LIN 从机自动波特率使用指南

概述

本手册适用范围：

适用范围
SPC1169，SPD1179，SPD1176，SPC1185，SPC2188

本文描述了 LIN 从机自动波特率功能的使用。

本手册中：

- SPD1179、SPD1176 内部集成 LIN PHY，不需要外挂 LIN PHY 使用；
- SPC1169、SPC1185、SPC2188 内部没集成 LIN PHY，需要外挂 LIN PHY 使用。

注意： LIN 主机永远没有自动波特率的概念。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-08-10	Hengfang Huang	Outdated	1. 首次发布。
C/0	2024-07-10	Hang Su	Released	1. 更新章节 1，章节 2。

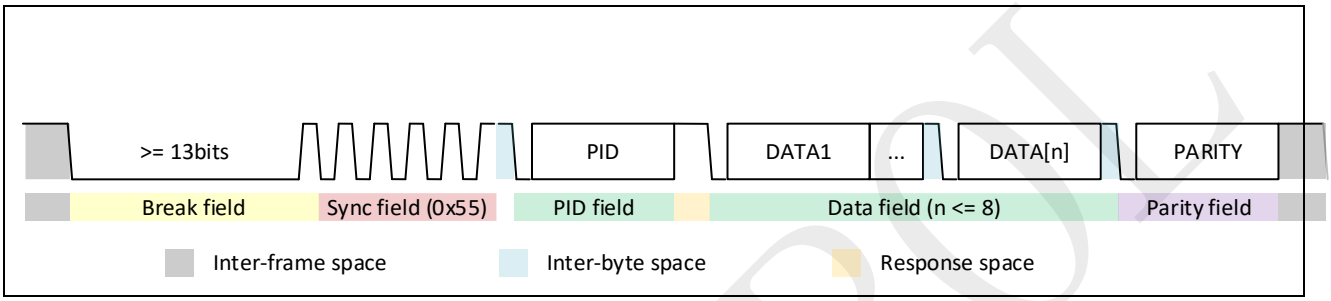
1 LIN 从机自动波特率

1.1 从机 Break field 识别

LIN 从机自动波特率匹配，核心在于从机是否能够正确识别出主机发出的 Break field。

主机按照**主机波特率**发出 LIN 帧，如图 1-1 所示。

图 1-1：LIN 帧



对从机来说，当 UARTCTL.ABEN 置为 1 后（开启自动波特率检测），开始监控总线，并将**从机波特率下**检出的 11bit 低电平认定为 Break filed。

注意：
1. 主机波特率由主机时钟以及主机侧 UARTBDCNT 寄存器决定；
2. 从机波特率由从机时钟以及从机侧 UARTBDCNT 寄存器决定。

表 1-1：从机 11bit 低电平检测

主从波特率比	从机 11bit 低电平检测	可用性
$ratio > \frac{13}{11}$	主机 Break field 的 13bit 低电平长度 < 从机连续 11bit 低电平检测长度。	在主机 Break field 固定为 13bit 时（默认），该$ratio$区间不可用；在主机 Break field 可调为 >13bit 时，该$ratio$区间可用。
$\frac{9}{11} < ratio < \frac{13}{11}$	主机 Break field 的 13bit 低电平长度 > 从机连续 11bit 低电平检测长度； 主机 Data field 的 9bit 低电平长度 < 从机连续 11bit 低电平检测长度。	该$ratio$区间可用
$ratio < \frac{9}{11}$	主机 Break field 的 13bit 低电平长度 > 从机连续 11bit 低电平检测长度； 当$\frac{8}{11} < ratio < \frac{9}{11}$时，主机 Data field 的 9bit 低电平 >从机连续 11bit 低电平检测长度； 当$\frac{7}{11} < ratio < \frac{8}{11}$时，主机 PID field 或 Data field 的 8bit 低电平 >从机连续 11bit 低电平检测长度；	该$ratio$区间可用，但存在误检。 1，如果不能将 UARTCTL.ABEN 置 1 时机控制到 Inter-frame space（帧间隔），会**误**

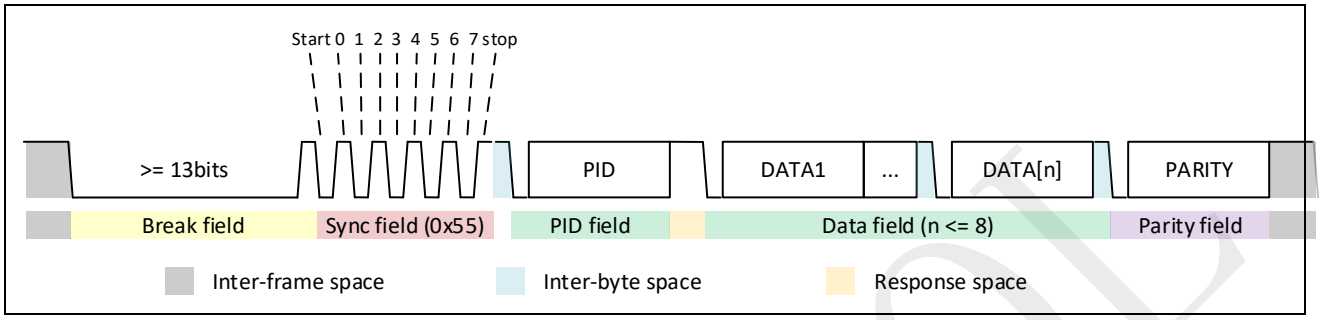
主从波特率比	从机 11bit 低电平检测	可用性
	…… 当$\frac{2}{11} < ratio < \frac{3}{11}$时，主机 PID field 或 Data field 的 3bit 低电平 >从机连续 11bit 低电平检测长度； 当$\frac{1}{11} < ratio < \frac{2}{11}$时，主机 PID field 或 Data field 的 2bit 低电平 >从机连续 11bit 低电平检测长度。	2，当 $ratio$ 靠近 $\frac{1}{11}$ 时，主机帧头会超过从机最大长度，此时从机会放弃已经识别的 Break field。

考虑到$\frac{9}{11} < ratio < \frac{13}{11}$区间使用起来限制条件最少，通常采用该区间进行自动波特率同步。例如：在主机波特率 20,000bps 下，根据$\frac{9}{11} < ratio < \frac{13}{11}$，从机波特率位于$(16{,}924, 24{,}444)$时，可以自动同步。

1.2 从机自动波特率计算

从机自动波特率计算，以 Sync filed 的下降沿为开始计算的判断标志（Sync filed 的字节域如图 1-2 所示，为数字 0x55，转换为二进制位 01010101b）。

图 1-2：Sync filed



1 位时间计算公式如下：

$$\mathbf{1}\text{位时间} = \frac{\text{第}\mathbf{7}\text{位的下降沿时刻} - \text{起始位的下降沿时刻}}{\mathbf{8}}$$

波特率与 1 位时间关系：

$$BaudRate = \frac{1}{\mathbf{1}\text{位时间}}$$

正常通信下，从机完成第 7 位的下降沿识别后，硬件关闭自动波特率功能（UARTCTL.ABEN 自动清 0），并将计数值锁定到 UARTBDCNT 寄存器，完成同步。

UARTBDCNT 与波特率对应关系：

$$\text{UARTBDCNT} = \frac{\text{对应 UART clock}}{BaudRate}$$

1.3 总线 Sync filed 故障

总线 Sync filed 故障会导致从机自动波特率失败。

1.3.1 使用自动波特率

使用自动波特率需要考虑 Sync filed 上的总线故障，如表 1-2 所示。

表 1-2：自动波特率匹配时故障注入

主机故障注入	从机第 7 位的下降沿识别	从机动作	下一帧帧前准备
主机 0x55 临时提升波特率发出； 主机 0x55 临时降低波特率发出； 主机 0x55 某 bit 高电平被拉低； ……	从机可以识别到第 7 位的下降沿时刻（**若主机 0x55 某 bit 高电平被拉低，Sync field 中第 7 位的下降沿识别推迟到 PID field**）	同步完成，硬件关闭自动波特率功能（UARTCTL.ABEN 自动清 0），值锁定到 UARTBDCNT 寄存器。	将 UARTCTL.ABEN 置为 1 以重新开启从机自动波特率，在下一正常帧时将能够自动匹配波特率，恢复通讯。
主机 0x55 发送过程掉电； **主机 0x55 以及 PID 所有 bit 高电平被拉低，且发完 PID 后不再发送其他 Data；** ……	从机无法识别到第 7 位的下降沿时刻	同步无法完成，自动波特率功能维持开启（UARTCTL.ABEN 维持 1），UARTBDCNT 寄存器维持原状。	由于 UARTCTL.ABEN 维持 1，下一正常帧时将自动匹配波特率，恢复正常通信。

若主机 0x55 某 bit 高电平被拉低，Sync field 中第 7 位的下降沿识别推迟到 PID field，识别到从机波特率会偏小，此时 ratio 会变大，如果$ratio > \frac{13}{11}$，从机将再也检测不到主机。

一旦发现从机无法检测到主机，需要重新设定从机波特率，以恢复 ratio 到区间$\left(\frac{9}{11}, \frac{13}{11}\right)$内。

注意： 1. 自动波特率下，不能使用 LINSBERR（LIN received sync byte error）中断。

1.3.2 不使用自动波特率

相比使用自动波特率，不使用自动波特率下，硬件自身即可过滤掉异常 Sync filed 所在帧。

1. 如果主机发出 Sync field 不是 0x55，或者虽然是 0x55 但是频率不符合从机预期，LIN 硬件会自动进入新的 Break field 检测周期；
2. 如果下一帧是正常帧，则正常通信；否则回到步骤 1。

2 LIN 从机波特率应用场景

狭义上 LIN 从机自动波特率，指的是在 LIN 主机时钟校准，从机时钟有±14%偏差的情况下能够同步主机 LIN 帧的波特率。

实际使用中会结合表 2-1，表 1-1，根据具体应用场景进行开发。

表 2-1：主从时钟组合

主机时钟	从机时钟
校准	校准
校准	未校准
未校准	校准

2.1 固定波特率通讯

固定波特率通信应用在开机后主机和从机波特率不会发生变更的通信场景（最常见）。

2.1.1 不使用同步

若 LIN 主从时钟均校准，可将主从波特率直接设为一致，从而不需要使用自动波特率同步，这种方式不会因为主从双方 Sync filed 长短不一致导致通信通信失败，所以相对而言抗干扰能力比自动波特率的方式要强一些。

2.1.2 首帧同步

首帧同步是 SDK 默认的配置方式，UARTCTL.ABEN 置 1 位于 LIN_Init()中。

2.1.2.1 场景一：

若 LIN 主从时钟均校准，可通过首帧同步，使得从机可匹配波特率不完全一致的主机（调节范围$\frac{9}{11} < ratio < \frac{13}{11}$）。

例如：在从机波特率 20,000bps 下，不使用自动同步则只能匹配波特率 20,000bps 主机，开机过程使用自动波特率同步后，其可以匹配波特率（16,364bps 到 23,636bps）的主机。**根据表 1-1，可匹配主机波特率可下探到 2,400bps（$ratio < \frac{9}{11}$），但不推荐使用，因为存在误识别的可能。**

0_Examples 下的 24_1_LIN_Slave_RX 以及 24_2_LIN_Slave_TX 均采用此同步方式。

2.1.2.2 场景二：

若主机时钟校准，从机时钟有静态偏差，可在开机后调用自动波特率完成从机波特率校准（调节范围$\frac{9}{11} < ratio < \frac{13}{11}$）

例如：主机波特率 20,000bps，从机设定波特率也是 20,000bps，但实际有偏差，在从机实际波特率位于（16,924bps 到 24,444bps）之间时，可被校准。**根据表 1-1，可匹配从机波特率可超过 24,444bps（$ratio < \frac{9}{11}$），但不推荐使用，因为存在误识别的可能。**

2.2 非固定波特率通讯

非固定波特率通信应用在开机后主机或从机波特率会发生变更的通信场景。

2.2.1 每帧同步

就是在每次通信结束后，将 UARTCTL.ABEN 再次置为 1，以使能下一帧的自动波特率。

2.2.1.1 场景一：

若 LIN 主从时钟均校准，可通过每帧同步，使得从机可匹配波特率在一定范围内变化的主机（调节范围$\frac{9}{11} < ratio < \frac{13}{11}$）。

例如：在从机波特率 20,000bps 下，其可匹配主机波特率在（16,364bps 到 23,636bps）范围内。

- 第一帧主机 16,364bps，从机采用 16,364bps；
- 第二帧主机 23,636bps，从机采用 23,636bps；
- ……

注意：

1. 每次同步后，都需要将从机波特率再次设为 20,000bps，否则由于同步后从机波特率发生改变，其下一帧可匹配主机波特率范围不再是（16,364bps 到 23,636bps）；
2. 例如：首帧主机波特率 16,364bps，同步后从机波特率变为 16,364bps，此时可匹配主机波特率在（13,389bps 到 19,339bps）范围内。
3. 每次同步后，因为硬件已经把 UARTCTL.ABEN 自动清 0，需要在 Inter-frame space（帧间隔）将 UARTCTL.ABEN 再次置为 1，使能下一帧的自动波特率同步；
4. 根据表 1-1，可匹配主机波特率可下探到 2,400bps（$ratio < \frac{9}{11}$），但不推荐使用，因为存在误识别的可能。

1_Application 下的 LIN_Slave_node 采用此同步方式。

2.2.2 宽范围波特率同步

LIN 主从时钟均校准，当主机波特率在 2400 和 20,000 之间变化的时候，从机能够自动跟随主机波特率。

在本示例中，主机波特率按照 192,00-> 4800-> 9600-> 2400-> 20,000 的顺序间隔 20s 依次切换，且主机每间隔 100ms 会发送帧头。

因为单次匹配范围只有$\frac{9}{11} < ratio < \frac{13}{11}$，为了完成宽范围的主机波特率匹配，只能引入软件层协议，每当从机匹配不上主机时，就将从机波特率设到 2400，尝试匹配，如果因为此时$ratio > \frac{13}{11}$导致匹配不成功，则逐渐提升从机波特率（以减少$ratio$），并重新尝试匹配，直到$\frac{9}{11} < ratio < \frac{13}{11}$，匹配成功。

从机初始波特率设定到主机波特率下限 2400。初始状态机设定到 BAUDRIGHT（连通态），状态机每 500ms 更新一次。

在 BAUDRIGHT（连通态），从机会检测 500ms 内正确通讯次数（检测后会清零），小于 3 次认为通讯异常，跳转到 BAUDINIT（初始态）。

BAUDINIT（初始态）会设定波特率到下限 2400，开启自动波特率，跳转到 BAUDUP（向上计数态）。

BAUDUP（向上计数态）会将波特率提升 14%，开启自动波特率。判断波特率是否超过上限（这里选为 23000），超过跳转到 BAUDINIT（初始态）。

Get ID 后会跳转到 BAUDRIGHT（连通态）。

注意：	如果采用 UART0 进行 ISP 下载，需要确保下载过程 LIN 上无电平翻转，否则 boot 会进入 LIN 下载模式。

2.2.2.1 从机代码

如下示例代码的配置步骤为：

- 调用 LIN_Init，设定为从机；
- 开启高压模块，完成 LIN PHY 的设定；
- 设定过滤值，示例过滤出 bit[0]==1 的 PID，例如 0x11, 0x21, 0x31，其他的帧头被丢弃，也可将过滤功能关闭；
- 过滤出的帧头会触发中断，在中断中设定校验方式，并根据从 PID 解析出的 ID，设定为发送方式并设定发送长度 8byte；
- 因为关闭校验段的自动发送，TRANSMIT1 置 1 会将校验段发出。

Example Code

```
/*****************************************************************************
 * @file     main.c
 * @brief    Main program body
 * @version  VXX.XX.XX
 * @date     XX-XXX-XXXX
 *
 * @note
 * Copyright (C) 2022 Spintrol Electronic Technology (Shanghai) Co., Ltd.. All
rights reserved.
 *
```

Example Code

```
 * @attention
 * THIS SOFTWARE JUST PROVIDES CUSTOMERS WITH CODING INFORMATION REGARDING
 * THEIR PRODUCTS, WHICH AIMS AT SAVING TIME FOR THEM. SPINTROL SHALL NOT BE
 * LIABLE FOR THE USE OF THE SOFTWARE. SPINTROL DOES NOT GUARANTEE THE
 * CORRECTNESS OF THIS SOFTWARE AND RESERVES THE RIGHT TO MODIFY THE SOFTWARE
 * WITHOUT NOTIFICATION.
 *
*******************************************************************************/
#include "spd1179.h"
#include <stdio.h>
#include <stdlib.h>

#define             BAUDINIT            0
#define             BAUDUP              1
#define             BAUDRIGHT           2
#define             BAUDRATE            2400            /* LIN Rate */
#define             REFID               0x01            /* REFID State */
#define             IDMASK              0x01            /* IDMASK State */
uint8_t             u8Id;                               /* Receive Id */
uint8_t             u8Txd[8];                           /* TX Data */
uint8_t             i;
uint16_t            u16PREDRIID;                        /* PRE-DRIVER mode ID */
ErrorStatus         eErrorState;                        /* Function State */
uint8_t             u8State             = BAUDRIGHT;
uint32_t            u32Count            = 0;


/*******************************************************************************
*****************************************
 *
 * @brief     In this case, the LIN slave send the data to master.
 *
 *       Key_points:
 *                (1) If unconnected, the baud of slave add from 2400
 *

*******************************************************************************
*****************************************/


int main(void)
{
    CLOCK_InitWithRCO(100000000);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* HV init */
    eErrorState = HV_Init(&u16PREDRIID);
    if (eErrorState == ERROR)
    {
        printf("Init HV mode FAIL\n");
```

Example Code

```
        return 0;
    }
    else
    {
        printf("Init HV mode SUCCESS[ID:%d]\n", u16PREDRIID);
    }

    /* HV parameter write enable */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_USER_REG);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    /*
    Init LIN parameter
    when Baud rate 9600, LINCTL_TXSLOPE_19P0US
    when Baud rate 19200, LINCTL_TXSLOPE_7PUS
    Because of autobaud function in slave, parameter must be set according to
master baud
    */
    eErrorState = EPWR_WriteRegisterField(HV_REG_LINCTL, LINCTL_TXSLOPE_Msk |
LINCTL_STRENGTH_Msk | LINCTL_TXEN_Msk | LINCTL_EN_Msk, LINCTL_TXSLOPE_19P0US |
LINCTL_STRENGTH_47P2MA | LINCTL_TXEN_ENABLE | LINCTL_EN_ENABLE);
    if (eErrorState == ERROR)
    {
        printf("LINCTL_WriteRegisterField FAIL\n");
        return 0;
    }

    /* LIN_Init */
    PIN_SetChannel(PIN_GPIO25, PIN_GPIO25_UART1_TXD);
    PIN_SetChannel(PIN_GPIO26, PIN_GPIO26_UART1_RXD);
    LIN_Init(UART1, LIN_SLAVE, BAUDRATE);

    /* Set the id filter */
    LIN_SetIDFilter(UART1, REFID, IDMASK);

    /* Enable the RX time out INT */
    UART_EnableInt(UART1, UART_INT_LIN_ID_MATCH | UART_INT_RX_TIMEOUT |
UART_INT_RX_FRAME_ERROR | UART_INT_LIN_BIT_ERROR | UART_INT_AUTOBAUD_LOCK);

    /* Enable UART1_IRQn trigger INT in MCU side */
    NVIC_EnableIRQ(UART1_IRQn);

    /* Init Timer */
    TIMER_Init(TIMER1, 500);

    /* Open Global INT for Timer */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    while (1)
    {
    }
}
```

Example Code

```
void UART1_IRQHandler(void)
{
    if (UART_GetIntFlag(UART1, UART_INT_LIN_ID_MATCH) != 0)
    {
        /* Set the check mode, the check mode must be set before receive the ID,
otherwise the set will be ignored */
        LIN_SetCheckSumMode(UART1, LIN_ENHANCED_CHECKSUM);

        /* Get the id */
        u8Id = LIN_GetRxID(UART1);

        u8Id = u8Id & 0x3F;

        /* Set the respond mode */
        LIN_SetResponse(UART1, LIN_RESPONSE_TX);

        /* Set the respond lenth */
        LIN_SetResponseLen(UART1, LIN_RESPONSE_8_BYTE);

        /* Set the respond data */
        for (i = 0; i < LIN_RESPONSE_8_BYTE; i++)
        {
            u8Txd[i] = i;
            u8State = BAUDRIGHT;
        }

        for (i = 0; i < LIN_RESPONSE_8_BYTE; i++)
        {
            UART_WriteByte(UART1, u8Txd[i]);
        }

        SET_BITS(UART1->LINCTL, LINCTL_TXCHKSUM_TRANSMIT);

        for (i = 0; i < LIN_RESPONSE_8_BYTE; i++)
        {
            printf("The send data is %x\n", u8Txd[i]);
        }

        printf("baud %d\n",
CLOCK_GetModuleClock(UART0_MODULE)/UART_GetBaudCount(UART1));

        u32Count++;


        /* Clear the start signal flag */
        UART_ClearInt(UART1, UART_INT_LIN_ID_MATCH);
    }
    /* GetRxTimeoutIntFlag */
    else if (UART_GetIntFlag(UART1, UART_INT_RX_TIMEOUT) != 0)
    {
        printf("LIN Rx Time out\n");
        UART_ClearInt(UART1, UART_INT_RX_TIMEOUT);
    }
    /* GetRxStopbitErrorIntFlag */
    else if (UART_GetIntFlag(UART1, UART_INT_RX_FRAME_ERROR) != 0)
    {
        printf("LIN Rx Stop bit error\n");
        UART_ClearInt(UART1, UART_INT_RX_FRAME_ERROR);
    }
    /* GetTxBitErrorIntFlag */
    else if (UART_GetIntFlag(UART1, UART_INT_LIN_BIT_ERROR) != 0)
```

Example Code

```
    {
        printf("LIN Tx bit error\n");
        UART_ClearInt(UART1, UART_INT_LIN_BIT_ERROR);
    }
    /* GetAutobaudlockFlag */
    if (UART_GetIntFlag(UART1, UART_INT_AUTOBAUD_LOCK) != 0)
    {
        /* The BaudCount must be set once again */
        UART_SetBaudCount(UART1, UART_GetBaudCount(UART1));

        UART_ClearInt(UART1, UART_INT_AUTOBAUD_LOCK);
    }
    UART_ClearInt(UART1, UART_INT_ALL);
}


void TIMER1_IRQHandler()
{
    if (u8State == BAUDINIT)
    {
        UART_SetBaudCount(UART1, BAUDRATE);
        UART_EnableAutoBaud(UART1);
        u8State = BAUDUP;
    }
    else if (u8State == BAUDUP)
    {
        if (u32Count == 0)
        {
            UART_SetBaudCount(UART1, UART_GetBaudCount(UART1) * 114 / 100);
            UART_EnableAutoBaud(UART1);
        }
        else
        {
            u8State = BAUDRIGHT;
        }
    }
    else
    {
        if (u32Count < 3)
        {
            u8State = BAUDINIT;
        }
        u32Count = 0;
    }

    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}


/****************** Copyright (C) 2022 Spintrol Electronic Technology
(Shanghai) Co., Ltd. ***** END OF FILE ****/
```

[1] 示例为 SPD1179，SPD1176 代码。

2.2.2.2 主机代码

如下示例代码的配置步骤为：

- 调用 LIN_Init，其设定了 LIN 帧的超时阀值，设定为主机；
- 开启高压模块，完成 LIN PHY 的设定；
- 设定 RX_REQ 的阀值 8byte；
- 设定为接收方式并设定接收长度 8byte，设定校验方式；
- 发送帧头；
- 当 FIFO 中长度大于 8byte 时，触发中断，将 FIFO 中 9byte 数据读出，包含校验位。

Example Code

```
#define             BAUDRATE            19200           /* LIN Rate */


uint8_t             u8Id                = 0x31;         /* Send Id */
uint16_t            u16PREDRIID;                        /* PRE-DRIVER mode ID */
uint16_t            u16PREDRIDATA;                      /* Data read from PRE-
DRIVER */
ErrorStatus         eErrorState;                        /* Function State */
uint8_t             i;                                 /* Print Num */
volatile uint8_t    u8Rxd[9];                           /* RX Data */
uint32_t            u32Baud             = BAUDRATE;

/*******************************************************************************
****************************************
 *
 * @brief     In this case, the LIN master receive the data from slave.
 *
 *       Key_points:
 *                (1)First start the slave code to wait the master code to send
data.
 *                (2)Reset the baud rate before each communication with slave
 *                (3)Select the Master pull-up resistor based on the load
capacitance,
 *                observation point is between LIN and GND, when Load
capacitance 10nF,
 *                Pull-up resistor 500 ou; when Load capacitance 8nF, Pull-up
resistor
 *                660 ou; when Load capacitance 1nF, Pull-up resistor 1000 ou
 *                (4)Master baud must = or < slave baud, otherwise autobaud in
slave do
 *                 not work.
 *
 *                              observation point
 *
 *                            *    *
 *                           *    *
 *            *************     *    *     LIN     *************
 *            *          ****************************          *
 *            *  Master   *       *               *   Slave   *
 *            *          ****************************          *
 *            *************                GND     *************
 *

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

Example Code

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_50MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* HV init */
    eErrorState = HV_Init(&u16PREDRIID);
    if (eErrorState == ERROR)
    {
        printf("Init HV mode FAIL\n");
        return 0;
    }
    else
    {
        printf("Init HV mode SUCCESS[ID:%d]\n", u16PREDRIID);
    }

    /* HV parameter write enable */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_USER_REG);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    /*
    Init LIN parameter
    when Baud rate 9600, LINCTL_TXSLOPE_19P0US
    when Baud rate 19200, LINCTL_TXSLOPE_7PUS
    */
    eErrorState = EPWR_WriteRegisterField(HV_REG_LINCTL, LINCTL_TXSLOPE_Msk |
LINCTL_STRENGTH_Msk | LINCTL_TXEN_Msk | LINCTL_EN_Msk, LINCTL_TXSLOPE_19P0US |
LINCTL_TXSLOPE_7PUS | LINCTL_TXEN_ENABLE | LINCTL_EN_ENABLE);
    if (eErrorState == ERROR)
    {
        printf("LINCTL_WriteRegisterField FAIL\n");
        return 0;
    }

    /* LIN_Init */
    PIN_SetChannel(PIN_GPIO25, PIN_GPIO25_UART1_TXD);
    PIN_SetChannel(PIN_GPIO26, PIN_GPIO26_UART1_RXD);
    LIN_Init(UART1, LIN_MASTER, BAUDRATE);

    UART_SetRxFIFOThreshold(UART1, LIN_RESPONSE_8_BYTE);

    /* Enable the RX time out INT */
    UART_EnableInt(UART1, UART_INT_RX_REQ | UART_INT_RX_TIMEOUT |
UART_INT_RX_FRAME_ERROR | UART_INT_LIN_BIT_ERROR);
```

Example Code

```
    /* Enable UART1_IRQn trigger INT in MCU side */
    NVIC_EnableIRQ(UART1_IRQn);

    /* Init Timer */
    TIMER_Init(TIMER1, 20000);

    /* Open Global INT for Timer */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    while (1)
    {
        /* Set the respond mode */
        LIN_SetResponse(UART1, LIN_RESPONSE_RX);

        /* Set the respond lenth */
        LIN_SetResponseLen(UART1, LIN_RESPONSE_8_BYTE);

        /* Set the check mode */
        if (u8Id == 0x3C || u8Id == 0x3D)
        {
            LIN_SetCheckSumMode(UART1, LIN_CLASSIC_CHECKSUM);
        }
        else
        {
            LIN_SetCheckSumMode(UART1, LIN_ENHANCED_CHECKSUM);
        }

        /* Set the id, then the communication is start*/
        LIN_SetTxID(UART1, u8Id);
        UART_SetBreak(UART1);

        /* Wait slave */
        Delay_Ms(100);
    }
}


void UART1_IRQHandler(void)
{
    if (UART_GetIntFlag(UART1, UART_INT_RX_REQ) != 0)
    {
        for (i = 0; i < LIN_RESPONSE_8_BYTE + 1; i++)
        {
            u8Rxd[i] = UART_ReadByte(UART1);
        }

        for (i = 0; i < LIN_RESPONSE_8_BYTE + 1; i++)
        {
            printf("u8Rxd[%d] is %x\n", i, u8Rxd[i]);
        }

        /* Clear the start signal flag */
        UART_ClearInt(UART1, UART_INT_RX_REQ);
    }
    /* GetRxTimeoutIntFlag */
    else if (UART_GetIntFlag(UART1, UART_INT_RX_TIMEOUT) != 0)
    {
```

Example Code

```
        printf("LIN Rx Time out\n");
        LIN_SetResponse(UART1, LIN_RESPONSE_NONE);
        UART_ClearRxFIFO(UART1);
        UART_ClearTxFIFO(UART1);
        UART_ClearInt(UART1, UART_INT_RX_TIMEOUT);
    }
    /* GetRxStopbitErrorIntFlag */
    else if (UART_GetIntFlag(UART1, UART_INT_RX_FRAME_ERROR) != 0)
    {
        printf("LIN Rx Stop bit error\n");
        LIN_SetResponse(UART1, LIN_RESPONSE_NONE);
        UART_ClearRxFIFO(UART1);
        UART_ClearTxFIFO(UART1);
        UART_ClearInt(UART1, UART_INT_RX_FRAME_ERROR);
    }
    /* GetTxBitErrorIntFlag */
    else if (UART_GetIntFlag(UART1, UART_INT_LIN_BIT_ERROR) != 0)
    {
        printf("LIN Tx bit error\n");
        LIN_SetResponse(UART1, LIN_RESPONSE_NONE);
        UART_ClearRxFIFO(UART1);
        UART_ClearTxFIFO(UART1);
        UART_ClearInt(UART1, UART_INT_LIN_BIT_ERROR);
    }
    UART_ClearInt(UART1, UART_INT_ALL);
}


void TIMER1_IRQHandler()
{
    UART_Disable(UART1);

    if (u32Baud == 20000)
    {
        u32Baud = 19200;
        UART_InitSpeed(UART1, u32Baud);
    }
    else if (u32Baud == 19200)
    {
        u32Baud = 4800;
        UART_InitSpeed(UART1, u32Baud);
    }
    else if (u32Baud == 4800)
    {
        u32Baud = 9600;
        UART_InitSpeed(UART1, u32Baud);
    }
    else if (u32Baud == 9600)
    {
        u32Baud = 2400;
        UART_InitSpeed(UART1, u32Baud);
    }
    else if (u32Baud == 2400)
    {
        u32Baud = 20000;
        UART_InitSpeed(UART1, u32Baud);
    }

    UART_ClearRxFIFO(UART1);
    UART_ClearTxFIFO(UART1);
    UART_Enable(UART1);
```

Example Code

```
    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

[1] 示例为 SPD1179，SPD1176 代码。