SPD1179 LIN 校准使用指南

版本 A/0 – 2023 年 5 月

概述

LIN 总线是一种低成本、低速度、短距离的串行通信总线，主要用于车内通讯和控制系统。它通常用于连接车内设备（如仪表板、车门控制器、座椅控制器等）和主控制器（如汽车电脑）。LIN 总线使用单线通信，具有较低的数据传输速率（通常在 20Kbps 左右），总线电平 12V。

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 年 5 月 6 日	Hang Su	Released	首次发布。

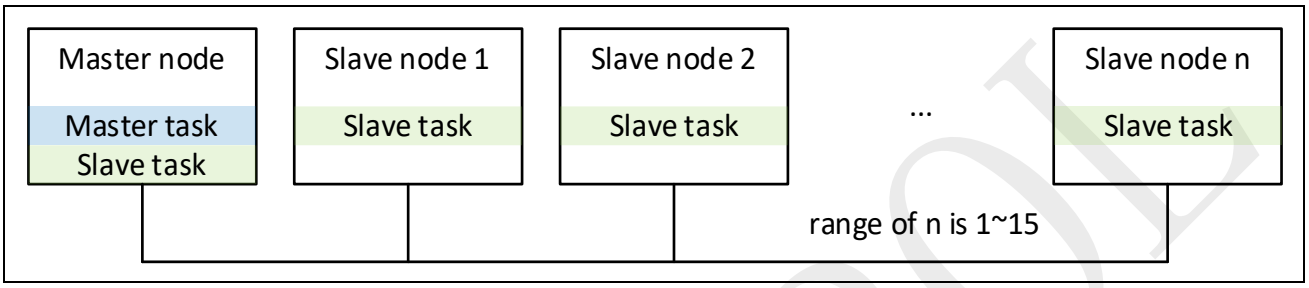
术语或缩写

术语或缩写	描述

1 LIN 概述

LIN 的总线的数据传输速率较低，通常在 20Kbps 左右，总线电平 12V。其总线连接如图 1-1 所示。LIN 总线拓扑图包括 1 个主节点和最多 15 个从节点。每个节点都有一个从任务，只有主节点有一个主任务，负责发送帧头。

图 1-1：LIN 总线连接

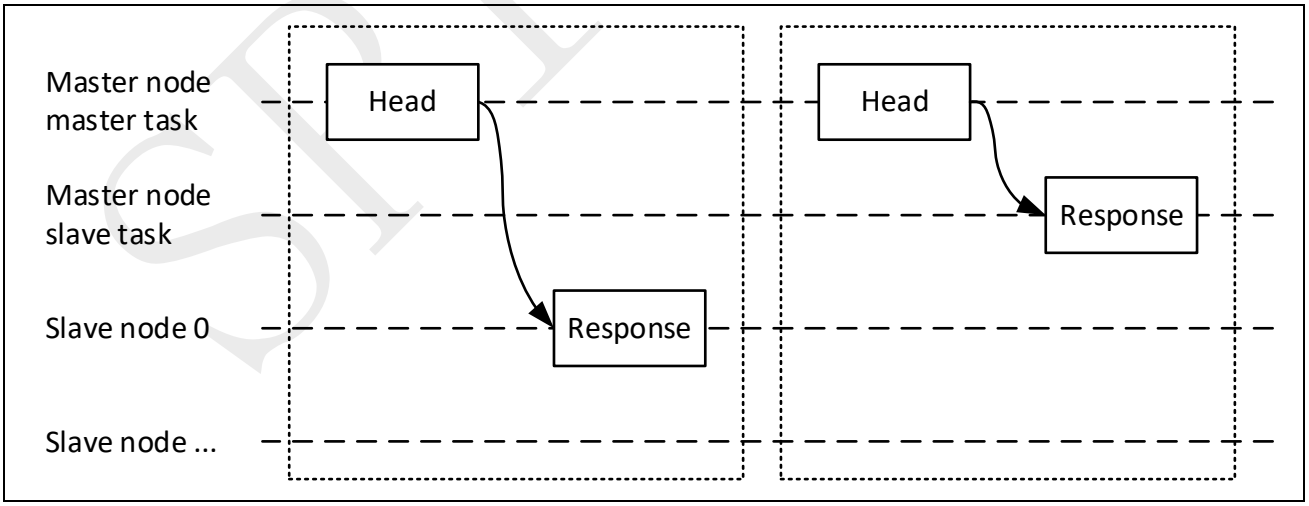


1.1 LIN 帧的响应方式

LIN 帧的响应方式如图 1-2 所示，帧头都由主机发出，在主收从发的场景中，从机产生响应；在主发从收的场景中，主机产生响应。

注意：这里说的主收从发以及主发从收都是针对响应来说的，不是针对帧头，帧头在任何条件下都是主机发出的。

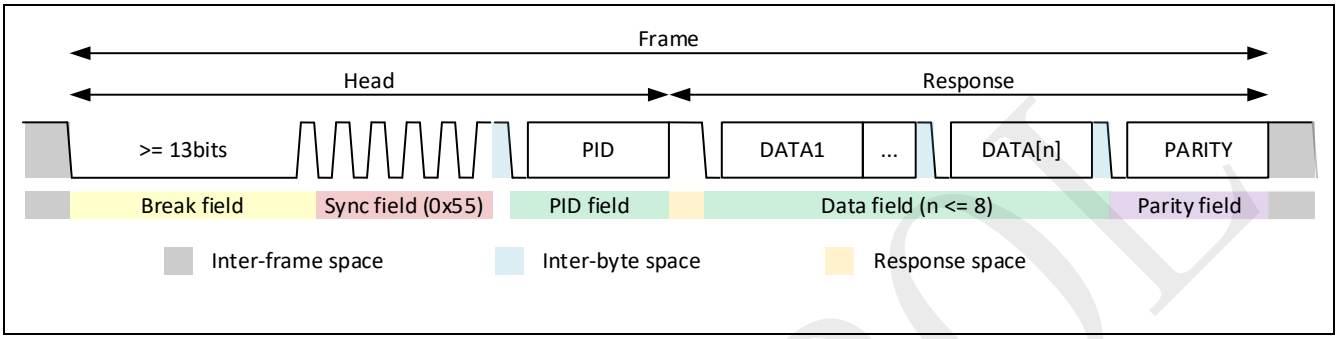
图 1-2：响应的收发



1.2 LIN 帧

LIN 帧的结构如图 1-3 所示，起始信号是超过 13bits 的低电平，起始信号后紧跟通常为 1bit 的高电平，LIN 帧中的其他字段，同步段，PID 段，数据段，校验段均符合标准 UART 协议，起始信号为低电平，字节的 8 个 bit 按从低到高顺序发送，结束信号为高电平。

图 1-3：LIN 帧



同步段固定为 0x55，用于实现自动波特率（从机设定波特率和主机波特率偏差不能超过 ±14%）

PID 段的低 6bit 对应 ID，高 2bit(校验位)通过以下公式算出:

bit[6] = bit[0] ^ bit[1] ^ bit[2] ^ bit[4]

bit[7] = ~(bit[1] ^ bit[3] ^ bit[4] ^ bit[5])

ID 的范围在 0x00 到 0x3F 之间。

数据段的长度是主从机根据 ID 在软件中提前约定好的，最长为 8byte；

校验段有两种计算方法，标准校验只对数据段各字节进行校验，增强校验对数据段各字节以及 PID 进行校验。

1.3 LIN Master 上拉电阻选型

LIN 主机端上拉电阻的选型取决于 LIN 和 GND 之间的负载电容

- 负载电容 10nF，上拉电阻 500ou;
- 负载电容 8nF，上拉电阻 660ou;
- 负载电容 1nF，上拉电阻 1000ou。

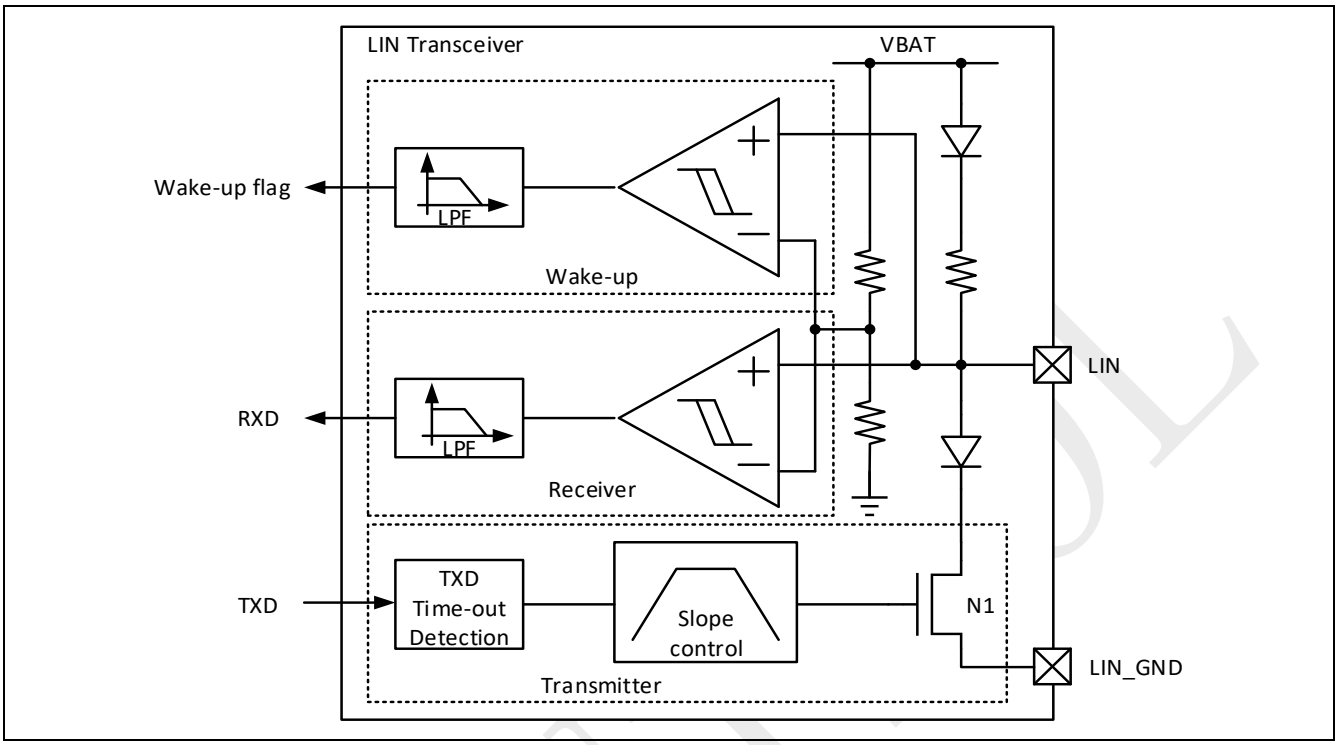
1.4 LIN PHY

LIN PHY 用于产生符合 12V 总线电平的驱动，如图 1-4 所示，Spintrol 的 LIN PHY 有可控制 LIN 信号斜率的寄存器，可控制在 LIN 总线数据传输过程中 VBAT 至 GND 所需的时间，为保证输出信号的方正，LIN PHY 对应 TXSLOPE 需要根据波特率选定：

- 波特率 9600，TXSLOPE 选为 19US;

– 波特率 19200，TXSLOPE 选为 7PUS；

图 1-4：LIN PHY



如图 1-4：LIN PHY 所示，LIN PHY 的 LIN 脚电平通过迟滞比较器链接到 MCU RXD（GPIO26），而 MCU TXD（GPIO25）也链接到 LIN 脚。由此可以看出，若想把 LIN 当做输入 GPIO 使用，此时，输入的电平信号会反应在 GPIO26 上；若当做输出 GPIO 使用，此时，当 TXD 输出高时，LIN 脚电平会被拉成 LIN_GND，而 EXD 输出低时，LIN 脚电平会被上拉电阻拉成 VBAT。

2 LIN 示例

2.1 主发从收

2.1.1 LIN_Slave_RX

如下示例代码的配置步骤为：

- 调用 LIN_Init，其开启了自动波特率（从机设定波特率和主机波特率偏差不能超过±14%），设定了 LIN 帧的超时阀值，设定为从机模式；
- 开启高压模块，完成 LIN PHY 的设定；
- 设定 RX_REQ 的阀值 8byte；
- 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，设定为接收方式并设定接收长度 8byte；
- 当 FIFO 中长度大于 8byte 时，再次触发中断，将 FIFO 中 9byte 数据读出，包含校验位；
- 在 main 中将收到的 9byte 数据打印出来。

Example Code

```
#define             BAUDRATE            19200       /* LIN Rate */
#define             REFID               0x01        /* REFID State */
#define             IDMASK              0x01        /* IDMASK State */
uint8_t             u8Id;                           /* Receive Id */
uint16_t            u16PREDRIID;                    /* PRE-DRIVER mode ID
*/
ErrorStatus         eErrorState;                    /* Function State */
uint16_t            i;                              /* Print Num */
volatile uint8_t    u8Rxd[9];                       /* RX Data */
volatile uint8_t    state               = 0;

/*******************************************************************************
 *
 * @brief      In this case, the LIN slave receive the data from master.
 *
 *        Key_points:
 *            (1)First start the slave code to wait the master code to send
data.
 *

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


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    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");

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

    /* 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_7PUS | LINCTL_STRENGTH_47P2MA | LINCTL_TXEN_ENABLE | 
LINCTL_EN_ENABLE);
    if (eErrorState == ERROR)
    {
        printf("LINCTL_WriteRegisterField FAIL\n");
        return 0;
    }

    UART_SetRxFIFOThreshold(UART1, LIN_RESPONSE_8_BYTE);

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

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

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

    while (1)
    {
        /* Init state */
        state = 0;
```

```
        /* Wait until leave RX_REQ */
        while (state == 0)
        {
        }

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


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_RX);

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

        /* Clear the start signal flag */
        UART_ClearInt(UART1, UART_INT_LIN_ID_MATCH);
    }
    else if (UART_GetIntFlag(UART1, UART_INT_RX_REQ) != 0)
    {
        for (i = 0; i < LIN_RESPONSE_8_BYTE + 1; i++)
        {
            u8Rxd[i] = UART_ReadByte(UART1);
        }

        state = 1;

        /* Clear the start signal flag */
        UART_ClearInt(UART1, UART_INT_RX_REQ);
    }
    /* 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)
    {
```

```
        printf("LIN Tx bit error\n");
        UART_ClearInt(UART1, UART_INT_LIN_BIT_ERROR);
    }
    UART_ClearInt(UART1, UART_INT_ALL);
}
```

2.1.2 LIN_Master_TX

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，使能了校验段的自动发送，设定为主机；
- 开启高压模块，完成 LIN PHY 的设定；
- 设定为发送方式并设定发送长度 8byte，设定校验方式；
- 将 8byte 数据写入 FIFO，并发送帧头；
- 因为开启了校验段的自动发送，硬件会自动将 8byte 数据及校验段发出。

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 */
uint8_t                 u8Txd[8];                                   /* TX Data */
uint8_t                 i;
ErrorStatus             eErrorState;                                /* Function State
*/


/*****************************************************************************
 *
 * @brief       In this case, the LIN master send the data to 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
 *
 *                                       observation point
 *
 *                                    *    *
 *                                   *    *
 *                 *************     *    *      LIN     *************
 *                 *           ****************************           *
 *                 *  Master   *        *                 *   Slave   *
```

```
 *               *           ***************************           *
 *               *************                   GND     *************
 *

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


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    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");

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

    /* 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_7PUS | LINCTL_STRENGTH_47P2MA | LINCTL_TXEN_ENABLE |
LINCTL_EN_ENABLE);
    if (eErrorState == ERROR)
    {
        printf("EPWR_WriteRegisterField FAIL\n");
        return 0;
    }

    UART_EnableInt(UART1, \
```

```
UART_INT_RX_FRAME_ERROR | UART_INT_LIN_BIT_ERROR | UART_INT_RX_TIMEOUT |
UART_INT_LIN_CHECKSUM_ERROR);

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

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

        /* 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 respond data */
        for (i = 0; i < LIN_RESPONSE_8_BYTE; i++)
        {
            u8Txd[i] = rand() & 0xff;
        }

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

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

        /* Wait the TX end */
        while (UART_GetTxFIFOLevel(UART1) != 0)
        {
        }

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

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


void UART1_IRQHandler()
{
    /* Judge RX timeout event and clear */
    if (UART_GetIntFlag(UART1, UART_INT_RX_TIMEOUT) != 0)
    {
        printf("RX_TIMEOUT\n");
        UART_ClearInt(UART1, UART_INT_RX_TIMEOUT);
    }
```

```
    /* Judge RX frame stop bit error event and clear */
    else if (UART_GetIntFlag(UART1, UART_INT_RX_FRAME_ERROR) != 0)
    {
        printf("FRAME_ERROR\n");
        UART_ClearInt(UART1, UART_INT_RX_FRAME_ERROR);
    }

    /* Judge TX frame bit error event and clear */
    else if (UART_GetIntFlag(UART1, UART_INT_LIN_BIT_ERROR) != 0)
    {
        printf("BIT_ERROR\n");
        UART_ClearInt(UART1, UART_INT_LIN_BIT_ERROR);
    }
    /* Judge RX checksum error event and clear */
    else if (UART_GetIntFlag(UART1, UART_INT_LIN_CHECKSUM_ERROR) != 0)
    {
        printf("CHECKSUM_ERROR\n");
        UART_ClearInt(UART1, UART_INT_LIN_CHECKSUM_ERROR);
    }
    /* Clear global flag */
    UART_ClearInt(UART1, UART_INT_GLOBAL);
}
```

2.2 主收从发

2.2.1 LIN_Slave_TX

如下示例代码的配置步骤为：

- 调用 LIN_Init，其开启了自动波特率（从机设定波特率和主机波特率偏差不能超过±14%），设定了 LIN 帧的超时阀值，使能了校验段的自动发送，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；
- 因为开启了校验段的自动发送，硬件会自动将 8byte 数据及校验段发出。

Example Code

```
#define         BAUDRATE        19200           /* 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
*/
```

```
/*****************************************************************************
*****
 *
 * @brief      In this case, the LIN slave send the data to master.
 *
 *        Key_points:
 *                  (1)First start the slave code to wait the master code to
send data.
 *

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


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    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");

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

    /* 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_7PUS | LINCTL_STRENGTH_47P2MA | LINCTL_TXEN_ENABLE |
LINCTL_EN_ENABLE);
```

```
    if (eErrorState == ERROR)
    {
        printf("LINCTL_WriteRegisterField FAIL\n");
        return 0;
    }

    /* 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);

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

    while (1)
    {
    }
}


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] = rand() & 0xff;

        }

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

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

        /* 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)
    {
        printf("LIN Tx bit error\n");
        UART_ClearInt(UART1, UART_INT_LIN_BIT_ERROR);
    }
    UART_ClearInt(UART1, UART_INT_ALL);
}
```

2.2.2 LIN_Master_RX

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 */
volatile uint8_t    state               = 0;


/*****************************************************************************
*****
 *
 * @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
 *
 *                                  observation point
 *
 *                                *    *
 *                               *    *
 *              *************     *    *      LIN     *************
 *              *           ***************************           *
 *              *  Master   *        *                 *   Slave   *
 *              *           ***************************           *
 *              *************                  GND     *************
 *
*****************************************************************************
****/


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    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");

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

    /* 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_7PUS | LINCTL_STRENGTH_47P2MA | LINCTL_TXEN_ENABLE |
LINCTL_EN_ENABLE);
    if (eErrorState == ERROR)
    {
        printf("LINCTL_WriteRegisterField FAIL\n");
        return 0;
    }

    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);

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

    while (1)
    {
        /* Init the state */
        state = 0;

        /* 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*/
        UART_SetBreak(UART1);
        LIN_SetTxID(UART1, u8Id);

        /* Wait until leave RX_REQ */
        while (state == 0)
        {
        }

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

        /* 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);
        }

        state = 1;

        /* Clear the start signal flag */
        UART_ClearInt(UART1, UART_INT_RX_REQ);
    }
    /* 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)
    {
        printf("LIN Tx bit error\n");
        UART_ClearInt(UART1, UART_INT_LIN_BIT_ERROR);
    }
    UART_ClearInt(UART1, UART_INT_ALL);
}
```