To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

M16C/Tiny Series

Clock-Synchronous Serial I/O Mode Applied on LED Matrix Drive

1. Abstract

The application note describes an application to display characters on LED matrix. The application employed UART0 (Clock-Synchronous Serial I/O Mode) and 2 I/O ports to control LED driver chips. The MCU used for current application belongs to M16C/29 group.

By running the sample program, a string of Chinese characters will be displayed on 2 continuous LED matrixes in different styles (static display, shift left, shift right, shift up, shift down, static display with color inverted, etc.). The aim of the application is to bring forward a design of LED display which occupies less CPU resources and less peripheral I/O.

Please just take the hardware structure and parameters of the following description as a reference, and confirm or make modifications according to actual conditions in experiment or evaluation.

2. Introduction

The application example described in this document is applied to the following:

• MCU: M16C/26A, M16C/28, M16C/29 Group

This program can be used with other microcomputers within the M16C family, which have the same SFR (special function register) as the M16C/26A, M16C/28, and M16C/29 microcomputers. Please check the manual for any additions and modifications to functions. Careful evaluation is recommended before using this application note.

3. Specification

3.1 Clock-Synchronous Serial I/O Mode of UARTi

All MCU types mentioned above in introduction contain UARTi (i = 0~2) channels, which are serial I/O channels. Each UART channel is independent and is able to work in several modes. In current application, the clock-synchronous serial I/O mode of UART0 is employed. As the port structure and setting of each UART port varies from each other, the settings of UART1~UART2 port for this application need careful examination (please refer to Hardware Manual).

The clock-synchronous serial I/O mode uses a transfer clock to transmit and receive data. In current application, only simplex communication is required; therefore, the occupied pins are transfer clock pin of UART0 (CLK0) and data transfer pin of UART0 (TxD0).

Please refer to Table 3.1 for the chosen functions of UART0.

Table 3.1 Chosen functions of UART0

Item	Setup		Item	Setup	
Transfer clock source	o	Internal clock ($f_1/f_2/f_8/f_{32}$)	Transfer format	o	LSB first
		External clock			MSB first
CLK polarity	o	Output transmission data at the falling edge of the transfer clock	Transmission interrupt factor	o	Transmission buffer empty
					Transmission complete
		Output transmission data at the rising edge of the transfer clock	Output transfer clock to multiple pins	o	Not selected
					Selected

3.2 LED Matrix

The development of LED technology leads to more and more universal applications of LED matrix. For example, unicolor LED matrix is widely employed to display traffic signals, information board etc, and multi-color LED matrix is used for LED screen. A sample structure of a unicolor 4×4 LED matrix is shown in figure 3.1.

Characters can be coded and displayed in dot matrix. Coded characters with the same resolution and font compose a library. One of the most common libraries for Chinese characters is 16×16 SimSun library, which is the one used in current application.

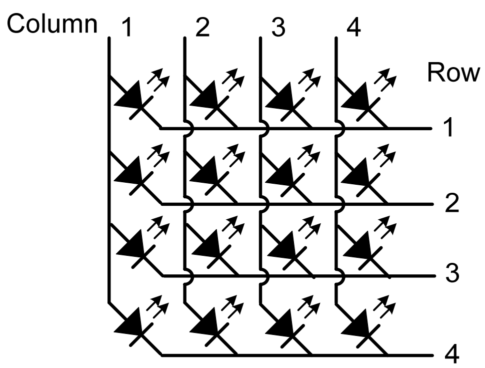


Figure 3.1 A Sample Structure of a Unicolor 4×4 LED Matrix

Normally, a cost-efficient method to display on LED matrix is dynamic display. Therefore the refresh frequency is to be considered. For human eyes, the frequency at which a flickering light is indistinguishable from a steady, non-flickering light is defined as Critical Flicker Frequency (CFF), which is about 48Hz.

In order to get a better effect, the refresh frequency (also called Vertical Scanning Frequency in TV) is set to 76.29Hz (16-division of Line-Scanning frequency); the Line-Scanning frequency (also called Horizontal Scanning Frequency in TV) is set to 1.22 kHz (2^{11}-division of main clock which is 8-division of 20MHz).

3.3 Renesas 16-Bit LED Driver

M66310P/FP is a LED array driver having a 16bit serial-input and parallel output shift-register function with direct coupled reset input and output latch function.

This product guarantees the output electric current of 24mA which is sufficient for cathode common LED drive, capable of flowing 16bits continuously at the same time. Parallel output is open drain output and serial output is available for expansion.

The structure diagram of M6631P/FB is shown in figure 3.2.

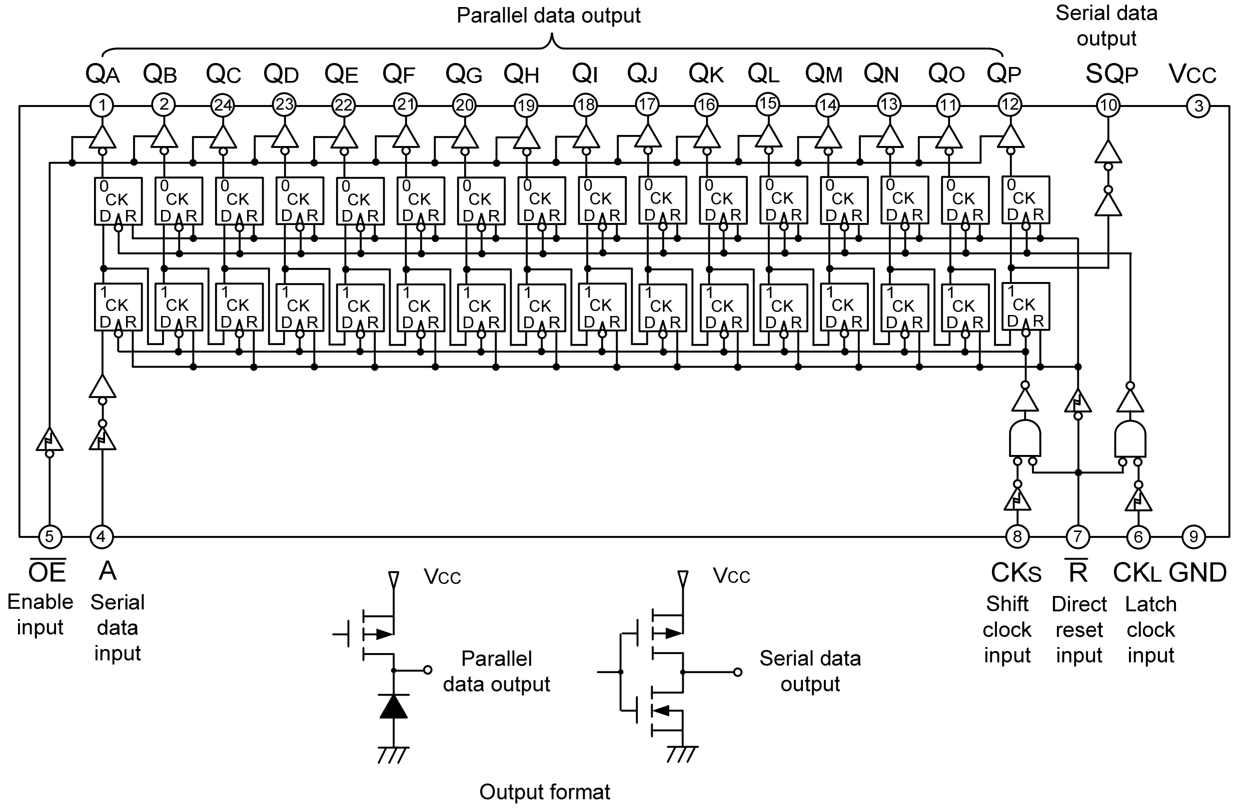


Figure 3.2 M66310P/FP Structure Diagram

4. Design Description

4.1 Hardware structure

In current application, the structure of hardware system is shown in figure 4.1.

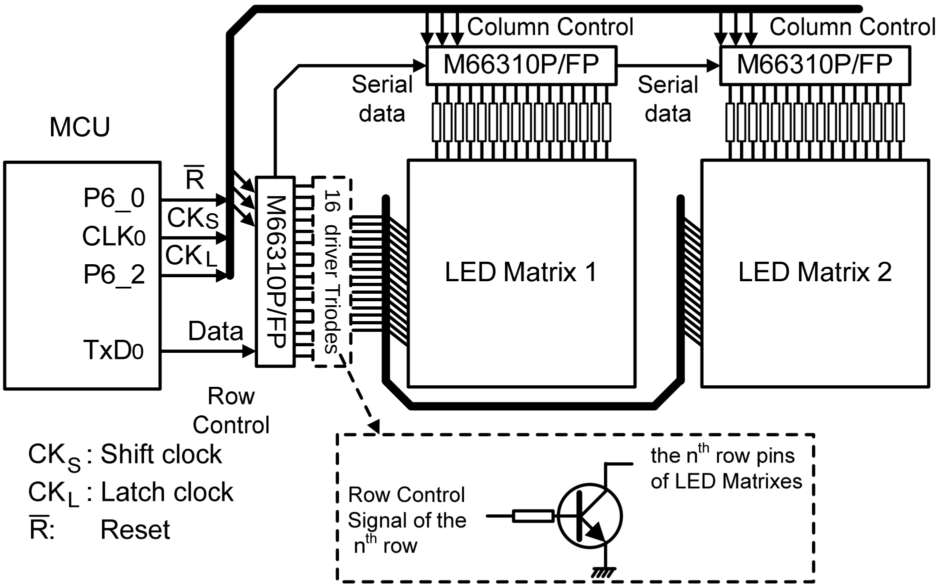


Figure 4.1 Hardware Structure of Application

As shown in figure 4.1, (n+1) chips of M66310P/FP are used (in current application, n = 2), in which n chips are for the column information of corresponding LED matrix and the other 1 chip is for the row control of all LED matrix (additional driver circuit for row control are required according to the actual current consumption). Therefore, at a certain time, respective information is displayed at the same line of respective matrix.

Based on the above hardware structure, the serial data should be transferred in following order:

a) Column signal of the n^{th} LED matrix (information to be displayed on certain line of the n^{th} LED matrix)

b) Column signal of the $(n-1)^{th}$ LED matrix (information to be displayed on certain line of the $(n-1)^{th}$ LED matrix)

c) ……

d) Column signal of the 1^{st} LED matrix (information to be displayed on certain line of the 1^{st} LED matrix)

e) Row selection signal of all LED matrixes

On the end of above sequence, if a latch clock is provided after all data are transferred, all the fixed data will be outputted to the parallel output pins of LED driver chips, and then a certain line of all LED matrixes is lighted. By repeating above sequence with selecting row in turn, a dynamic display will be generated. If a proper refresh frequency is set, the information displayed on LED matrix looks static.

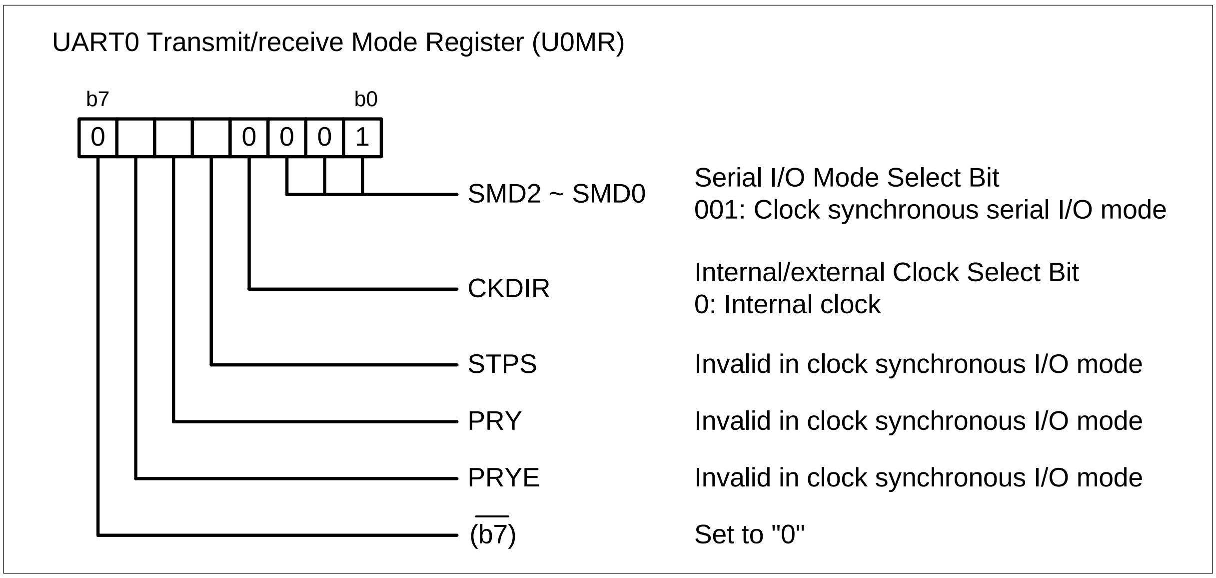
The structure shown in figure 4.1 has a good expansibility. In the condition of current application, which are 1Mbit/s serial communication bit rate and 1.22 KHz Line-Scanning frequency, up to a maximum of 50 LED matrixes in series are feasible. More LED matrixes means more CPU resources occupancy; in the acceptance range of actual conditions, by increasing bit rate or decreasing Line-Scanning frequency, a system will support more LED matrixes.

In addition, the bottle-neck of above hardware structure is the row control driver circuit. With an increasing numbers of LED matrixes, the parameters of corresponding row-driver circuit must be checked carefully.

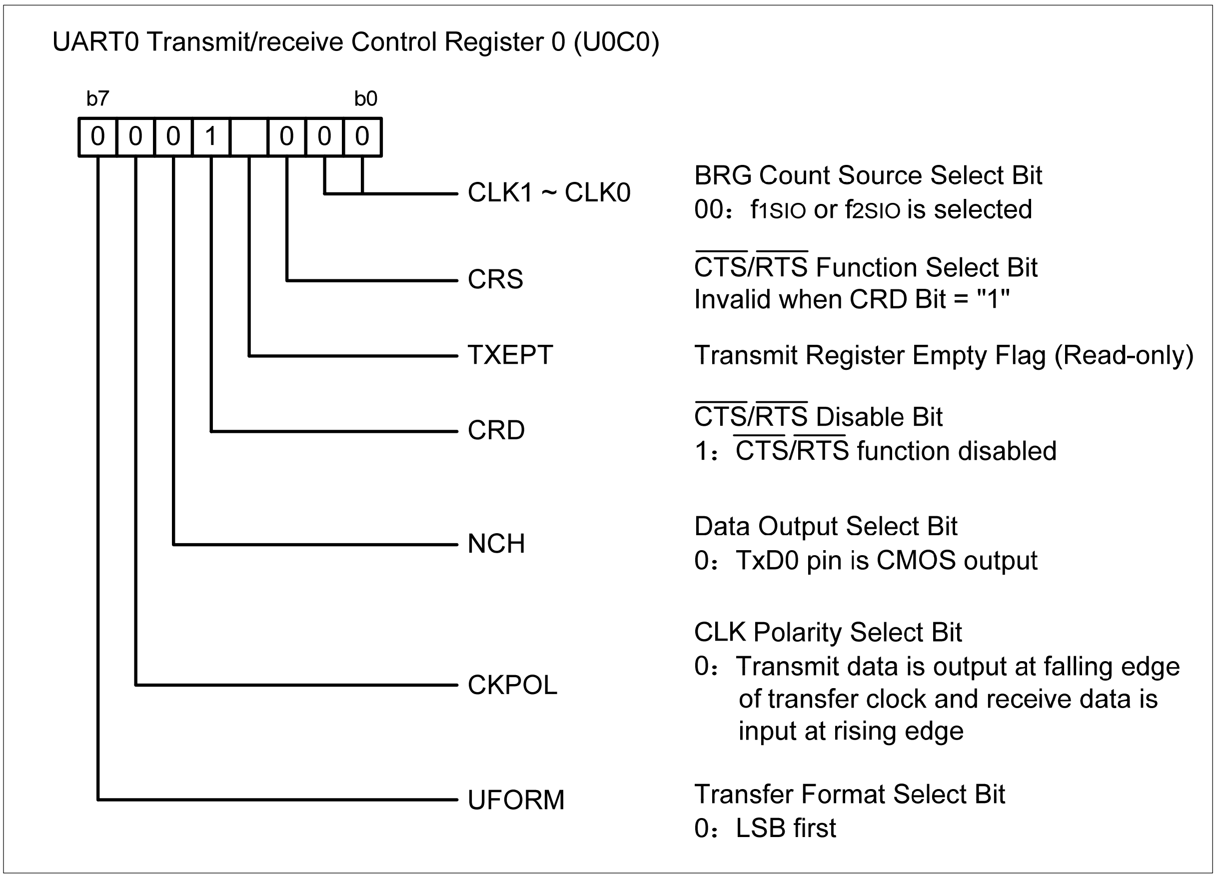
4.2 Register Setting

The following procedure in this application note is based on M16C/29 group products, M16C/28 and M16C/26A group's setup procedure please refer to the hardware user's manual.

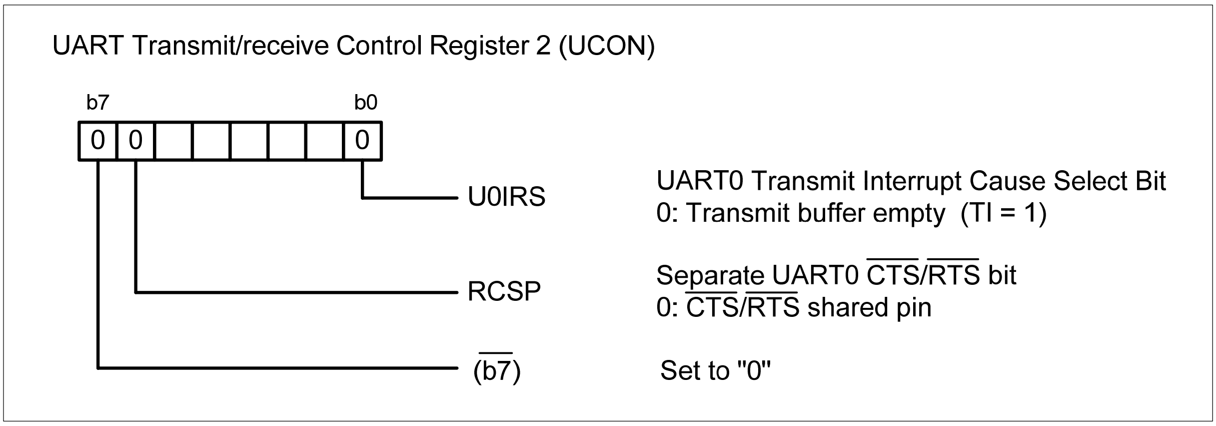
(1) Set UART0 Transmit/receive Mode Register



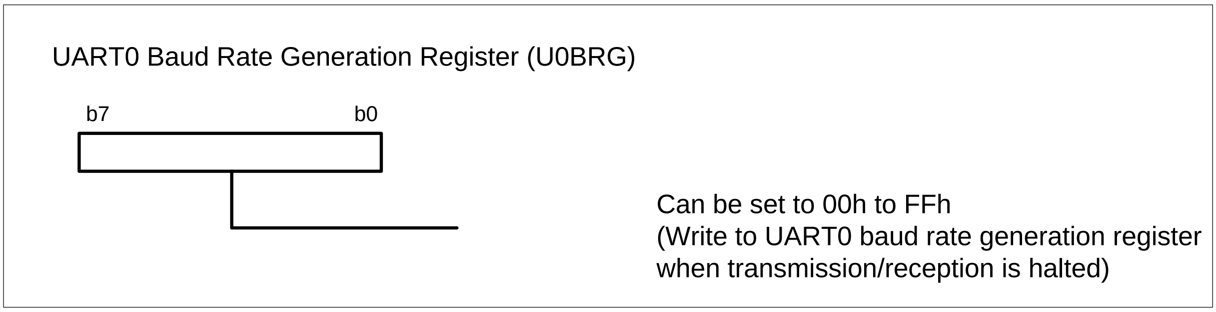
(2) Set UART0 Transmit/receive Control Register 0



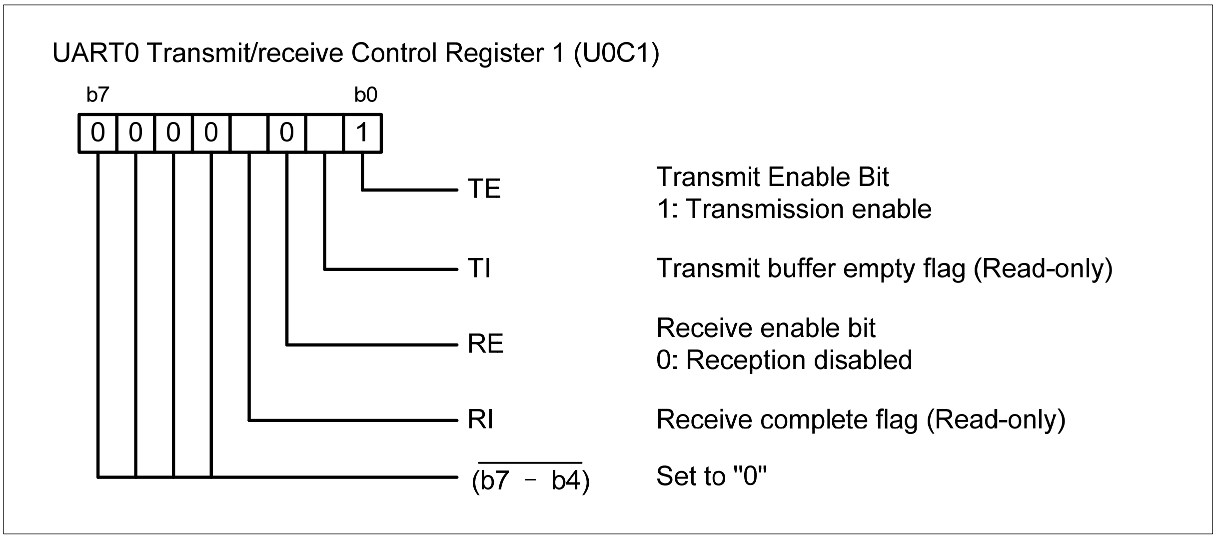
(3) Set UART Transmit/receive Control Register 2



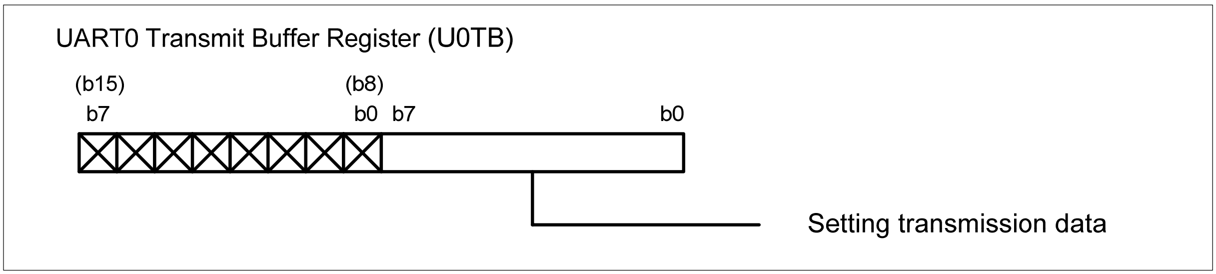
(4) Set UART0 Baud Rate Generation Register



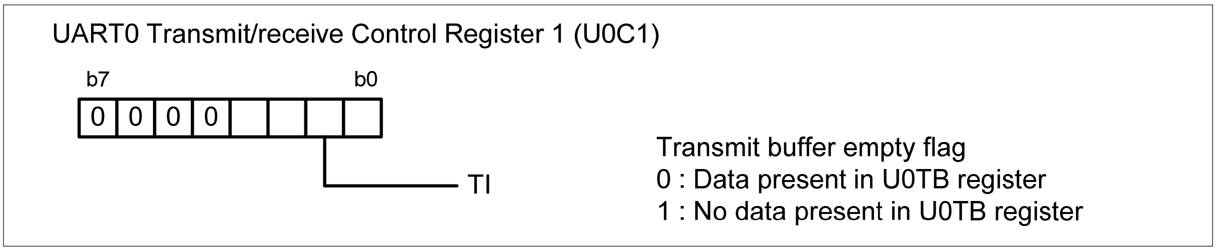
(5) Set UART0 Transmit/receive Control Register 1 to enable transmit



(6) Set UART0 Transmit Buffer Register to write transmit data



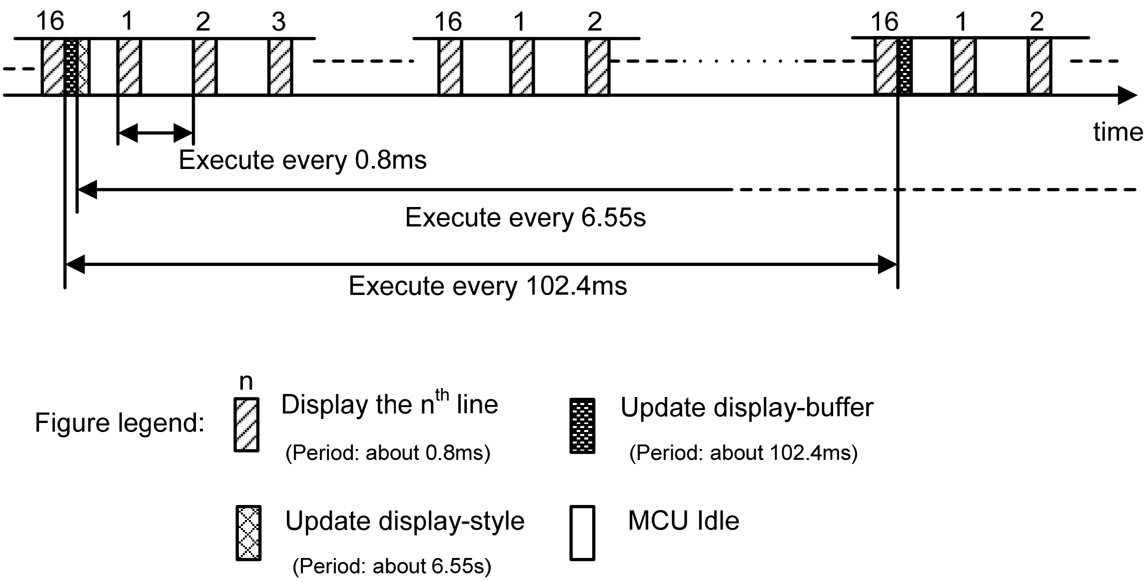
(7) Check UART0 Transmit/receive Control Register 1 to get the status of UART0 Transmit Buffer Register



4.3 Time-Sequence and Structure of Program

In current application, the basic function is to display information on LED matrix and switch display styles periodically. In program, a display-buffer is generated according to the hardware structure of LED matrix. Every 0.8ms, MCU transfers data of a line of display-buffer as well as row control signal to LED drivers, and then generate latch clock; every 102.4ms, MCU refreshes display-buffer according to current display style; every 6.55s, MCU changes display style. Please refer to figure 4.2 for a segment of MCU operation sequence.

The display style changes in turn as follows: Static Display → Shifting Left → Shifting Right → Shifting Up → Shifting Down → Static Display with color inverted → Static Display ……



Note: the proportion of time-spending varies in different designs and settings

Figure 4.2 A Segment of MCU Operation Sequence

The sequence diagram shown in figure 4.3 is a segment of signal relationship. The time segment is from the beginning of a Row-Scanning period to the end of transferring the 2nd 8bit data.

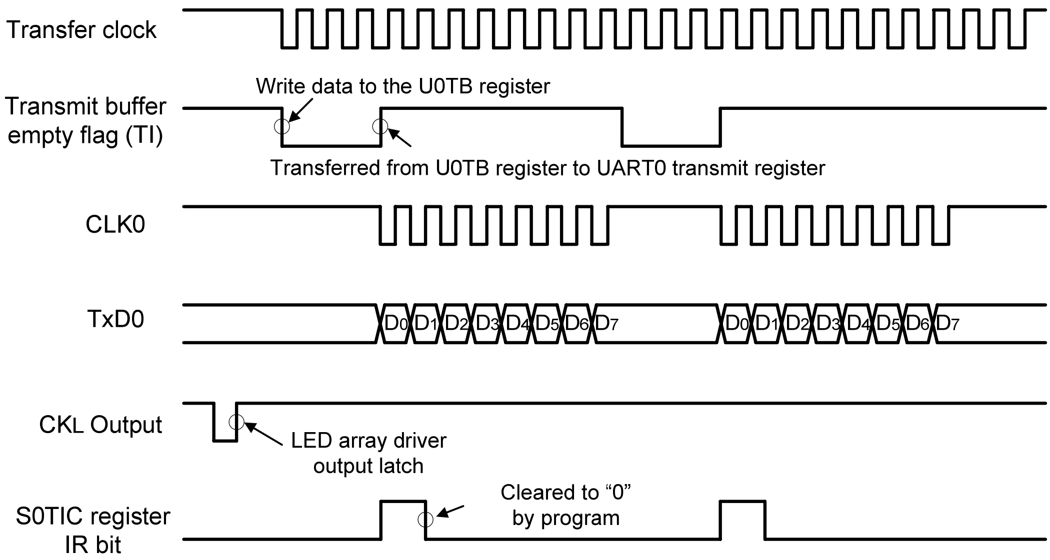
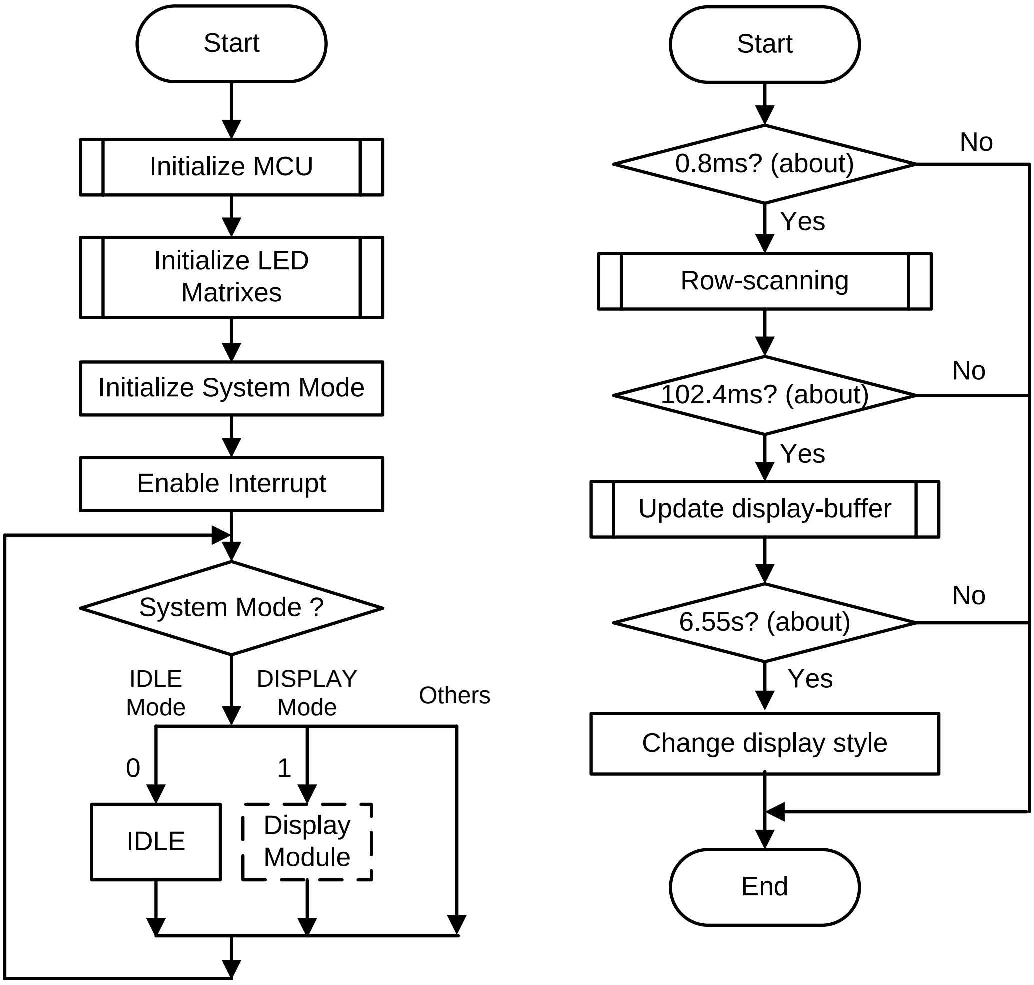


Figure 4.3 A Segment of Transmitting Time-sequence

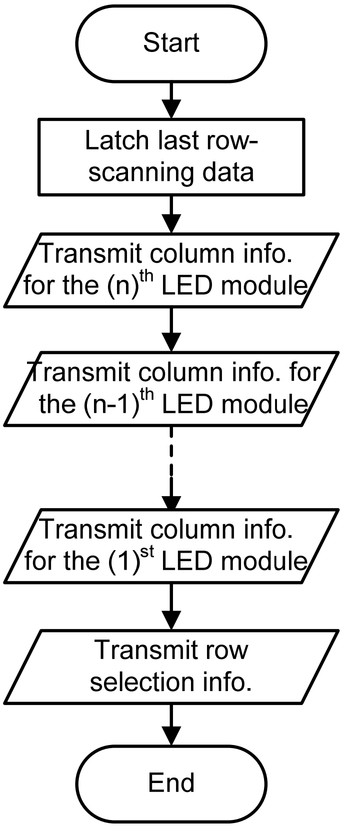
Figure 4.4 shows the flow chart of main program and display module. Figure 4.5 shows the flow chart of row-scanning sub function and updating display-buffer sub function.



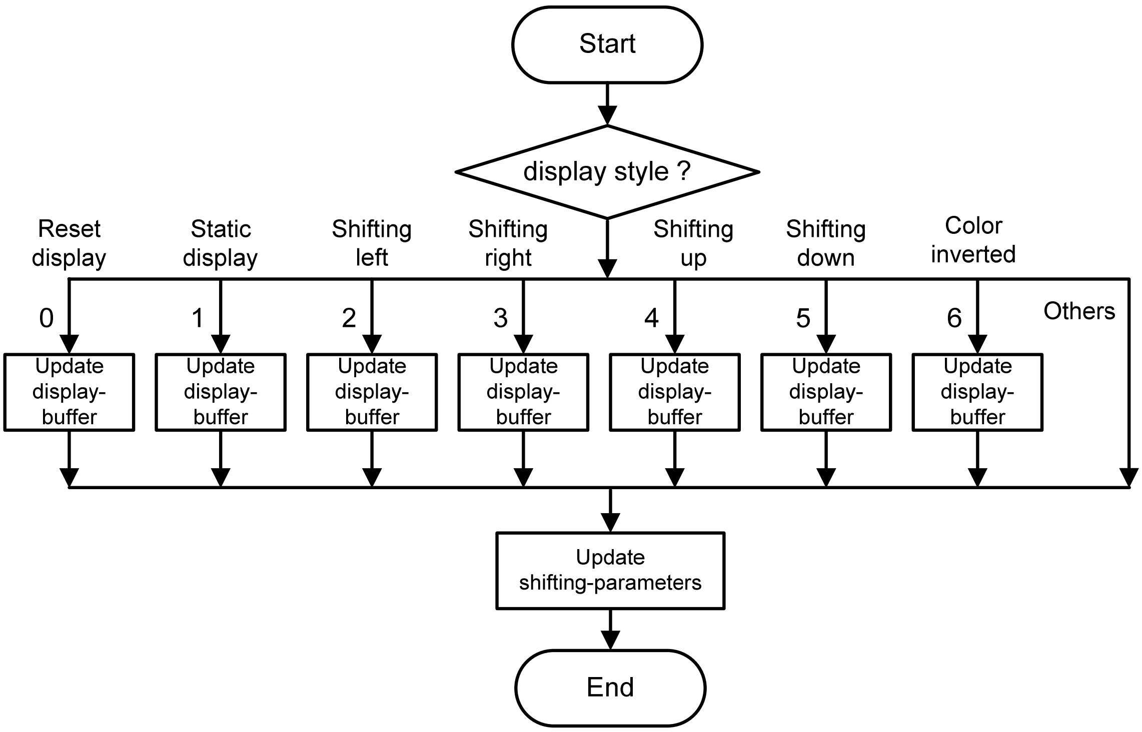
(a) Flow Chart of Main Program

(b) Flow Chart of Display Module

Figure 4.4 Flow Chart of Main Program and Display Module



(a) Flow Chart of Row-Scanning Sub Function



(b) Flow Chart of Updating Display-buffer Sub Function

Figure 4.5 Flow Chart of Row-Scanning Sub Function and Updating Display-buffer Sub Function

5. Sample Program

```
/************************************************************************/
/*    Project         : M16C/Tiny Application Note                     */
/*                    : UARTi (Clock-Synchronous Serial I/O Mode)*/
/*                      Application on LED Matrix Driver               */
/*    MCU             : M30290FCTHP                                     */
/*    C Compiler      : NC30WA, version 5.40                            */
/*    File name       : ClockSS.c                                       */
/*    Function        : Display characters in different styles          */
/*    Code Version    : 1.0                                             */
/*                                                                      */
/*    Copyright (C) 2007 Renesas Technology Corp.                       */
/*    All right reserved.                                               */
/************************************************************************/

/************************************************************************/
/*    Header Including                                                  */
/************************************************************************/
   #include "sfr29.h"      /* SFR definition */
   #include "fontlib.h"    /* Font library    */

/************************************************************************/
/*    Macro Definition                                                  */
/************************************************************************/
   unsigned char system_mode;           /* System mode  */
   #define MODE_IDLE        0x00         /* Idle mode    */
   #define MODE_DISPLAY     0x01         /* Display mode */

   #define  BLOCKS          2          /* LED Matrix modules (Hardware)*/
   #define LINES            16         /* Columns (rows) of LED Matrix */
   #define CHARACTERS       6          /* Characters to be displayed  */

   #define BUFREFRESH_COUNTER 128 /* Time interval to update display-buffer */
                                  /*  = n * (time interval of Row-Scanning)  */

   #define MODECHANGE_COUNTER 64 /* Time interval to update display-style   */
                                 /*  = n * (BUFREFRESH_COUNTER)              */

/************************************************************************/
/*    Function Declaration                                              */
/************************************************************************/
   void init_mcu(void);
   void ini_ledmatrix(unsigned int near * buf);
```

```
   void refresh_buffer(unsigned int near * buf, unsigned char set);
   void output_blocks(unsigned int near * buf);
   void output_word(unsigned int word_data);
   void output_byte(unsigned char byte_data);

/*****************************************************************/
/*    Global Variables Definition                               */
/*****************************************************************/
   unsigned int g_RowScan = 0;       /* Row-Scanning counter      */
   unsigned int g_BufRef = 0;        /* Buffer-refreshing counter */

/*****************************************************************
      Function:       main
      Description:    main function
      Calls:          init_mcu(void)
                      ini_ledmatrix(unsigned int near *)
                      output_blocks(unsigned int near *)
                      refresh_buffer(unsigned int near *,unsigned char)
      Input:          None
      Output:         None
      Return:         None
      Others:         None
*****************************************************************/
void main(void){

   unsigned int buffer[BLOCKS][LINES] = {}; /* Define Display Buffer  Variable */
   unsigned char display_style = 1;         /* Define Display style Veriable */

   init_mcu();                          /* Initialize MCU */

   ini_ledmatrix((unsigned int near *)buffer);  /* Initialize LED Matrix */

   system_mode = MODE_IDLE;                 /* Initialize System Mode */
                                            /* (Set as Idle Mode)     */
   ta0s = 1;                                /* Start Timer A0         */
   asm("fset I");                           /* Enable Interrupt       */

   while(1){

      /* Chose working mode for the system */
      switch(system_mode){

         /* Idle mode */
```

```
        case MODE_IDLE:

              break;

        /* Display mode */
        case MODE_DISPLAY:

              output_blocks((unsigned int near *)buffer);    /* Row-Scanning */

              /* Check whether the condition to update display-buffer is met */
              if(g_RowScan >= BUFREFRESH_COUNTER) {

                 g_RowScan = 0;
                 g_BufRef++;

                 /* Update display-buffer */
                 refresh_buffer((unsigned int near *)buffer, display_style);
              }

              /* Check whether the condition to update display-style is met */
              if(g_BufRef >= MODECHANGE_COUNTER) {

                 g_BufRef = 0;

                 /* Reset display (clear the screen) */
                 refresh_buffer((unsigned int near *)buffer, 0);

                 /* Update display-style, and make sure it is valid */
                 display_style++;
                 if(display_style > 6) {
                    display_style = 1;
                 }
              }

              system_mode = MODE_IDLE;     /* Set system mode to Idle mode */
              break;

        default:
              break;
     }
  }
}
```

```
/*****************************************************************
      Function:      init_mcu
      Description:   initialize MCU
      Calls:         None
      Input:         None
      Output:        None
      Return:        None
      Others:        None
*****************************************************************/
void init_mcu(void)
{
   /* Setting system clock related registers */
   prcr = 0x01;                  /* cm0,cm1,cm2 writing enable         */
   cm2  = 0x00;                  /* system register2 Initialize        */
   cm1  = 0x20;                  /* system register1 Xcin-Xcout:High   */
   cm0  = 0x08;                  /* system register0 Xcin-Xcout:High   */
   prcr = 0x00;                  /* cm0,cm1,cm2 writing disable        */

   prcr = 0x04;                  /* pacr writing enable                */
   pacr = 0x03;                  /* 80pin type                         */
   prcr = 0x00;                  /* pacr writing disable               */

   /* Setting Timer A0 related registers */
   ta0mr = 0x40;                 /* Timer A0 mode register, count f8   */
   ta0   = 2048 - 1;             /* Timer A0 1.22 KHz count            */
   ta0ic = 0x01;                 /* Timer A0 interrupt, level 1        */

   /* Setting UART0 related registers    */
   u0mr = 0x01;   /* Setting UART0 transmit/receive mode register
                     Clock synchronous serial I/O mode Internal clock select */
   u0c0 = 0x10;  /* Setting UART0 transmit/receive control register 0
                   ~CTS/~RTS function disabled TxD0 pin is CMOS output
                  Transmission data is output at falling edge of transfer
                  clock and reception data is input at rising edge LSB first   */
   ucon = 0x00;  /* Setting UART transmit/receive control register 2 setting
                    UART0 transmit interrupt cause is selected to
                    "Transmission buffer empty (TI=1)"
                    transfer clock output from CLK0 only                     */
   u0brg = 10-1;  /* Setting UART1 bit rate generator (1MHz @20MHz f1)       */
   u0c1 = 0x01;   /* Setting UART transmit/receive control register 1
                     Transmit enabled                                        */
   s0tic = 0;     /* Clear UART0 transmit interrupt request bit              */
}
```

```
/****************************************************************
      Function:      ini_ledmatrix
      Description:   initialize ledmatrix related HW and SW
      Calls:         None
      Input:         buf:  starting address of display-buffer
      Output:        None
      Return:        None
      Others:        None
****************************************************************/
void ini_ledmatrix(unsigned int near * buf)
{
   unsigned char i;
   unsigned char j;

   /* Fill display-buffer in advance (not necessary)   */
   for(i = 0; i<BLOCKS; i++) {
      for(j = 0; j<LINES; j++) {
         *(buf + i * LINES + j) = font[i][j];
      }
   }

   p6 = 0x01;     /* Set p6_0 pin to 1 in advance                    */
                  /* (connected to RESET pins of LED driver chips)   */
                  /* Set p6_2 pin to 0 in advance                    */
                  /* (connected to Latch Clock pins of LED driver chips) */

   pd6_0 = 1;     /* Set direction of p6_0 pin as Output             */
   pd6_2 = 1;     /* Set direction of p6_2 pin as Output             */

   p6_0 = 0;      /* Generate RESET signal for LED driver chips      */
   p6_0 = 1;      /* (Operation on the falling edge)                 */
}

/****************************************************************
      Function:      refresh_buffer
      Description:   update display-buffer
      Calls:         None
      Input:         buf:  starting address of display-buffer
                     Set:  display style
                           0: Reset display
                           1: Static display
                           2: Shifting left
                           3: Shifting right
```

```
                          4: Shifting up
                          5: Shifting down
                          6: Static display with color inverted
      Output:        None
      Return:        None
      Others:        None
******************************************************************/
void refresh_buffer(unsigned int near * buf, unsigned char set)
{
   static unsigned char character_scale = 0;    /* Parameter  for shifting */
   static unsigned char shift_scale = 0;        /* Parameter  for shifting */
   unsigned char i;
   unsigned char j;

   /* Chose the display style */
   switch(set) {

      /* Reset display */
      case 0:

         /* Reset parameters for shifting */
         character_scale = 0;
         shift_scale = 0;

         /* Clear display-buffer */
         for(i = 0; i<BLOCKS; i++) {
            for(j = 0; j<LINES; j++) {
               *(buf + i * LINES + j) = 0x00;
            }
         }
         break;

      /* Static display */
      case 1:

         for(i = 0; i<BLOCKS; i++) {
            for(j = 0; j<LINES; j++) {
               *(buf + i * LINES + j) = font[i][j];
            }
         }
         break;

      /* Shifting left */
```

```
case 2:

    for(i = 0; i<BLOCKS; i++) {
        for(j = 0; j<LINES; j++) {
            *(buf + i * LINES + j) =
                                  (font[(character_scale+i)%CHARACTERS][j]
                                    <<shift_scale)
                                 |(font[(character_scale+i+1)%CHARACTERS][j]
                                    >>(16-shift_scale));
        }
    }
    break;

/* Shifting right */
case 3:

    for(i = 0; i<BLOCKS; i++) {
        for(j = 0; j<LINES; j++) {
            *(buf + i * LINES + j) =
                            (font[(character_scale+BLOCKS-i-1)%CHARACTERS][j]
                              >>shift_scale)
                           |(font[(character_scale+BLOCKS-i)%CHARACTERS][j]
                              <<(16-shift_scale));
        }
    }
    break;

/* Shifting up */
case 4:

    for(j = 0; j<(LINES-shift_scale); j++) {
        *(buf + 0 * LINES + j) =
                          (font[(character_scale)%CHARACTERS][j+shift_scale]);
    }

    for(j = 0; j<shift_scale; j++) {
        *(buf + 0 * LINES + j + (LINES - shift_scale)) =
                                   (font[(character_scale+1)%CHARACTERS][j]);
    }
    break;

/* Shifting down */
case 5:
```

```
        for(j = 0; j<(shift_scale); j++) {
            *(buf + 1 * LINES + j) =
                        (font[(character_scale+1)%CHARACTERS]
                                                    [j+(LINES - shift_scale)]);
        }

        for(j = 0; j<(LINES-shift_scale); j++) {
            *(buf + 1 * LINES + j + shift_scale) =
                    (font[(character_scale)%CHARACTERS][j]);
        }
        break;

    /* Static display with color inverted */
    case 6:

        for(i = 0; i<BLOCKS; i++) {
            for(j = 0; j<LINES; j++) {
                *(buf + i * LINES + j) = ~font[i][j];
            }
        }
        break;

    default:
        break;
  }

  /* Update parameters for shifting (until the end of current function) */
  shift_scale++;

  if(shift_scale >= LINES) {

    shift_scale = 0;
    character_scale++;

    if(character_scale >= CHARACTERS) {

        character_scale = 0;
    }
  }
}

/******************************************************************
      Function:      output_blocks
```

```
      Description:   output display-buffer to LED matrix driver
                     chips serially
      Calls:         output_word(unsigned int)
      Input:         buf:  starting address of display-buffer
      Output:        None
      Return:        None
      Others:        None
*****************************************************************/
void output_blocks(unsigned int near * buf)
{
   unsigned char line_index;
   unsigned char i;

   /* Using lower 4bits of Row-Scanning counter
      as row-selection synchronization signal */
   line_index = (g_RowScan - 1) & 0x0F;

   p6_2 = 0;      /* Generate LED matrix driver chip latch signal    */
   p6_2 = 1;      /* (operation on rising edge)                    */

   /* Output column signals for each driver chips in tern */
   for(i = BLOCKS; i>0; i--) {
      output_word(*(buf + (i-1) * LINES + line_index));
   }

   /* Output row selection signal */
   output_word((0x0001<<(LINES-1-line_index)));
}

/*****************************************************************
      Function:      output_word
      Description:   output a word data to LED matrix driver chip
      Calls:         output_byte(unsigned char)
      Input:         word_data:   a word data
      Output:        None
      Return:        None
      Others:        None
 *****************************************************************/
void output_word(unsigned int word_data)
{
   output_byte((unsigned char)(word_data));        /* Output lower 8 bits  */
   output_byte((unsigned char)(word_data >> 8));   /* Output higher 8 bits */
}
```

```
/*****************************************************************
      Function:      output_byte
      Description:   output a byte data to LED matrix driver chip
      Calls:         None
      Input:         byte_data:   a byte data
      Output:        None
      Return:        None
      Others:        None
*****************************************************************/
void output_byte(unsigned char byte_data)
{
   u0tb = byte_data;   /* Writing transmit data */

   /* Check & wait the status of UART0 transmit interrupt request flag */
   while (ir_s0tic == 0) {
   }
   ir_s0tic = 0;        /* Clear UART0 transmit interrupt request flag */
}

/*****************************************************************
      Function:      timerA0_int
      Description:   Timer A0 interrupt handling program
      Calls:         None
      Input:         byte_data:   a byte data
      Output:        None
      Return:        None
      Others:        None
*****************************************************************/
#pragma INTERRUPT timerA0_int
void timerA0_int(void){

   system_mode = MODE_DISPLAY;       /* Set system mode to Display mode */
   g_RowScan++;                      /* Increasing Row-Scanning counter */
}
```

```
/*****************************************************************/
/*    Project      : M16C/Tiny Application Note                 */
/*                 : UARTi (Clock-Synchronous Serial I/O Mode)   */
/*                   Application on LED Matrix Driver           */
/*    MCU          : M30290FCTHP                                */
/*    C Compiler   : NC30WA, version 5.40                       */
/*    File name    : fontlib.h                                  */
/*    Function     : Characters font                            */
/*    Code Version: 1.0                                         */
/*                                                              */
/*    Copyright (C) 2007 Renesas Technology Corp.               */
/*    All right reserved.                                       */
/*****************************************************************/

unsigned int font[][16] = {

   /* 0 Chinese Character "RUI" */
   { 0x0020, 0xf924, 0x2124, 0x2124, 0x21fc, 0x2000, 0xfbfe, 0x2040,
   0x23fe, 0x2252, 0x2a52, 0x3252, 0xc252, 0x0252, 0x024a, 0x0204 },

   /* 1 Chinese Character "SA" */
   { 0x0440, 0x0440, 0xfffe, 0x0440, 0x0440, 0x7820, 0x4bfe, 0x5088,
   0x6050, 0x53fe, 0x4a00, 0x6a00, 0x5200, 0x4400, 0x4400, 0x4800 },

   /* 2 Chinese Character "KE" */
   { 0x0608, 0x7888, 0x0848, 0x0848, 0xfe08, 0x1888, 0x1c48, 0x2a48,
   0x280e, 0x4878, 0x8b88, 0x0808, 0x0808, 0x0808, 0x0808, 0x0808 },

   /* 3 Chinese Character "JI" */
   { 0x1020, 0x1020, 0x1020, 0xfdfe, 0x1020, 0x1420, 0x19fc, 0x3108,
   0xd088, 0x1090, 0x1060, 0x1060, 0x1090, 0x110e, 0x5604, 0x2000 },

   /* 4 blank */
   { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

   /* 5 blank */
   { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

};
```

6. Reference

Renesas Technology Corporation Home Page

http://www.renesas.com

Enquiry

http://www.renesas.com/inquiry

csc@renesas.com

Hardware Manual

M16C/29 Group Hardware Manual

M16C/28 Group Hardware Manual

M16C/26A Group (M16C/26A, M16C/26B, M16C/26T) Hardware Manual

M66310P/FP Hardware Manual

(Use the latest version on the home page: http://www.renesas.com)

Technical Updates / Technical News

(Use the latest information on the home page: http://www.renesas.com)

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	2007.04.03	-	First edition issued