RL78/F12

R01AN1839ED0100
Rev.1.00
Sep 25, 2013

LIN Slave Mode (UARTF)

Introduction

This document describes how to use the UARTF module in slave mode of LIN communication.

Target Device

RL78 F12 Group (R5F109GE)

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

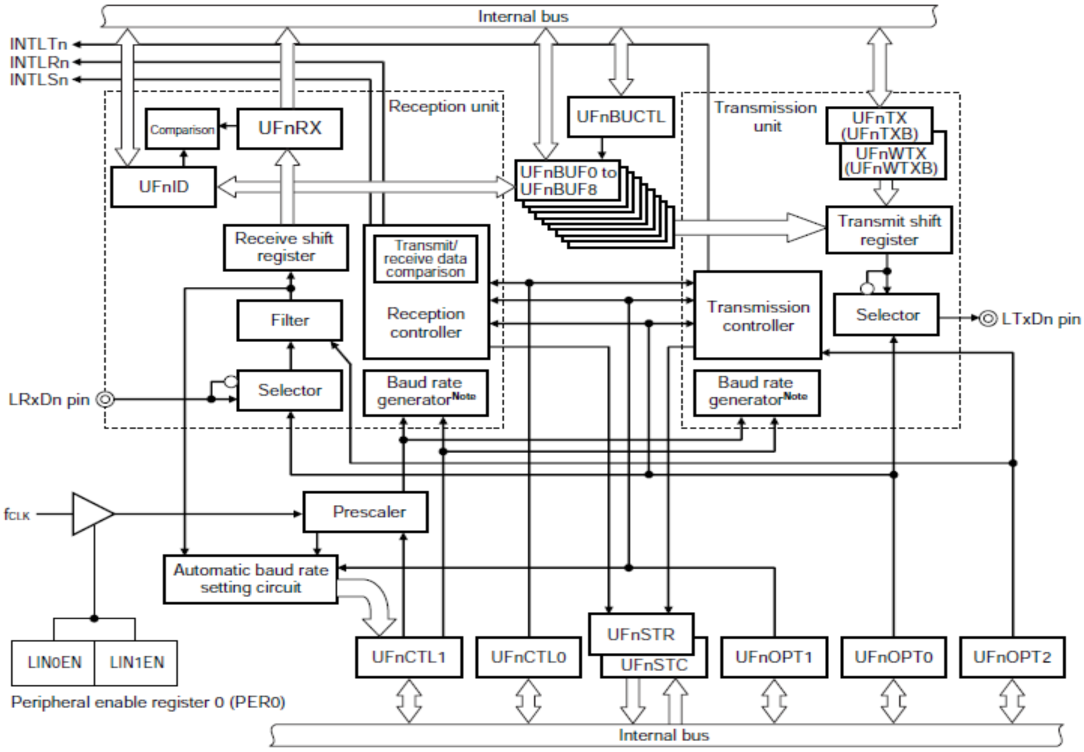
Development environment

IAR Embedded workbench for Renesas RL78 V1.30.3

Contents

1. UARTF module specifications

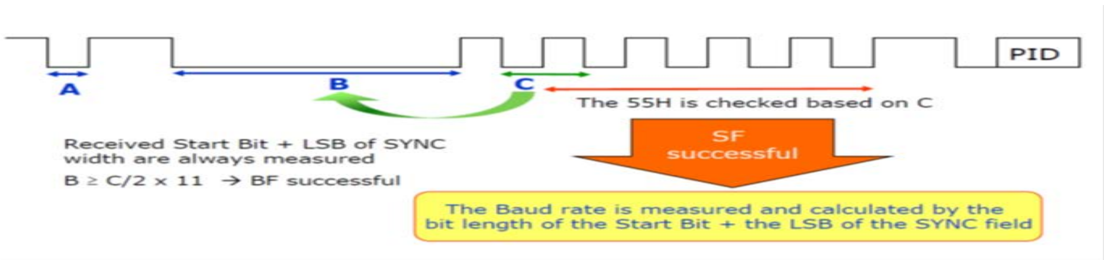
The UARTF interface supporting LIN slave and master communication



Figue1.1 UARTF Macro Block Diagram

UART functionality

- NORMAL UART MODE
  - Standard UART operation, bytewise, full-duplex transmission/reception
- LIN COMMUNICATION MODE
  - Standard UART operation, BF (LIN break field) transmission/reception
- LIN AUTO BAUD RATE MODE
  - LIN slave protocol engine using a 9 byte buffer for transmission/reception of LIN frame responses
  - Macro acts as LIN slave protocol engine
  - Frame header (SBF, SF and ID) will be recognized automatically
  - LIN baud rate will be measured and adjusted automatically based on the Sync-Field

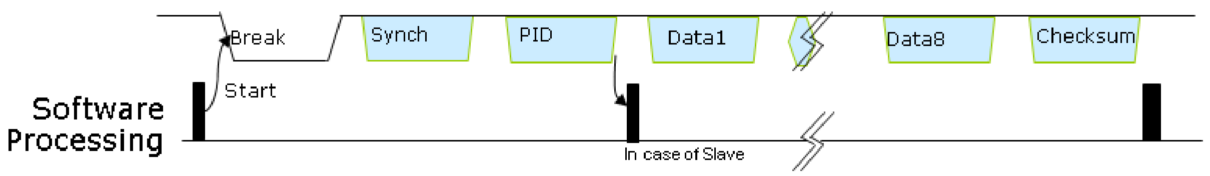


Figue1.2 Auto Baud Rate Detection

- Checksum (enhanced/classical) will be calculated automatically
- LIN response-data (up to 8 data bytes) written into the message buffer can be sent at once
- LIN response data (up to 8 bytes) can be received at once into the message buffer
- ID Parity check function, response preparation error detection
- Data consistency check function.
- Conform to LIN Specification Package Revision 1.3, 2.0, 2.1 and SAEJ2602.

▪ UART BUFFER MODE
  - Buffered transmission of up to 9 data bytes in UART format (9 bytes -> 1 Tx-IRQ)

▪ Software processing flow
  - During a complete LIN message only two interrupts are generated. The first one after the successful PID reception and the second one after the complete message.



2. Development environment

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Development environment

Item	Contents
MCU	RL78/F12 R5F109GE (ES1.0)
Operate frequencies	Xin : 20MHz System clock: 20MHz CPU clock: 20MHz
Operating voltage	5.0V for MCU, 12V for LIN transceiver
Integrated development environment	IAR Embedded workbench for Renesas RL78 V1.30.x
LIN protocol versions	V2.1
Evaluation board	See figure 2.1

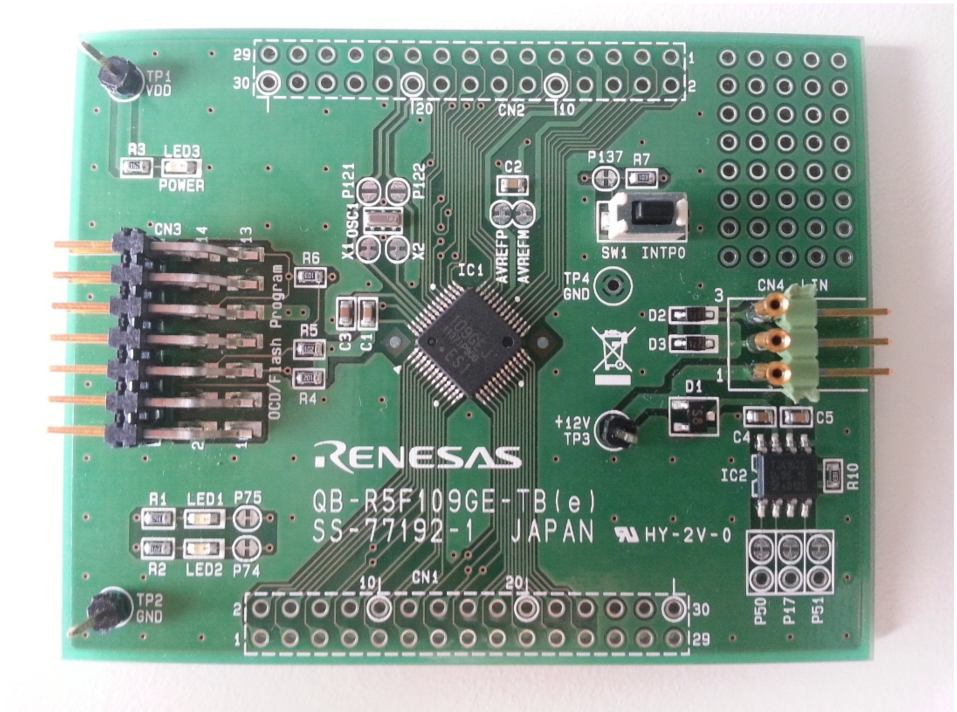


Figure 2.1 Evaluation board

3. Software

The sample code demonstrates the usage of the UARTF module implementing LIN communication. The program runs on the QB-R5F109GE-TB, which is a target board for the RL78/F12 microcontroller family including a LIN transceiver. In slave mode, the UARTF waits for reception of header frame from the master. Upon detection of the header frame, the slave checks ID and response transmission or reception according to ID. A proper communication will be indicated by LED1 and LED2 mounted on the target boards.

3.1 Operation overview

Settings:

- Use UARTF0 to perform LIN communication in slave mode.
- Use the P5.1/LTXD0 pin for the transmit data output.
- Use the P5.0/LRXD0 pin for the receive data input.
- Set the automatic baud rate mode, UARTF can automatically measure synch field and setting baud rate by itself .
- Use the INTLR interrupt; The INTLR interrupt is generated after a LIN successful header reception or response reception.
- Use the INTLS interrupt; The INTLS interrupt is generated when an Error on the bus was detected. A complete error handling is not implemented.
- Communication direction and number of transmit/receive date at a response field are determined by the ID data received at the ID field.
- ID data store in the ID buffer register UF0ID.
- Auto store data received at the field to data buffer register UF0BUF0 to UF0BUF8, then get data from ID Buffer and store to Slave_RxData1[ ], Slave_RxData2[ ], Slave_RxData3[ ]according to ID and clear the Data buffer.
- Set Slave_TxData[ ] to data buffer UF0BUF0 to UF0BUF8 and setting RTS bit to start transmission.

3.2 Functions and resource Consumption

Function Name	Outline	Code size (bytes)
LIN_Slave_Init	Initial setting	91
LIN_Slave_HeaderReceive	Header receive preparation	26
LIN_Slave_Transmit	Data transmission preparation	59
LIN_Slave_Receive	Data reception preparation	25
LIN_Slave_NoResponse	No response to LIN bus	13
Clear_databuffer	Setting data buffer to 0	26
Get_response_RxData	Store data to variables array from Data buffer	51

Table 3.1 lists the Functions

3.3 Function Specifications

The following tables list the sample code function specifications

LIN_Slave_Init	
Outline	Initial setting of UARTF's registers
Header	None
Declaration	void LIN_Slave_Init(void)
Description	Setting clock, auto baud rate, enable interrupts, header format.
Arguments	None
Returned value	None

Table 3.2 LIN_Slave_Init

LIN_Slave_HeaderReceive	
Outline	Header receive preparation
Header	None
Declaration	void LIN_Slave_headerReceive(void)
Description	Setting LIN in auto baud rate mode, set reception start
Arguments	None
Returned value	None

Table 3.3 RLIN_lave_HeaderReceive

LIN_Slave_Transmit		
Outline	Data transmission preparation	
Header	None	
Declaration	void LIN_Slave_Transmit(uint8_t * databuf, uint8_t data_length)	
Description	Setting data buffer and response transmission start	
Arguments	uint8_t * databuf	Transmit data
	uint8_t data_length	Transmission data length
Returned value	None	

Table 3.4 LIN_Slave_Transmit

LIN_Slave_Receive		
Outline	Data reception preparation	
Header	None	
Declaration	void LIN_Slave_Receive(uint8_t data_length)	
Description	Clear data buffer, setting reception format, response reception start	
Arguments	Uint8_t data_length	Receive data length
Returned value	None	

Table 3.5 LIN_Slave_Receive

LIN_NoResponse	
Outline	No response to LIN bus
Header	None
Declaration	void LIN_Slave_NoResponse(void)
Description	Slave node does not response anything when ID invalid
Arguments	None
Returned value	None

Table 3.6 LIN_NoResponse

Clear_DataBuffer	
Outline	Clear all data buffer to 0
Header	None
Declaration	void Clear_DataBuffer(void)
Description	Clear the complete data buffer
Arguments	None
Returned value	None

Table 3.7 Clear_DataBuffer

Get_response_RxData		
Outline	Store data to variable array from ID buffer	
Header	None	
Declaration	uint8_t Get_response_RxData(uint8_t * RxData)	
Description	Get reception data to variable array	
Arguments	Uint8_t * RxData	Data variable array
Returned value	RxData[2]	

Table 3.8 Get_response_RxData

3.4 Flowcharts

3.4.1 Main flowchart

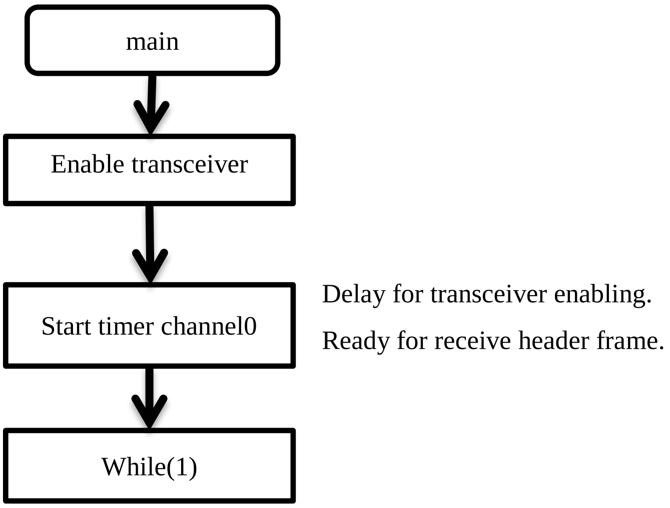


Figure 3.1 show the main processing

3.4.2 Initial RLIN flowchart

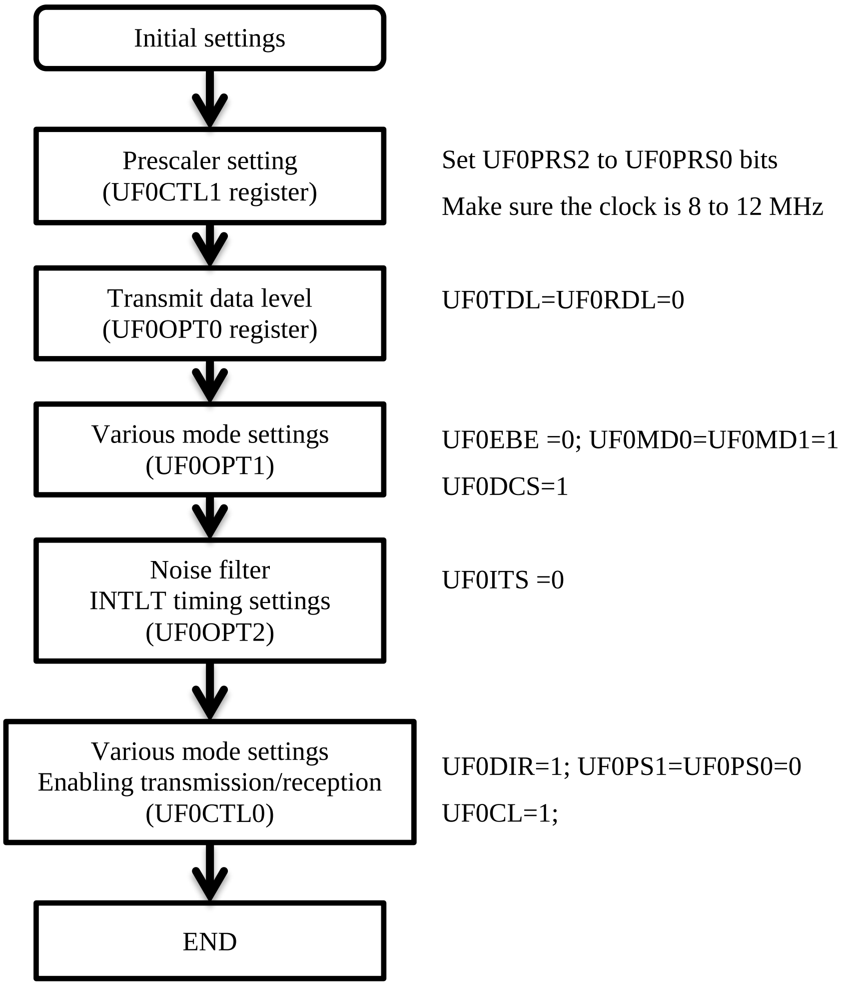


Figure 3.2 show the LIN initial processing

3.4.3 Slave transmit flowchart

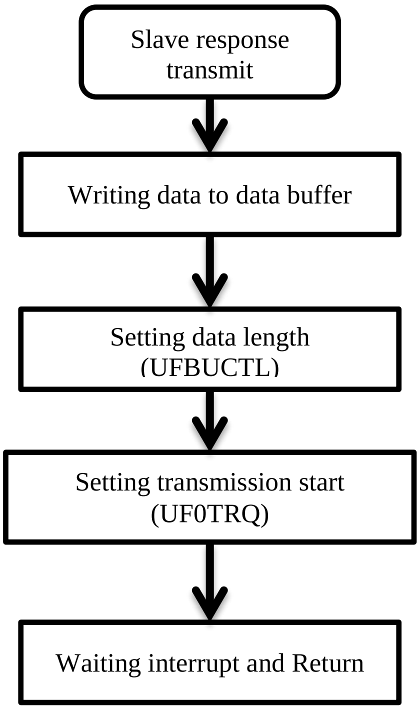


Figure 3.3 show the slave transmit processing

3.4.4 Slave receive flowchart

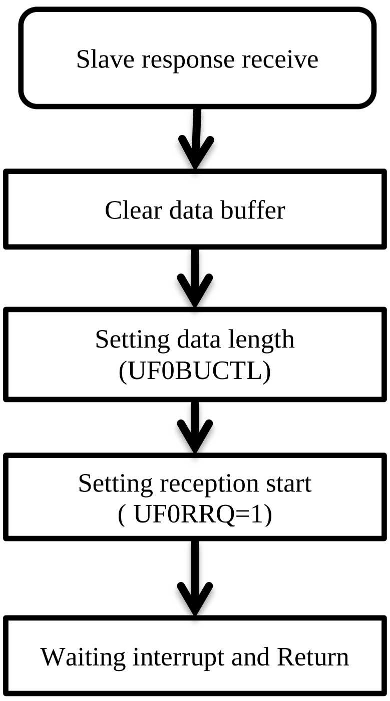


Figure 3.4 show the slave receive processing

3.4.5 Slave interrupt flowcharts

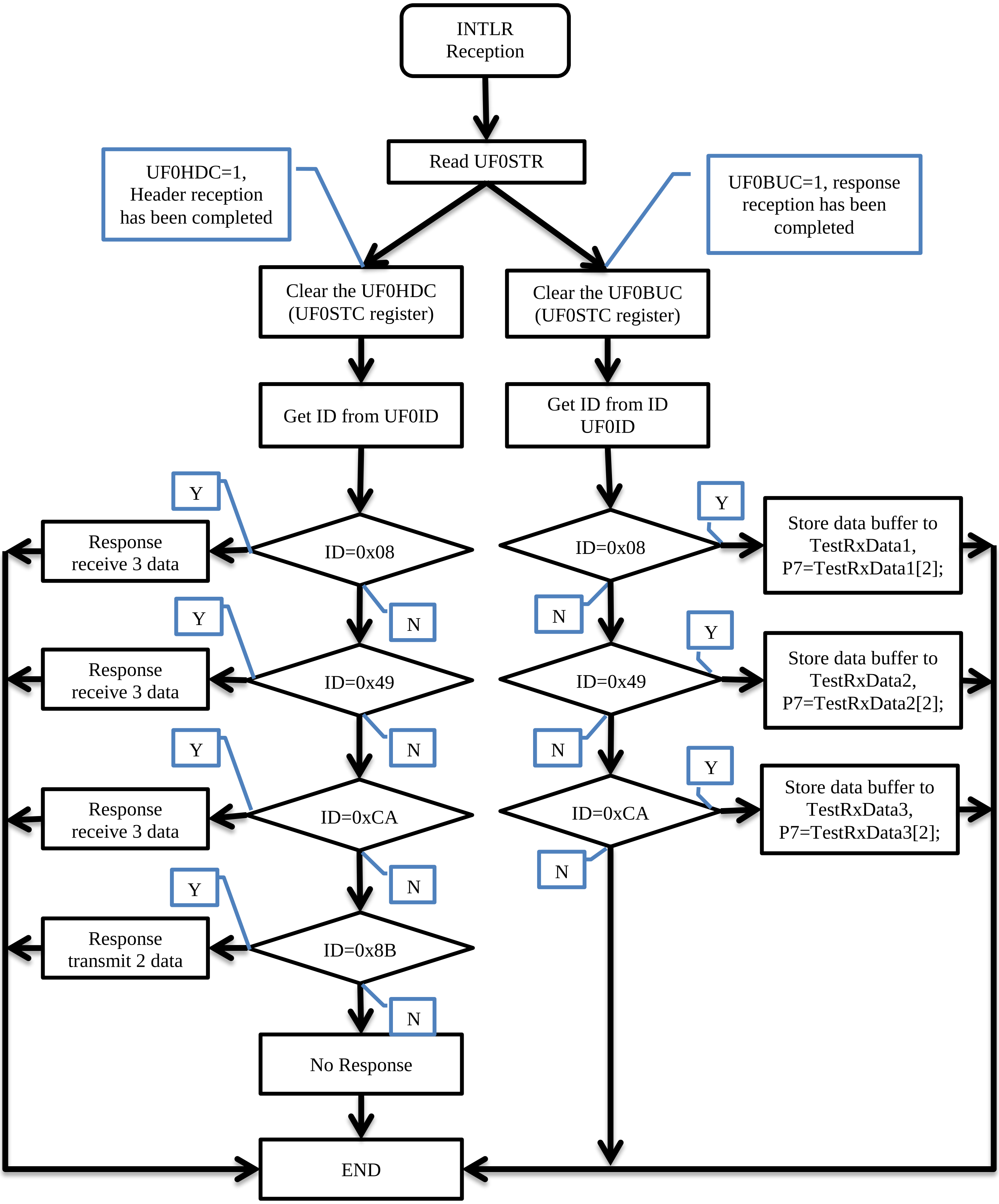


Figure 3.5 show the reception interrupt processing

4. Demo system

The below pictures shows the demo system consists out of RL78/F12 and RL78/F14 target boards. RL78/F14 board is running in master mode and the RL78/F12 board in slave mode. The software from the RL78/F12 slave mode is part of this application note, where the RL78/F14 master mode is described in a separate document. Both boards are connected via the LIN interface. The slave is indicating proper data communication via the two LEDs mounted on the target boards

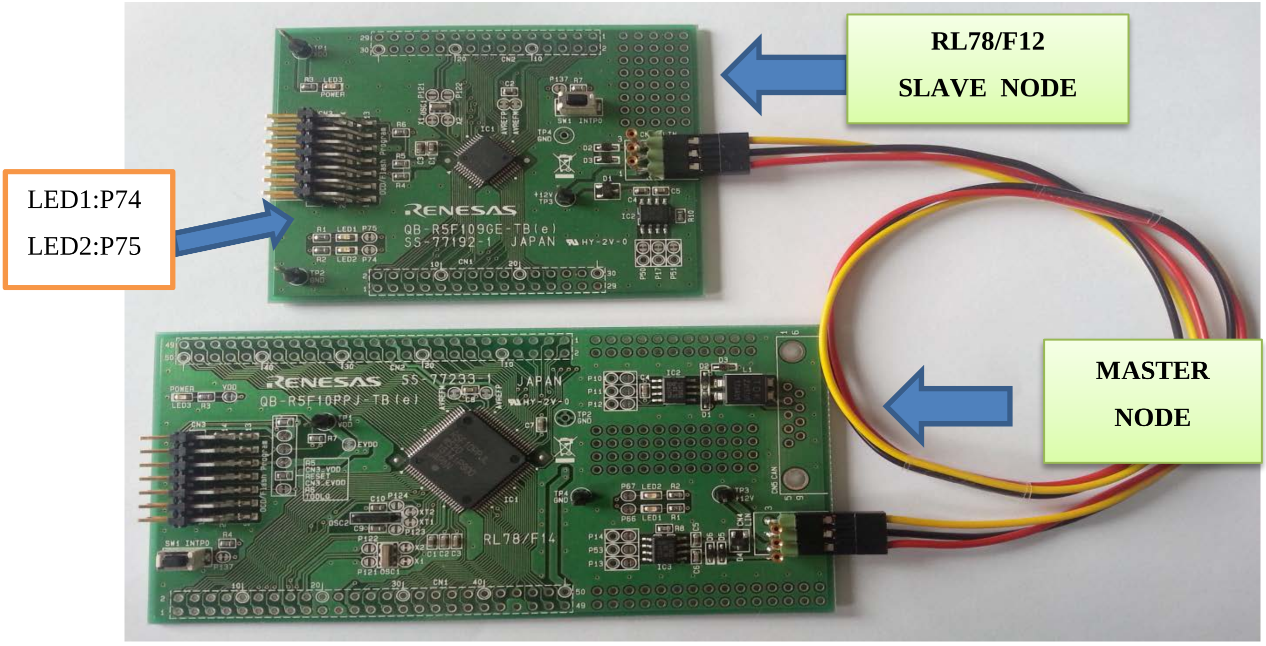
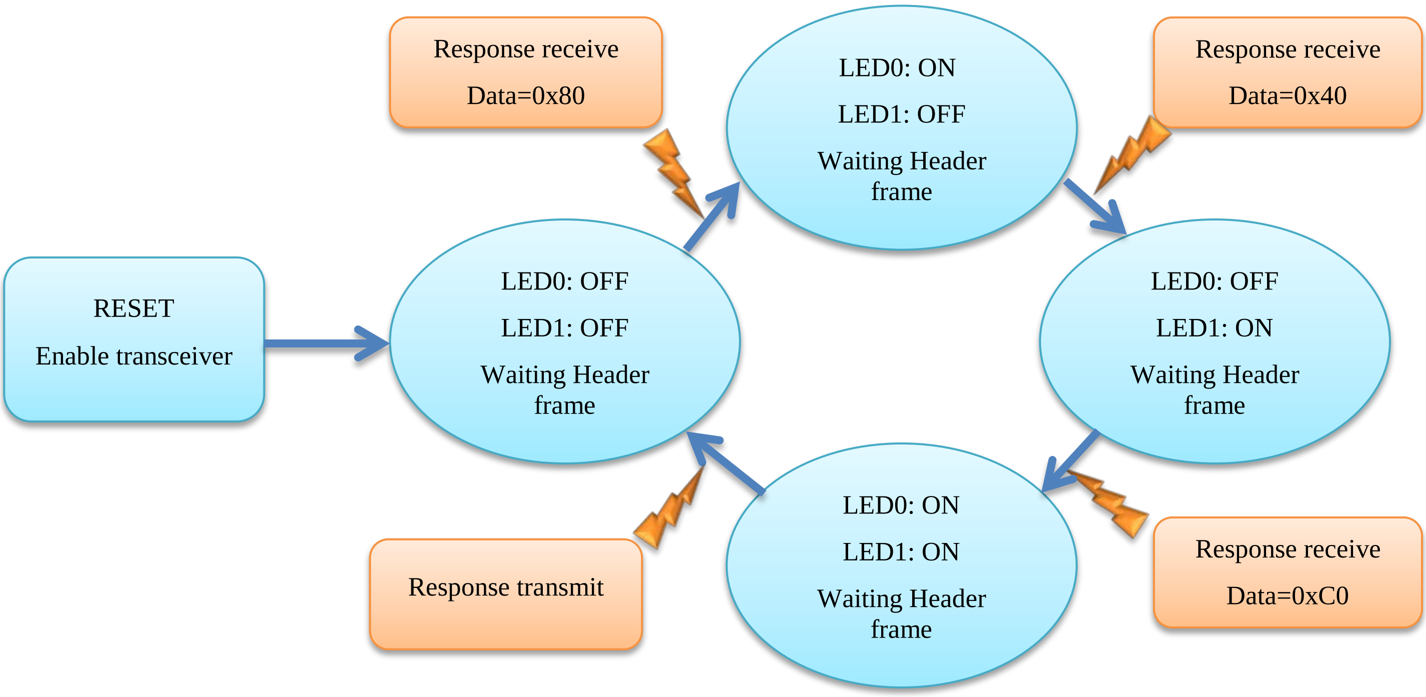


Figure 4.1 Picture of the demo system

In the below state diagram of the slave you will find the different internal states of the slave demo with the corresponding LED

Slave Demo :



5. Sample code

5.1 LIN_driver.c

```
/************************************************************************************************
* File Name    : LIN_Slave_driver.c
* Device(s)    : R5F109GE
* Tool-Chain   : IAR Systems iccrl78
* Description  : This file implements device driver for Serial module.
* Creation Date: 2013/8/22
************************************************************************************************/

/************************************************************************************************
Includes
************************************************************************************************/
#include "LIN_Slave_macrodriver.h"
#include "LIN_Slave_driver.h"
#include "LIN_Slave_userdefine.h"
/************************************************************************************************
* Function Name: LIN_Slave_Init
* Description  : This function initializes the UARTF0 module.
* Arguments    : None
* Return Value : None
************************************************************************************************/
void LIN_Slave_Init(void)
{
   UF0EN = 1U;
   UF0CTL0 &= (uint8_t)(~_40_UARTF_TRANSMISSION_ENABLE & ~_20_UARTF_RECEPTION_ENABLE);
/* disable UARTF0 operation */
   LTMK0 = 1U;  /* disable INTLT interrupt */
   LTIF0 = 0U;  /* clear INTLT interrupt flag */
   LRMK0 = 1U;  /* disable INTLR interrupt */
   LRIF0 = 0U;  /* clear INTLR interrupt flag */
   LSMK0 = 1U;  /* disable INTLS interrupt */
   LSIF0 = 0U;  /* clear INTLS interrupt flag */
   /* Set INTLT level1 priority */
   LTPR10 = 0U;
   LTPR00 = 1U;
   /* Set INTLR high priority */
   LRPR10 = 0U;
```

```
    LRPR00 = 0U;
    /* Set INTLS low priority */
    LSPR10 = 1U;
    LSPR00 = 1U;

    UF0CTL1 = _2000_UARTF_BASECLK_2 | _0823_UARTF0_K_VALUE;    /*baud rate 2400bps  20/2/0823/2 */
    UF0OPT0 = _14_UARTF_UFNOPT0_INITIALVALUE | _00_UARTF_TRAN_DATALEVEL_NORMAL |
_00_UARTF_REC_DATALEVEL_NORMAL;  /*13bit BF, transmit and receive data level bit = normal*/
    UF0OPT1 = _00_UARTF_EXPANSIONBIT_UNUSE | _06_LIN_UF0MD | _01_LIN_UF0DCS |
_10_LIN_UF0IPCS ;   /* automatic baud rate mode*/
    UF0OPT1 |= _08_LIN_UF0ACE;
    UF0OPT2 = _00_UARTF_LT_INT_GENTIME_0 | _02_UARTF_DATA_NOISE_FILTER_UNUSED;
    UF0CTL0 = _10_UARTF_TRANSFDIR_LSB | _00_UARTF_PARITY_NONE |
_02_UARTF_DATALENGTH_8BIT | _00_UARTF_STOPLENGTH_1BIT;
    /* Set LTXD0 pin */
    PM5 |= 0x02U;
    PMX2 = 0xFEU;
    /* Set LRXD0 pin */
    PM5 |= 0x01U;
    PU5 |= 0x01U;   /* Has to Pull up P5.0*/
}

/*******************************************************************************************************
* Function Name: LIN_Slave_Start
* Description  : This function starts the UARTF0 operation.
* Arguments    : None
* Return Value : None
*******************************************************************************************************/
void LIN_Slave_HeaderReceive(void)
{
    LTIF0 = 0U;  /* clear INTLT interrupt flag */
    LTMK0 = 0U;  /* enable INTLT interrupt */
    LRIF0 = 0U;  /* clear INTLR interrupt flag */
    LRMK0 = 0U;  /* enable INTLR interrupt */
    LSIF0 = 0U;  /* clear INTLS interrupt flag */
    LSMK0 = 0U;  /* enable INTLS interrupt */
    UF0CTL0 |= _40_UARTF_TRANSMISSION_ENABLE | _20_UARTF_RECEPTION_ENABLE; /* enable
UARTF0 operation */
}

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

```
* Function Name: LIN_Slave_Stop
* Description  : This function stops the UARTF0 operation.
* Arguments    : None
* Return Value : None
*******************************************************************************************************/
void LIN_Slave_Stop(void)
{
   UF0CTL0 &= (uint8_t)(~_40_UARTF_TRANSMISSION_ENABLE & ~_20_UARTF_RECEPTION_ENABLE);
/* disable UARTF0 operation */
   LTMK0 = 1U;  /* disable INTLT interrupt */
   LTIF0 = 0U;  /* clear INTLT interrupt flag */
   LRMK0 = 1U;  /* disable INTLR interrupt */
   LRIF0 = 0U;  /* clear INTLR interrupt flag */
   LSMK0 = 1U;  /* disable INTLS interrupt */
   LSIF0 = 0U;  /* clear INTLS interrupt flag */
}

/*******************************************************************************************************
* Function Name: LIN_Slave_Receive
* Description  : This function receives UARTF0 data.
* Arguments    : Data_length
*                receive data numbers
* Return Value : None
*******************************************************************************************************/
   void LIN_Slave_Receive(uint16_t Data_length)
   {
     Clear_DataBuffer();
     UF0BUCTL = 0x00A0;    /* 0000 0000 1010 0010;  UF0ECS=1: Enhanced checksum; UF0RRQ=1:Reception start
*/
     UF0BUCTL |= Data_length;    }

/*******************************************************************************************************
* Function Name: Get_reponse_RxData
* Description  : This function get data buffer value to a variable array
* Arguments    : uint8_t * RxData : a avriable array for store Data
* Return Value : RxData[2]
*******************************************************************************************************/
uint8_t Get_response_RxData(uint8_t * RxData)
{
  uint16_t i,k;
  uint16_t Databuf_adr;
```

```
  k=UF0BUCTL&0x000F;
 Databuf_adr=UARTF0_BUFFER_ADDRESSS;   /* get the data buffer address*/
 for(i=0;i<k;i++)
 {
   RxData[i]=(*((uint8_t *)(Databuf_adr+i)));
 }
 return RxData[2];
}
/**********************************************************************************************
* Function Name: LIN_Slave_Transmit(void)
* Description  : This function seting data buffer for response transmission start
* Arguments    : uint8_t* databuf    : variable array data.
               uint8_t Data_length : transmit data length.
* Return Value : None
**********************************************************************************************/
void LIN_Slave_Transmit(uint8_t* TxData,uint16_t Data_length)
{
 uint16_t i;
 uint16_t  Databuf_adr;
 Databuf_adr=UARTF0_BUFFER_ADDRESSS;   /* get the data buffer address*/
 for(i=0;i<Data_length;i++)            /* setting tansmission data to date buffer*/
 {
   *((uint8_t *)(Databuf_adr+i))=TxData[i];
 }

 UF0BUCTL = 0x0290;   /* 0000 0010 1001 0000; UF0TW=1; UF0ECS=1: Enhanced checksum;
UF0TRQ=1:Transmission start */
 UF0BUCTL |= Data_length;
}
/**********************************************************************************************
* Function Name: LIN_Slave_NoResponse(void)
* Description  : This function setting register when PID is not match.
* Arguments    : None
* Return Value : None
**********************************************************************************************/
void LIN_Slave_NoResponse()
{
UF0BUCTL |= 0x00C0;    /*UF0NO=1: NO Response requit*/
}
/**********************************************************************************************
* Function Name: Clear_DataBuffer
```

```
* Description  : This function setting all data buffer to some value
* Arguments    : None
* Return Value : None
***********************************************************************************************/
void Clear_DataBuffer()
{
  uint8_t i;
  uint16_t Databuf_adr;
  Databuf_adr=UARTF0_BUFFER_ADDRESSS;
  for(i=0;i<9;i++)
  {
   *((uint8_t *)(Databuf_adr+i))=0U;
  }
}
```

5.2 LIN_driver_user.c

```
/***********************************************************************************************
* File Name    : LIN_Slave_Driver_user.c
* Device(s)    : R5F109GE
* Tool-Chain   : IAR Systems iccrl78
* Description  : This file implements device driver for Serial module.
* Creation Date: 2013/8/22
***********************************************************************************************/

/***********************************************************************************************I
includes
***********************************************************************************************/
#include "LIN_Slave_macrodriver.h"
#include "LIN_Slave_driver.h"
#include "LIN_Slave_userdefine.h"
/***********************************************************************************************
Global variables and functions
***********************************************************************************************/

uint8_t Slave_RxData1[8];                /*reception data store array*/
uint8_t Slave_RxData2[8];                /*reception data store array*/
uint8_t Slave_RxData3[8];                /*reception data store array*/
uint8_t Slave_TxData[8]={0x8B,0xC0};  /*Transmission data store array*/
```

```
/***************************************************************************************************
* Function Name: LIN_Slave_interrupt_receive
* Description  : This function is INTLR interrupt service routine.
* Arguments    : None
* Return Value : None
***************************************************************************************************/
#pragma vector = INTLR_vect
__interrupt static void LIN_Slave_interrupt_receive(void)
{
  uint16_t  header_receive_flag;
  uint16_t  buffer_receive_flag;
  uint8_t  PID;

  header_receive_flag = UF0STR & 0x0800;
  buffer_receive_flag = UF0STR & 0x0400;
  if(header_receive_flag != 0)
  {
  UF0STC |= 0x0800;    /* clear UF0HDC*/
  PID = UF0ID;
  switch (PID)
  {
  case 0x08 : LIN_Slave_Receive(3);
            break;
  case 0x49 : LIN_Slave_Receive(3);
            break;
  case 0xCA : LIN_Slave_Receive(3);
            break;
  case 0x8B : LIN_Slave_Transmit(Slave_TxData, 2);
            LED1=OFF;
            LED2=OFF;
            break;
   default:   break;
  }
  }

  if(buffer_receive_flag != 0)
  {
    PID = UF0ID;
    switch (PID)
  {
```

```
    case 0x08 :    P7 = Get_response_RxData(Slave_RxData1);
                   break;
    case 0x49 :    P7 = Get_response_RxData(Slave_RxData2);
                   break;
    case 0xca  :   P7 = Get_response_RxData(Slave_RxData3);
                   break;
    default:   LIN_Slave_NoResponse();
                  break;
  }
     UF0STC |= 0x0400;   /*Clear UF0BUC*/
  }
}

/*****************************************************************************************************
* Function Name: LIN_Slave_interrupt_send
* Description  : This function is INTLT interrupt service routine.
* Arguments    : None
* Return Value : None
*****************************************************************************************************/
#pragma vector = INTLT_vect
__interrupt static void LIN_Slave_interrupt_send(void)
{
}
/*****************************************************************************************************
* Function Name: LIN_Slave_interrupt_error
* Description  : This function is INTLS interrupt service routine.
* Arguments    : None
* Return Value : None
*****************************************************************************************************/
#pragma vector = INTLS_vect
__interrupt static void LIN_Slave_interrupt_error(void)
{
  while(1)
  {    ;
  }
}
```

5.3 LIN _driver.h

```
/*************************************************************************************************
* File Name    : LIN_Slave_Driver.h
* Device(s)    : R5F109GE
* Tool-Chain   : IAR Systems iccrl78
* Description  : This file implements device driver for Serial module.
* Creation Date: 2013/8/22
*************************************************************************************************/
#ifndef SERIAL_H
#define SERIAL_H
/*************************************************************************************************
Macro definitions (Register bit)
*************************************************************************************************/
/*
   UARTFn control register 0 (UFnCTL0)
*/
#define _10_UARTF_UFNCTL0_INITIALVALUE          (0x10U)
/* Transmission operation enable (UFnTXE) */
#define _00_UARTF_TRANSMISSION_DISABLE          (0x00U)   /* disable transmission operation */
#define _40_UARTF_TRANSMISSION_ENABLE           (0x40U)   /* enable transmission operation */
/* Reception operation enable (UFnRXE) */
#define _00_UARTF_RECEPTION_DISABLE             (0x00U)   /* disable reception operation */
#define _20_UARTF_RECEPTION_ENABLE              (0x20U)   /* enable reception operation */
/* Transfer direction selection (UFnDIR) */
#define _00_UARTF_TRANSFDIR_MSB                 (0x00U)   /* MSB-first transfer */
#define _10_UARTF_TRANSFDIR_LSB                 (0x10U)   /* LSB-first transfer */
/* Parity selection during transmission/reception (UFnPS1, UFnPS0) */
#define _00_UARTF_PARITY_NONE                   (0x00U)   /* no parity */
#define _04_UARTF_PARITY_ZREO                   (0x04U)   /* 0 parity */
#define _08_UARTF_PARITY_ODD                    (0x08U)   /* odd parity */
#define _0C_UARTF_PARITY_EVEN                   (0x0CU)   /* even parity */
/* Specification of data character length of 1 frame of transmit/receive data (UFnCL) */
#define _00_UARTF_DATALENGTH_7BIT               (0x00U)   /* 7 bits */
#define _02_UARTF_DATALENGTH_8BIT               (0x02U)   /* 8 bits */
/* Specification of length of stop bit for transmit data (UFnSL) */
#define _00_UARTF_STOPLENGTH_1BIT               (0x00U)   /* 1 bit */
#define _01_UARTF_STOPLENGTH_2BIT               (0x01U)   /* 2 bits */

/*
```

```
    UARTFn control register (UFnCTL1)
*/
/* Prescaler clock frequency division value (UFnPRS2 - UFnPRS0)*/
#define _0000_UARTF_BASECLK_1             (0x0000U) /* fXX */
#define _2000_UARTF_BASECLK_2             (0x2000U) /* fXX/2^1 */
#define _4000_UARTF_BASECLK_4             (0x4000U) /* fXX/2^2 */
#define _6000_UARTF_BASECLK_8             (0x6000U) /* fXX/2^3 */
#define _8000_UARTF_BASECLK_16            (0x8000U) /* fXX/2^4 */
#define _A000_UARTF_BASECLK_32            (0xA000U) /* fXX/2^5 */
#define _C000_UARTF_BASECLK_64            (0xC000U) /* fXX/2^6 */
#define _E000_UARTF_BASECLK_128           (0xE000U) /* fXX/2^7 */

/*
    UARTFn option control register 0 (UFnOPT0)
*/
#define _14_UARTF_UFNOPT0_INITIALVALUE         (0x14U)
/* Transmit data level bit (UFnTDL) */
#define _00_UARTF_TRAN_DATALEVEL_NORMAL        (0x00U)   /* normal output of transfer data */
#define _02_UARTF_TRAN_DATALEVEL_INVERTED      (0x02U)   /* inverted output of transfer data */
/* Receive data level bit (UFnRDL) */
#define _00_UARTF_REC_DATALEVEL_NORMAL         (0x00U)   /* normal input of transfer data */
#define _01_UARTF_REC_DATALEVEL_INVERTED       (0x01U)   /* inverted input of transfer data */

/*
    UARTFn option control register 1 (UFnOPT1)
*/
/* Transmit data expansion bit enable bit (UFnEBE) */
#define _00_UARTF_EXPANSIONBIT_UNUSE           (0x00U)   /* disable expansion bit */
#define _80_UARTF_EXPANSIONBIT_USE             (0x80U)   /* enable expansion bit */
/* Transmit data expansion bit detection level (UFnEBL) */
#define _00_UARTF_EXPANSIONBIT_VALUE_0         (0x00U)   /* expansion value 0 */
#define _40_UARTF_EXPANSIONBIT_VALUE_1         (0x40U)   /* expansion value 1 */
/* Transmit data expansion bit data comparsion enable bit (UFnEBC) */
#define _00_UARTF_EXPANSIONBIT_COMP_UNUSE      (0x00U)   /* disable expansion bit with comparsion */
#define _20_UARTF_EXPANSIONBIT_COMP_USE        (0x20U)   /* enable expansion bit with comparsion */
/* Communication mode (UFnMD1, UFnMD0) */
#define _00_UARTF_NORMAL_MODE                  (0x00U)   /* normal mode */
#define _06_LIN_UF0MD                          (0x06U)   /* automatic baud rate mode*/
#define _01_LIN_UF0DCS                         (0x01U)   /* check data consistency*/
#define _10_LIN_UF0IPCS                        (0x10)    /* automatic check PID*/
```

```
#define _08_LIN_UF0ACE                      (0x08)   /* automation checksum*/
/*
   UARTFn option control register 1 (UFnSTC)
*/
/* Normal error flag clear trigger (UFnOVE, UFnFE, UFnPE) */
#define _0007_UARTF_COMMONERROE_CLEAR          (0x0007U) /* clear commom error flag bit */
/* Buffer transmission/reception completion flag clear trigger (UFnCLBUC) */
#define _0400_UARTF_BUC_CLEAR                  (0x0400U) /* clear buffer transmit/reception completion flag bit */
/* Expansion bit detection flag clear trigger (UFnCLEBD) */
#define _0100_UARTF_EBD_CLEAR                  (0x0100U) /* clear EBD flag */
/* ID match flag clear trigger (UFnCLIDM) */
#define _0200_UARTF_IDM_CLEAR                  (0x0200U) /* clear IDM flag */

/*
   UARTFn option control register 2 (UFnOPT2)
*/
/* Bit to select use of receive data noise filter (UFnRXFL) */
#define _00_UARTF_DATA_NOISE_FILTER_USED       (0x00U)   /* enables noise filter */
#define _02_UARTF_DATA_NOISE_FILTER_UNUSED     (0x02U)   /* disables noise filter */
/* Transmission interrupt (INTLTn) generation timing select bit (UFnITS) */
#define _00_UARTF_LT_INT_GENTIME_0             (0x00U)   /* output INTTn upon transmit completion */
#define _01_UARTF_LT_INT_GENTIME_1             (0x01U)   /* output INTTn upon transmit start */

/*
   UARTFn buffer control register (UFnBUCTL)
*/
/* Buffer length bits (UFnBUL3~UFnBUL0) */
#define _0001_UARTF_BUFFER_LENGTH_1            (0x0001U) /* buffer length 1 byte */
#define _0002_UARTF_BUFFER_LENGTH_2            (0x0002U) /* buffer length 2 bytes */
#define _0003_UARTF_BUFFER_LENGTH_3            (0x0003U) /* buffer length 3 bytes */
#define _0004_UARTF_BUFFER_LENGTH_4            (0x0004U) /* buffer length 4 bytes */
#define _0005_UARTF_BUFFER_LENGTH_5            (0x0005U) /* buffer length 5 bytes */
#define _0006_UARTF_BUFFER_LENGTH_6            (0x0006U) /* buffer length 6 bytes */
#define _0007_UARTF_BUFFER_LENGTH_7            (0x0007U) /* buffer length 7 bytes */
#define _0008_UARTF_BUFFER_LENGTH_8            (0x0008U) /* buffer length 8 bytes */
#define _0009_UARTF_BUFFER_LENGTH_9            (0x0009U) /* buffer length 9 bytes */
/* Buffer transmission request bit (UFnTRQ) */
#define _0000_UARTF_BUFFER_TRAN_START_NOREQUEST  (0x0000U) /* no transmission start request */
#define _0010_UARTF_BUFFER_TRAN_START_REQUEST    (0x0010U) /* transmission start request */
```

```
/* Buffer address */
#define UARTF0_BUFFER_ADDRESSS          (0x052FU) /* UARTF0 transmit buffer address in buffer mode */


/******************************************************************************************************
Macro definitions
******************************************************************************************************/
/* Selection of 16-bit counter output clock (UF0BRS7 - UF0BRS0) */
#define _0823_UARTF0_K_VALUE            (0x0823U)

/******************************************************************************************************
Typedef definitions
******************************************************************************************************/

/******************************************************************************************************
Global functions
******************************************************************************************************/
void LIN_Slave_Init(void);
void LIN_Slave_HeaderReceive(void);
void LIN_Slave_Stop(void);
void LIN_Slave_Receive(uint16_t Data_length);
void LIN_Slave_Transmit(uint8_t * TxData, uint16_t Data_length);
uint8_t Get_response_RxData(uint8_t * RxData);
void LIN_Slave_NoResponse(void);
void Clear_DataBuffer(void);
#endif
```

5.4 LIN_main.c

```
/*************************************************************************************************
* File Name    : LIN_Slave_main.c
* Device(s)    : R5F109GE
* Tool-Chain   : IAR Systems iccrl78
* Description  : This file implements main function.
* Creation Date: 2013/8/22
*************************************************************************************************/

/*************************************************************************************************
Includes
*************************************************************************************************/
#include "LIN_Slave_macrodriver.h"
#include "LIN_Slave_cgc.h"
#include "LIN_Slave_port.h"
#include "LIN_Slave_driver.h"
#include "LIN_Slave_timer.h"
#include "LIN_Slave_userdefine.h"
#include "LIN_Slave_wdt.h"

/*************************************************************************************************
************************
Global variables and functions
**************************************************************************************************
***********************/
/* Set option bytes */
#pragma location = "OPTBYTE"
__root const uint8_t opbyte0 = 0x79U;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte1 = 0xFFU;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte2 = 0xE9U;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte3 = 0x84U;

/* Set security ID */
#pragma location = "SECUID"
__root const uint8_t secuid[10] =
        {0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U};
/* Start user code for global. Do not edit comment generated here */
```

```
/* End user code. Do not edit comment generated here */
/***********************************************************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************************************************/
void  main(void)
{
  LED1=OFF;
  LED2=OFF;
  LIN_Enable=TRUE;
  R_TAU0_Channel0_Start();
    while (1U)
    {
       R_WDT_Restart() ;
    }
}
```

Website and Support <website and support,ws>

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

Inquiries
http://www.renesas.com/contact/

Revision Record <RL78/F1x Application note RLIN3 in slave mode >

		Description	
Rev.	**Date**	**Page**	**Summary**
1.0	25.Sep.2013		First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different type number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141