay during program execution of approximately 1s.

In this project an internal oscillation frequency of 4MHz was used. The ***Delay()*** function was implemented only as a demonstration; this time may suffer variation in other microcontrollers architecture. If you want to use a more precise time, it is recommended to use the timer interrupt of your microcontroller. You can use the 1ms interrupt and insert a counter up to 1000, so you have the value of 1s much more accurate.

```
                              /*main.c*/
/**********************************************************************
***********
Global variables and functions
**********************************************************************
**********/
void delay(uint32_t);
/**********************************************************************
***********
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
**********************************************************************
**********/
void main(void)
{
    while (1U)
    {
           // display numbers from 0 to 9
           Display_Numbers();
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
           // display letters A, B, C, D and colon
           Display_Alphanumeric(LETR_A, LETR_B, LETR_C , LETR_D ,
COLON);
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
           // display letters E, F, G, H and colon
           Display_Alphanumeric(LETR_E, LETR_F, LETR_G , LETR_H ,
COLON);
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
           // display letters I, J, K, L
           Display_Alphanumeric(LETR_I, LETR_J, LETR_K, LETR_L, SEG_K3
& SEG_L3);
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
           // display letters M, N, O, P
           Display_Alphanumeric(LETR_M, LETR_N, LETR_O, LETR_P, SEG_K1
& SEG_L2);
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
           // display letters Q, R, S, T
           Display_Alphanumeric(LETR_Q, LETR_R, LETR_S, LETR_T,
SEG_L1& SEG_L2);
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
           //display letters U, V, X, W
           Display_Alphanumeric(LETR_U, LETR_V, LETR_X, LETR_W, SEG_K2
& SEG_K3 & SEG_L3 & SEG_L4);
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
           // display letters Y, Z, digit 3 OFF, digit 4 OFF
           Display_Alphanumeric(LETR_Y, LETR_Z, D_OFF, D_OFF, SEG_K1 &
SEG_K2);
           // delay close to 1s using MCU frequency of 4MHz
           delay(0xFFFF);
    }
}
```

```
/*********************************************************************
***********
* Function Name: delay
* Description: This function runs delay. MCU frequency used = 4 MHz
* Arguments    : None
* Return Value: None
**********************************************************************
**********/
void delay(uint32_t number) {
           for (; number > 0; number--) {
           // no operation
                      NOP();
           }
}
```

4.5. Timer

This file defines a 1ms timer interrupt. Inside interrupt we call ***LedsMultiplexer()*** function. The longer the interruption time, the lower the display brightness.

/*timer.c*/

```
/*********************************************************************
***********
* File Name    : timer.c
* Version      : for RL78/G13
* Device(s)    : R5F100FE
* Tool-Chain   : GCCRL78
* Description  : This file implements device driver for TAU module.
**********************************************************************
**********/
/*********************************************************************
***********
Includes
**********************************************************************
**********/
#include "mcu.h" // microcontroller generated file
#include "display_handler.h"
/*********************************************************************
***********
Global variables and functions
**********************************************************************
**********/
/*********************************************************************
***********
* Function Name: mcu_1ms_interrupt
* Description  : This function is INTTM00 interrupt service
routine.1ms
* Arguments    : None
* Return Value : None
**********************************************************************
**********/
void mcu_1ms_interrupt(void)
{
// function implemented on display_handler
           LedsMultiplexer();
}
```

5. CHARACTERS CONFIGURATION

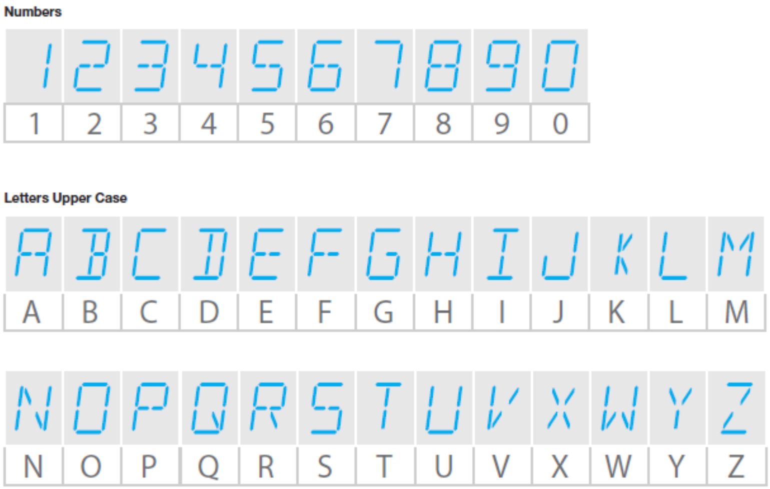
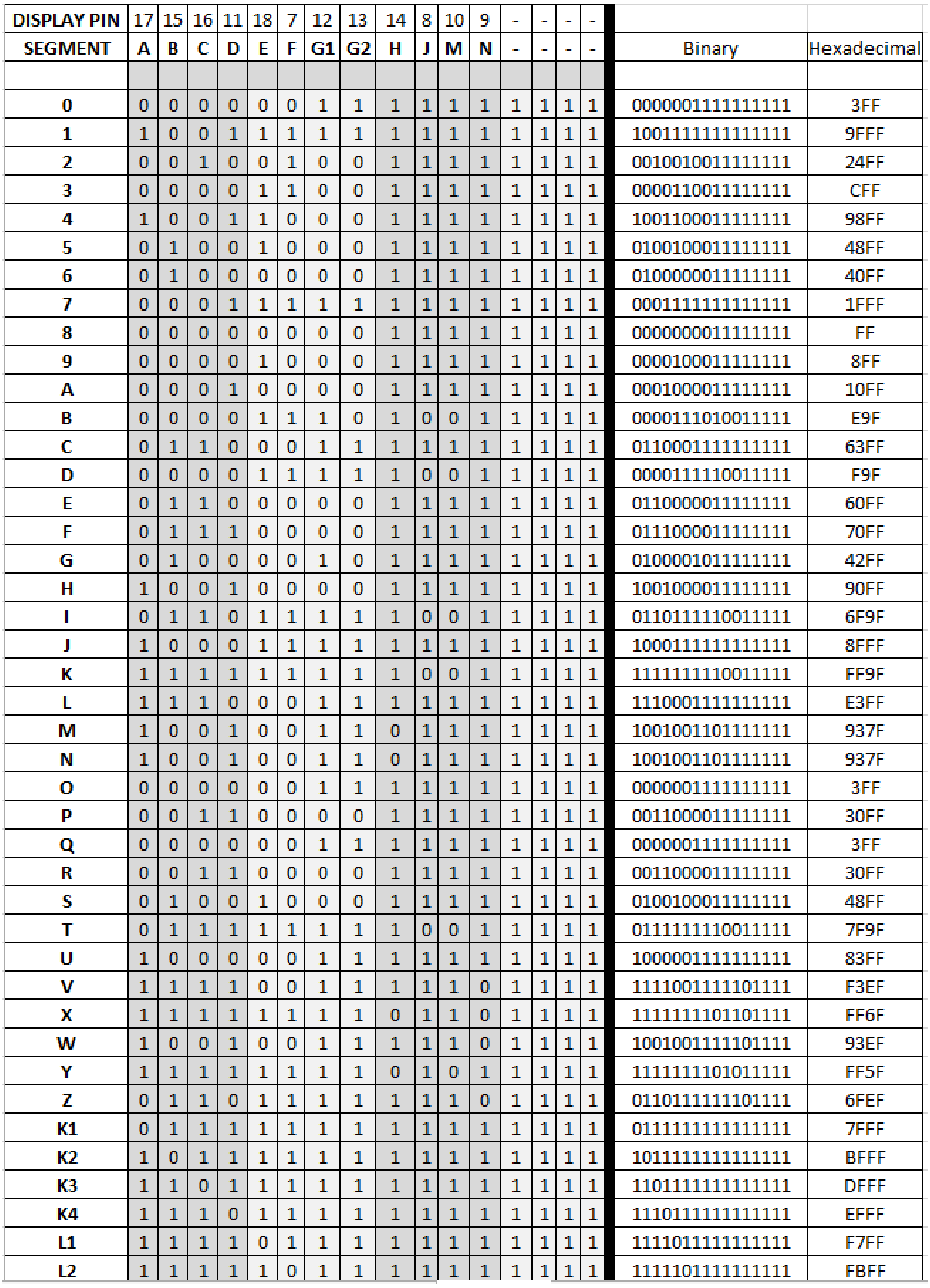


Figure 10. Display segments configuration.

5.1. REFERENCE TABLE

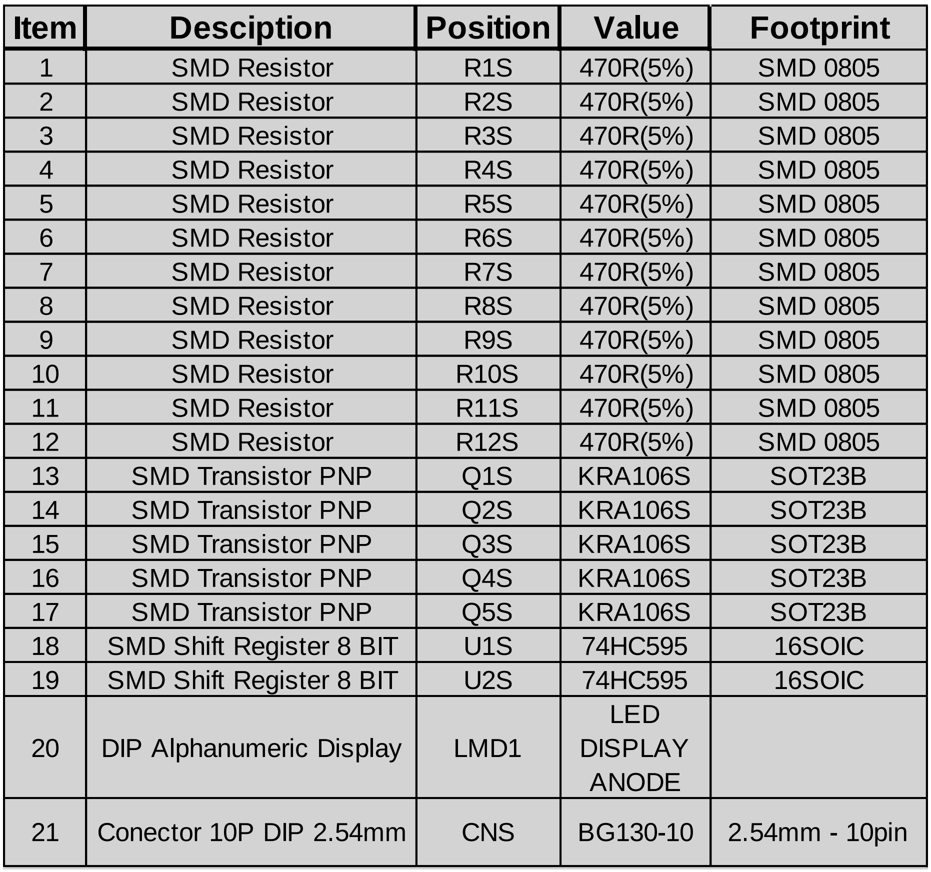
Below table can be used as reference. Shows characters conversion in binary and hexadecimal format.



DISPLAY PIN	17	15	16	11	18	7	12	13	14	8	10	9	-	-	-	-		
SEGMENT	A	B	C	D	E	F	G1	G2	H	J	M	N	-	-	-	-	Binary	Hexadecimal
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0000001111111111	3FF
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1001111111111111	9FFF
2	0	0	1	0	0	1	0	0	1	1	1	1	1	1	1	1	0010010011111111	24FF
3	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	0000110011111111	CFF
4	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1001100011111111	98FF
5	0	1	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0100100011111111	48FF
6	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0100000011111111	40FF
7	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0001111111111111	1FFF
8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0000000011111111	FF
9	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0000100011111111	8FF
A	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	0001000011111111	10FF
B	0	0	0	0	1	1	1	0	1	0	0	1	1	1	1	1	0000111010011111	E9F
C	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0110001111111111	63FF
D	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0000111110011111	F9F
E	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	0110000011111111	60FF
F	0	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0111000011111111	70FF
G	0	1	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0100001011111111	42FF
H	1	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1001000011111111	90FF
I	0	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1	0110111110011111	6F9F
J	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1000111111111111	8FFF
K	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1111111110011111	FF9F
L	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1110001111111111	E3FF
M	1	0	0	1	0	0	1	1	0	1	1	1	1	1	1	1	1001001101111111	937F
N	1	0	0	1	0	0	1	1	0	1	1	1	1	1	1	1	1001001101111111	937F
O	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0000001111111111	3FF
P	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0011000011111111	30FF
Q	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0000001111111111	3FF
R	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0011000011111111	30FF
S	0	1	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0100100011111111	48FF
T	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0111111110011111	7F9F
U	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1000001111111111	83FF
V	1	1	1	1	0	0	1	1	1	1	1	0	1	1	1	1	1111001111101111	F3EF
X	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1111111101101111	FF6F
W	1	0	0	1	0	0	1	1	1	1	1	0	1	1	1	1	1001001111101111	93EF
Y	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1111111101011111	FF5F
Z	0	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0110111111101111	6FEF
K1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0111111111111111	7FFF
K2	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1011111111111111	BFFF
K3	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1101111111111111	DFFF
K4	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1110111111111111	EFFF
L1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1111011111111111	F7FF
L2	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1111101111111111	FBFF

Tabela 1. Conversion table.

6. BILL OF MATERIALS



Item	Desciption	Position	Value	Footprint
1	SMD Resistor	R1S	470R(5%)	SMD 0805
2	SMD Resistor	R2S	470R(5%)	SMD 0805
3	SMD Resistor	R3S	470R(5%)	SMD 0805
4	SMD Resistor	R4S	470R(5%)	SMD 0805
5	SMD Resistor	R5S	470R(5%)	SMD 0805
6	SMD Resistor	R6S	470R(5%)	SMD 0805
7	SMD Resistor	R7S	470R(5%)	SMD 0805
8	SMD Resistor	R8S	470R(5%)	SMD 0805
9	SMD Resistor	R9S	470R(5%)	SMD 0805
10	SMD Resistor	R10S	470R(5%)	SMD 0805
11	SMD Resistor	R11S	470R(5%)	SMD 0805
12	SMD Resistor	R12S	470R(5%)	SMD 0805
13	SMD Transistor PNP	Q1S	KRA106S	SOT23B
14	SMD Transistor PNP	Q2S	KRA106S	SOT23B
15	SMD Transistor PNP	Q3S	KRA106S	SOT23B
16	SMD Transistor PNP	Q4S	KRA106S	SOT23B
17	SMD Transistor PNP	Q5S	KRA106S	SOT23B
18	SMD Shift Register 8 BIT	U1S	74HC595	16SOIC
19	SMD Shift Register 8 BIT	U2S	74HC595	16SOIC
20	DIP Alphanumeric Display	LMD1	LED DISPLAY ANODE	
21	Conector 10P DIP 2.54mm	CNS	BG130-10	2.54mm - 10pin

Figure 11. Components used.

7. REFERENCE

Datasheet component 74HC595
https://assets.nexperia.com/documents/data-sheet/74HC_HCT595.pdf