Step 1 - Project Name.

Read a key value and display it on Seven-Segment LED Display using R8C 24/25 micon.

Step 2 - Description.

This example will read the value of the pressed key using the keyboard interrupt function of R8C/Tiny device and display the same on a seven-segment LED display. In the LED display, 4 digits of common anode seven-segment LED displays are connected in multiplexed mode and the input data is given in seven segment format using 8 port lines. Timer RA is used for refreshing the LED display at a rate of 1KHZ.

In the main loop, the control will wait for the key press and the value of the pressed key will be displayed in the LED display.

In the Timer RA interrupt routine, the LED displays gets refreshed with a time interval of 1msec.

In the keyboard interrupt routine, the pressed key is identified and stored in a variable.
Step 3 - Schematic Diagram.
Step 4 - Program Flow Chart.

Main Program:

1. **Start**
2. **Initialize CPU.**
   (Selecting proper oscillator, CPU clock etc.)
3. **Initialize Keyboard.**
   (Initializing the port lines used for the keyboard and enable the keyboard interrupt.)
4. **Initialize Seven-Segment Display.**
   (Initializing the port lines used for the display and Initialize Timer RA to generate interrupt with a time interval of 1mSec.)
5. **Wait for a key press.**
6. **Display the key value on seven-segment display.**
Keyboard Interrupt:

![Flowchart](image-url)
Timer RA Interrupt (Display Refreshing):

1. Start
2. Get the lower 2 bits of digit selection variable (MuxData). (Only 4 digits)
3. Switch off all the digits
4. Send the 7-segment data to port 2 and digit selection data for currently selected digit to port 6.
5. Increment the currently selected digit variable (MuxData)
6. Increment millisecond count
7. Return from Interrupt
Step 5 - Program Development using HEW and Sango.

Flow Chart - Generating source code for this example using Sango.

1. Start Sango
2. Select output folder.
3. Select R8C 24/25 Application Model.
4. Add "4x4 Keyboard using keyboard interrupts" module by double clicking the module in the module selection window.
5. Add "4 Digit multiplexed 7-segment display with 7-segment I/P" module by double clicking the module in the module selection window.
6. Click Generate button.
7. Start HEW and create a empty application.
8. Copy the files generated by Sango from it's output folder to the HEW project folder.
9. Add the assembly language program (.a30) files and C program (.c) files to the current project in HEW.
10. Enter your main program in the main file of the project. In this example, add the keyboard read routine and display routine in the main function.
11. Build the project to get programmble code (.mot) file.
12. Program the generated "mot" file into flash area of the micon using "Flash Starter" software.

End
Start Sango. This will display an opening screen as shown below:

Select R8C 24/25 Application model from model selection window. Sango will display the modules available in this model.
Double click the “4 x 4 Matrix Keyboard Using Keyboard Interrupts” module and this module will be added to the select modules window as shown below:
To get more details like routine description, circuit diagram etc. click icon near to the Module Outline heading. This will open a PDF document, which will give you more details.

Similarly select the “4 Digit Multiplexed 7-Segment Display with Seven Segment I/P” module.
Select the output folder, using the command **User Option** from **Option(O)** menu.

After entering the folder, click **button.**
Now click button to generate the code for the selected two modules.

Sango will delete all files in the output folder before generating the code for selected modules after getting confirmation as shown below:

Files generated by Sango are listed below with short description.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compile.bat</td>
<td>This batch file contains the DOS commands for compiling the files generated.</td>
</tr>
<tr>
<td>cmd.dat</td>
<td>Linker command list file used by the linker.</td>
</tr>
<tr>
<td>sango.sng</td>
<td>Sango user information file.</td>
</tr>
<tr>
<td>R8C2425_FE_Appl_main.c</td>
<td>Main C file for the application. User has to include or add their main flow program in this file. This file contains the CPU initialization routines and main routine for the application. For any of the application module this file will be generated.</td>
</tr>
<tr>
<td>R8C2425_FE_Appl_main.h</td>
<td>Header file for the Main C file.</td>
</tr>
<tr>
<td>sfr_r8c2425.h</td>
<td>SFR declaration file for R8C 24/25 micons.</td>
</tr>
<tr>
<td>R8C2425_FE_A05.c</td>
<td>C file for 4 x 4 matrix keyboard using keyboard interrupts. This file contains the programs for initializing keyboard interrupts and reading the pressed key value.</td>
</tr>
<tr>
<td>R8C2425_FE_A05.h</td>
<td>Header file for the matrix keyboard module C file.</td>
</tr>
<tr>
<td>R8C2425_FE_A09.c</td>
<td>This C file contains the initialization routine for 7-segment display, display routine etc.</td>
</tr>
<tr>
<td>R8C2425_FE_A09.h</td>
<td>Header file for the 7-segment display module.</td>
</tr>
<tr>
<td>ncr0.a30</td>
<td>Assembly language file contains stack initialization and data initialization routines.</td>
</tr>
<tr>
<td>sect30.inc</td>
<td>Include file for “ncr0.a30” file. Contains the fixed vector, variable vector definitions.</td>
</tr>
</tbody>
</table>
Start HEW and create a new empty project. HEW will show the empty project as shown below:

Now add the assembly language program file “ncrt0.a30” and C files “R8C2425_FE_Appl_main.c”, “R8C2425_FE_A05.c” and “R8C2425_FE_A09.c” to current HEW project using Add Files command from Project menu.

After adding the files generated by Sango, these files are displayed in the workspace as shown below:
Step by Step Procedure to Start and Complete a Project

Open the file “R8C2425_FE_Appl_main.c”. The function “WaitForKeyPress()” in the keyboard module (R8C2425_FE_A05.c) will wait for a key press and return the value of the pressed key. The function “Display4Digits()” in the 7-segment display module (R8C2425_FE_A09.c) will display a 4 digit number on the 7-segment display.

So, in the main loop you have to wait for a key press and then read the key value. Finally display the key value on the display.

To do the modification, first delete the function “LEDDisplayBCDInputDemo()” in the main function, which will show simple counter (0000 to FFFF) on the display. This function is removed from the file because we want to display our key value on display.
Next add the functions “WaitForKeyPress()” and “Display4Digits()” and a variable “KeyData” in the main function and function declaration “void Display4Digits(unsigned int count);” before the main function as shown below with red rectangle.
Step by Step Procedure to Start and Complete a Project

Build the project using **Build** command in **Build** menu. The generated ".mot" by the HEW can be used to download into the flash area of the micon using Flash Starter software.

**Step 7 - Listing of the Program.**

**Program Listing - Main Loop.**

```c
/***************************************************************************/
/* */
/* System file name : R8C2425_FE_Study_Main.C */
/* */
/* System name : R8C24/25 Study Model */
/* */
/* Copyright(C)2007 Frontline Electronics */
/* */
/* All rights reserved. */
/* */
/* Version : 1.00 */
/* */
/* ROM size : - */
/* */
/* RAM size : - */
/* */
/* Contents : - */
```
Step by Step Procedure to Start and Complete a Project

```c
/*
* Frontline Electronics Pvt Ltd., India.
* www.MightyMicons.com
***************************************************************************/

#include "sfr_r8c2425.h" /* Definition of the R8C/24,25 SFR for Sango */
#include "r8c2425_FE_Appl_main.h" /* Definition of processing for main */

void MCUInitialize(void);
void Display4Digits(unsigned int count);

main()
{
    unsigned char KeyData;
    MCUInitialize();
    InitializeKeyboard(); // ##R8C24/25-FE-A05## 4 X 4 Matrix
    InitializeSevenSegmentLEDDisplay(); // ##R8C24/25-FE-A08## Initialize I/O
    while(1)
    {
        //Place your Code here
        KeyData = WaitForKeyPress(); // Wait and read the pressed key code
        Display4Digits(KeyData); // Display the read key value
    }
}
```
Step by Step Procedure to Start and Complete a Project

/* Select main clock */
void MCUInitialize(void)
{
    prc0 = 1; // Unlock CM0, CM1, OCD
    cm0 = 0x08; // Start main clock, CM16 and CM17 enable
    cm1 = 0x28; // Main clock = no division
    ocd2 = 0; // Change to Main clock operation
    cm14 = 1; // Turn off ring osc
    prc0 = 0; // Lock the System Clock Control Register
}

Program Listing - Keyboard.
**************************************************************************/
/* Scan a 4 X 4 Matrix Keyboard using Key Input Interrupts */
/* */
/* */
/* Key Input Interrupts KI0 to KI3 are used to scan the 4 X 4 Keyboard */
/* The column lines are connected to KI0 to KI3 */
/* */
/* */
/* Output: Variable - KeyboardFlag will be set 1, if any key is pressed */
/* Variable - KeyboardCode will hold the pressed keyvalue */
/* */
/* */
/* Keyboard Connection:- */
/* Row0 -> P14 */
/* Row1 -> P15 */
/* Row2 -> P16 */
/* Row3 -> P17 */
/* Column0 -> P10 (KI0) */
/* Column1 -> P11 (KI1) */
/* Column2 -> P12 (KI2) */
/* Column3 -> P13 (KI3) */
/* */
/* */
/* Frontline Electronics P Ltd, India */
/* www.MightyMicons.com */
**************************************************************************/

#include "sfr_r8c2425.h"

void SetRow(char t);
char ReadColumn(void);
char KeyboardFlag=0;
unsigned char KeyboardCode=0;

/* Returns the keyboard status */
char ReadKeyboardStatus(void)
{
    return(KeyboardFlag);
}

/* Wait for a key press and return the value of the pressed key */
char WaitForKeyPress(void)
{
    while(KeyboardFlag == 0);
    KeyboardFlag = 0;
    return(KeyboardCode);
}

/* Returns the last pressed keycode */
char ReadKeyCode(void)
{
    KeyboardFlag = 0;
    return(KeyboardCode);
}

/* Keyboard Initialization */
void InitializeKeyboard(void)
{
    pd1_7 = 1; // P17(ROW3) line as output
    pd1_6 = 1; // P16(ROW2) line as output
    pd1_5 = 1; // P15(ROW1) line as output
    pd1_4 = 1; // P14(ROW0) line as output
    pd1_0 = 0; // P10(Column0) line as input
    pd1_1 = 0; // P11(Column1) line as input
    pd1_2 = 0; // P12(Column2) line as input
    pd1_3 = 0; // P13(Column3) line as input

    pu02 = 1; // Enable pull up to P10 to P13 lines

    kien = 0x55; // Enable KI0 to KI3 interrupts and
                  // select Falling Edge
asm("FCLR I"); // disable irqs before setting irq
// registers
kupic = 0x02; // Select level 2 for keyboard interrupt
asm("FSET I"); // enable interrupts

SetRow(0x00);

/* Keyboard Interrupt Service routine */
void ProcessKey_Int(void)
{
    unsigned char a,b,c,f;
    SetRow(0x00);
    if((ReadColumn() != 0x0f) && KeyboardFlag == 0)
    {
        //any one key pressed
        KeyboardCode = 0;
        for(a=0;a<4;a++)
        {
            //Row Setting
            f = ~(0x01 << a);
            SetRow(f);
            b = ReadColumn();
            for(c=0;c<4;c++)
            {
                //Column checking
                f = 0x01 << c;
                if((b & f) == 0)
                {
                    //Key code
                    KeyboardFlag=1;
                    SetRow(0x00);
                    return;
                }
                KeyboardCode++;
            }
        }
    }
    SetRow(0x00);
/* Sets correct level to rows */
void SetRow(char t) {
    if(t & 0x01) {
        p1_4 = 1;
    } else {
        p1_4 = 0;
    }
    if(t & 0x02) {
        p1_5 = 1;
    } else {
        p1_5 = 0;
    }
    if(t & 0x04) {
        p1_6 = 1;
    } else {
        p1_6 = 0;
    }
    if(t & 0x08) {
        p1_7 = 1;
    } else {
        p1_7 = 0;
    }
}
Step by Step Procedure to Start and Complete a Project

/* Read column value */
char ReadColumn(void)
{
    char a;
    a = pl & 0x0f;
    return(a);
}

Program Listing - Seven Segment Display.

/****************************************************************************/
/* Seven Segment Display Interface - Multiplexed With 7 Segment I/P */
/* Displays a four digit counter from 0x0000 to 0xffff. */
/****************************************************************************/

#include "sfr_r8c2425.h"

void InitializeTimerRA(void);
void InitializeSevenSegmentLED(void);
void Display4Digits(unsigned int count);
unsigned char DigitData[4];
char MuxData;
unsigned int Count;
int OneSecCount;

// Segment connection and determining data
//
// D7 D6 D5 D4 D3 D2 D1 D0 - Data bit position
// dp g f e d c b a - Display Segment
// off off on on on on on - Segment status for '0'
// 1 1 0 0 0 0 0 0 - 0xc0 -> Actual Data for '0'

const unsigned char SevenSegmentCode[0x10] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                                              0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};
const unsigned char DigitSelectionData[4] = {0x0e,0x0d,0x0b,0x07};

void LEDDisplay7SegmentInputDemo(void)
{
    if(OneSecCount > 999)
    {
        OneSecCount = 0; // Clear delay counter
        Display4Digits(Count); // Display the count value
        Count++; // Increment count value
    }
}

/* Timer RA interrupt service routine */
void ProcessTimer_RA_Int(void)
{
    MuxData &= 0x03; // Get the multiplexer value (bit 0)
    p6 |= 0x0f; // Switch off the segments which are currently on
    p2 = DigitData[MuxData]; // Send the data to be displayed
    p6 &= 0xf0;
    p6 |= DigitSelectionData[MuxData];
    // Enable the digit corresponding to the
    // currently selected digit
    MuxData++; // Increment mux data
    OneSecCount++; // Increment timer count for 1 second delay
}
/* Initialize Timer RA to generate 1mSec Interrupt */
void InitializeTimerRA(void)
{
    tramr = 0x30; // timer mode, clock = f2
    trapre = 100; // div by 100
    tra   = 100; // initial value 100

    asm("FCLR I"); // disable irqs before setting irq registers
    traic = 2; // Set the timer RA’s interrupt priority to 2
    asm("FSET I"); // enable interrupts

    tstart_tracr = 1; // Start timer RA
}

/* Initialize I/O Lines */
void InitializeSevenSegmentLEDDisplay(void)
{
    pd2  = 0xff; // Select Port 2 as output port
    pd6  = 0x0f; // Select port lines P60 to P63 as output lines
    p6   = 0x0f; // Switch off all the digits
    p2   = 0xff; // Data for Blank
    MuxData = 0;
    OneSecCount = 0;
    DigitData[0] = 0xff; // Clear 1st Digit
    DigitData[1] = 0xff; // Clear 2nd Digit
    DigitData[2] = 0xff; // Clear 3rd Digit
    DigitData[3] = 0xff; // Clear 4th Digit
    InitializeTimerRA();
}

/* Display a word */
void Display4Digits(unsigned int count)
{
    DigitData[3] = SevenSegmentCode[(count & 0xf000) >> 12];
        // Get the seven segment code higher nibble and
        // store it digit data array
    DigitData[2] = SevenSegmentCode[(count & 0x0f00) >> 8];
        // Get the seven segment code lower nibble and
        // store it digit data array
    DigitData[1] = SevenSegmentCode[(count & 0xf0) >> 4];
        // Get the seven segment code lower nibble and
        // store it digit data array
        // Store it digit data array
        // Store it digit data array
        // Store it digit data array
}

/* Display a word */
void Display4Digits(unsigned int count)
{
Step by Step Procedure to Start and Complete a Project

```c
// Get the seven segment code higher nibble and
// store it digit data array
DigitData[0] = SevenSegmentCode[count & 0x0f];
// Get the seven segment code lower nibble and
// store it digit data array
```

**Step 9 - Testing.**

Make the connections as listed below:

**Keyboard Connection:-**

- Row1 -> P14
- Row2 -> P15
- Row3 -> P16
- Row4 -> P17
- Column1 -> P10 (KI0)
- Column2 -> P11 (KI1)
- Column3 -> P12 (KI2)
- Column4 -> P13 (KI3)

**LED Display Connection:-**

- 7 Segment I/P:
  - Segment a -> P20
  - Segment b -> P21
  - Segment c -> P22
  - Segment d -> P23
  - Segment e -> P24
Step by Step Procedure to Start and Complete a Project

Segment f -> P25
Segment g -> P26
Segment dp -> P27

Digit Selection:
1st Digit -> P60
2nd Digit -> P61
3rd Digit -> P62
4th Digit -> P63

Note: Now, you can use Topview Simulator to verify this design concept.