Tanmay Sane
Sean Hicks
Advanced Embedded – Lab2

Can Communications Between Boards

Objective: This lab exercise provides an introduction to CAN (Controller Area Network) communication between two Renesas RX63N development boards. It aims to establish communication between two boards using the CAN protocol to send and receive a three-character code. The received code is displayed on the LCD screen of the receiving board.

Requirements:

The requirements were communicated by the course instructor and are listed here for reference:

1. The code generated must be written in C for the RX63N Evaluation Board.
2. The code must be well commented and easy to follow.
3. Programmable bit rate can be any rate (CAN clock source> 8MHz)
4. Show the entered code on the top row of the LCD. Before any switch is pressed, show the character 'A'.
5. Only codes for capital letters 'A' to 'Z' are used.
6. When SW1 of the board is pressed, the displayed character "increments" by one, i.e. 'A' increases to 'B'.
7. When SW2 of the board is pressed, the displayed character "decrements" by one, i.e. 'Z' increases to 'Y'.
8. The incrementing of the character will roll over from 'Z' to 'A'.
9. The decrementing of the character will roll over from 'A' to 'Z'.
10. When SW3 of the board is pressed, the current character is "locked" and the next character is selected using SW1 and SW2.
11. When selecting the third character of the code and SW3 is pressed, the three character code should be sent via the CAN bus to the other board.
12. When the three character code is sent from a board, it should disappear from the top row of the LCD.
13. When the three character code is received from the CAN bus, it should be displayed on the second line of the LCD.
14. The same code should run in both boards (with the exception of addresses).
15. The boards should run without HEW running on a PC.
16. Complete your Lab Report.
17. Bring the new board to the lab TA and demonstrate the new code (without the HEW application running). When the TA checks your board, he will also take your lab report. You will not need to include a printout or soft copy all of the code – just “snippets”.

Pre lab:

Both the RX63N Boards were acquired .The required CAN API demonstration sample codes were imported into the HEW Workspace. The CAN connector was assembled for the lab exercise.

Tanmay Sane
Sean Hicks
Advanced Embedded – Lab2

Background:

The basic concepts regarding the serial asynchronous communication protocol, Normal and FIFO modes of operation, mailbox numbering, and CAN based interrupts are essential for implementation of the lab exercise.

General steps:

1. The CAN API for the RX63N Board was downloaded & reviewed.
2. State Diagram satisfying the lab requirements was developed.
3. The state diagram logic was coded into a C program for Embedded RX63N Board.
4. This code was integrated into the CAN API & complied using HEW.
5. Configuration of the Mail Boxes for transmission & reception
6. Baud rates of both the boards were ensured to be identical
7. Both the boards were downloaded with the API code in Debug mode.
8. Both the boards were connected using the CAN Connector.
9. Debugging was carried out till the required results were carried obtained.
10. The code was further optimized for better performance.

Detailed Steps:

1. The initial configuration of CAN related attributes was done & it was ensured that mailbox ID's for transmission & reception were identical. This id was also passed as parameter to the data frame structure.

```
g_tx_id_default = 0x001;
g_rx_id_default = 0x001;
g_tx_dataframe.id=g_tx_id_default;
```

2. The state diagram shown in Figure 1 satisfying the lab requirements was drawn.

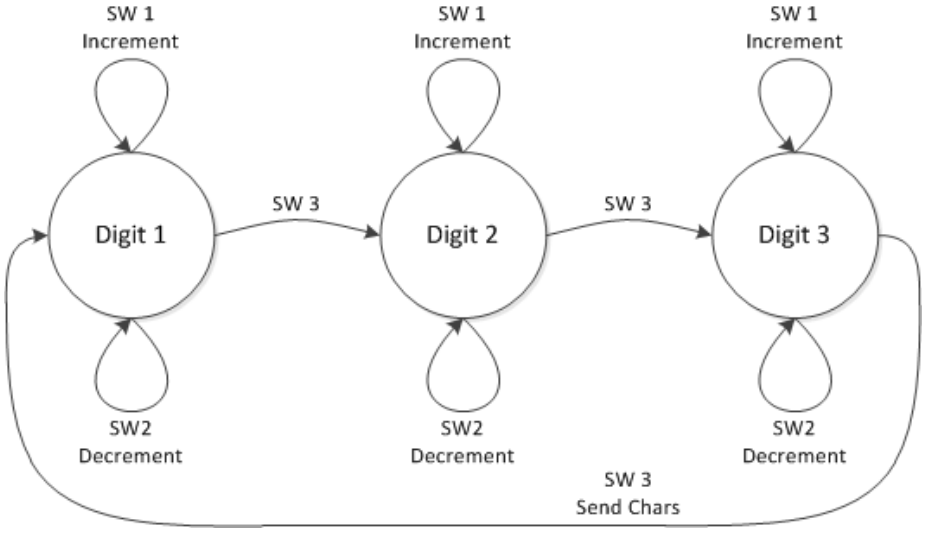

Figure 1: Truth table for CAN communication lab

To satisfy the state diagram, three stages of operation (state1, state2 & state3) were determined, with the current state held in the variable 'state'.. These states denote the character that is currently being edited. Each digit is represented by one variable, 'dig1', 'dig2', or 'dig3'. Code for the evaluation board was developed to meet the lab requirements. The variables dig1, dig2, and dig3 are initialized as follows:

```
char state = 1;
/* state = 1: 1st digit
*  state = 2: 2nd digit
*  state = 3: 3rd digit
*/

/*   initialize each to 'A'     */
char dig1 = 65;
char dig2 = 65;
char dig3 = 65;
uint8_t     IST[14] = {0};
```

3. Display was initialized as follows. Thus, before any switch was pressed 'A__' was shown on the top row.

```
sprintf((char *)IST, "%c%c%c",65,95,95);
lcd_display(LCD_LINE1, IST);
```

4. Pressing switch 1 would generate an increment in the shown character as 'B__','C__' etc. Rollover logic from Z to A was also incorporated. For this requirement the logic for switch 1 was implemented as follows:

```
void sw1_function(void)
{
if ( state == 1 )
{
     if ( ++dig1 > 90 ){ dig1 = 65; } /*  Rollover condition for
     state1 in  SW1*/
     sprintf((char *)IST, "%c%c%c",dig1,95,95);
     lcd_display(LCD_LINE1, IST);
}
if ( state == 2 )
{
     if ( ++dig2 > 90 ){ dig2 = 65; } /*  Rollover condition for
     state2 in  SW1*/
     sprintf((char *)IST, "%c%c%c", dig1, dig2,95 );
     lcd_display(LCD_LINE1, IST);
}
```

```
if ( state == 3 )
{
      if ( ++dig3 > 90 ){ dig3 = 65; } /*  Rollover condition for
      state3 in  SW1*/
      sprintf((char *)IST, "%c%c%c", dig1, dig2, dig3 );
      lcd_display(LCD_LINE1, IST);
}

} /*  End of function sw1_function(). */
```

5. Pressing the switch 2 would indicate a decrement, but for this a rollover condition was needed & hence implemented and the following characters were displayed 'Z__','Y__' etc.

```
void sw2_function(void)
{
if ( state == 1 )
{
        if ( --dig1 < 65 ){ dig1 = 90; }
        /*  Rollover condition for state1 in  SW2*/
         sprintf((char *)IST, "%c%c%c",dig1,95,95);
         lcd_display(LCD_LINE1, IST);
}
if ( state == 2 )
{
          if ( --dig2 < 65 ){ dig2 = 90; }
          /*  Rollover condition for state1 in  SW2*/
           sprintf((char *)IST, "%c%c%c", dig1, dig2,95 );
          lcd_display(LCD_LINE1, IST);
}
     if ( state == 3 )
{
            if ( --dig3 < 65 ){ dig3 = 90; }
            /*  Rollover condition for state1 in  SW2*/
             sprintf((char *)IST, "%c%c%c", dig1, dig2, dig3 );
            lcd_display(LCD_LINE1, IST);
     }
}/*  End of function sw2_function(). */
```

6. As the switch 3 is pressed the state variable is incremented until it is equal to three as we are just sending three bytes .It is re initialized to state 1 once it has exceeded three. The logic for this switch is implemented as follows :

```
void sw3_function(void)
{
if ( state++ == 3 )          //  Time to send
{            /* Tx Data frame is initialized */
. /* Reinitialization of display variables& state */
. /* Reinitialization of display*/
.
}
else
. /*Display  Condition  State 2*/
. /*Display  Condition  State 3*/
}
```

7. Pressing switch 3 would increment the stage to state 2 and would give output as ‘AA_’ and on further events of switch 2 or 1 would either decrement or increment the stage character.

8. Further events from switch 3 would increment the stage to stage 3 and the characters at the third place would respond to the increment & decrement commands from switches 1 or 2 respectively.

9. If the state variable is not three then the program proceeds to display the relevant data as follows:

```
else
{
     if ( state == 2 )     /*Display  Condition  State 2*/
     {
          sprintf((char *)IST, "%c%c%c",dig1,dig2,95);
          lcd_display(LCD_LINE1, IST);
     }
     if ( state == 3 ) /*Display  Condition  State 1*/
     {
          sprintf((char *)IST, "%c%c%c",dig1,dig2,dig3);
          lcd_display(LCD_LINE1, IST);
     }
}
```

10. On the third occurrence of press event from switch 3 the transmission routine for the CAN bus would be initialized, and the transmission would take place. Thus if the value of state is equal to three then the character space variables are feed to the transmission data frame to transmit data as.

```
/* Tx Data frame is initialized */
g_tx_dataframe.data[0]=dig1;
g_tx_dataframe.data[1]=dig2;
g_tx_dataframe.data[2]=dig3;
```

11. Since the transmit data structure gets initialized when state variable is equal to three the respective data is sent via the mailbox of transmitting board to that of the receiving board. The

transmitting board transmits it as follows .This transmission uses three out of the available 8 bytes and operations is carried out in FIFO Mode.

```
    if (FRAME_ID_MODE == STD_ID_MODE )
    {
        R_CAN_TxSet(0, CANBOX_TX, &g_tx_dataframe, DATA_FRAME);

    }
    else
    {   /* Extended ID mode. */

        R_CAN_TxSetXid(0, CANBOX_TX, &g_tx_dataframe, DATA_FRAME);

    }
       /* Transmission Code in FIFO Mode */
    #if TEST_FIFO
    /* Send three bytes to fill FIFO. */
    for (i == 0; i < 3; i++)
    {
        if (FRAME_ID_MODE == STD_ID_MODE )
        {
R_CAN_TxSetFifo(g_can_channel, &tx_fifo_dataframe, DATA_FRAME);

          }
        else
        {
R_CAN_TxSetFifoXid(g_can_channel, &tx_fifo_dataframe, DATA_FRAME);
        }
    }

    #endif


} /*  End of function sw3_function(). */
```

12. The Receiving board received the bytes in its receive mailboxes & then read them & displaysed them on its second line using an interrupt derived function .

```
   api_status = R_CAN_RxRead(g_can_channel, CANBOX_RX, g_rx_dataframe);

        /* Format read-data bytes into  a string. */
        sprintf((char *)disp_buf, "%c%c%c",
            g_rx_dataframe.data[0],
            g_rx_dataframe.data[1],
            g_rx_dataframe.data[2] );

        /* Display the formatted string. */
        lcd_display(LCD_LINE2, disp_buf);
```

13. Here in additional received data frames were discarded from the pre defined API .i.e. the fourth byte was discarded.

14. Data values of state & character space variables are re initialized as

```
 /* Re initialization of display variables& state */
  state = 1;
 dig1 = dig2 = dig3 = 65;
/* Re initialization of display*/
sprintf((char *)IST, "%c%c%c",dig1,95,95);
lcd_display(LCD_LINE1, IST);
```

15. A similar operation could be obtained by the second board irrespective of the press events on board one

Important observations

The character code was transmitted through the mailbox of one board & received in the mail box of the receiving board. This happened provided the transmitter board & receiver board mailbox were identical.

In this lab we learned.

CAN configuration

CAN Operation & Implementation

Operation of Switch ISR