APPLICATION NOTE

AT6490: Using LIN on SAMC Microcontroller

SMART ARM-Based Microcontroller

Introduction

The Local Interconnect Network (LIN) Bus was developed to create a standard for low-cost, low-end multiplexed communication in automotive networks.

Implement LIN relatively inexpensively using the standard serial universal asynchronous receiver/transmitter (UART) embedded into most modern low-cost microcontrollers.

This application note will cover the firmware required to initialize and start the SAM C21 SERCOM unit as a LIN Bus master or slave controller and send/receive messages.

Firmware within this document was created using Atmel® Software Framework (ASF), which is an extension to Atmel Studio.

Features

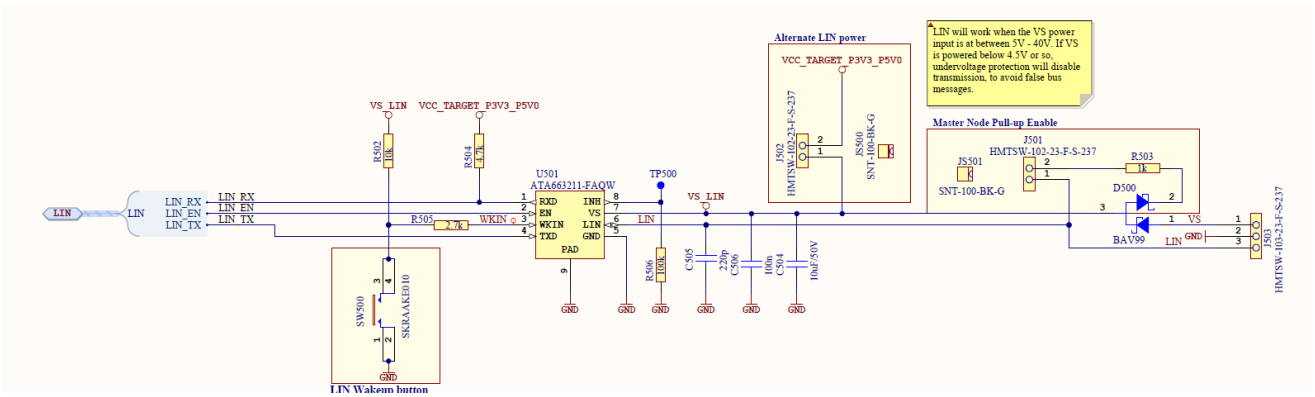
- SAM C21 @ +5V Operation
- Up to six Serial Communications Modules (SERCOM) can be configured for use as a LIN master or slave
- Auto-baud and break character detection

1 LIN Hardware

1.1 Connecting the SAM C21 to the LIN BUS

In LIN, physical signal transmission requires only one conductor (single wire). To maintain radiated electrical emissions within limits, the transmission rate is limited to 20kbit/sec for LIN. Another restriction is the maximum recommended number of 16 nodes. In order to connect the SAM C21 SERCOM's TX and RX pins to the LIN Bus, a LIN Bus transceiver is required. Figure 1-1 shows a typical LIN interface between the SAM C21 and the LIN Bus. This LIN interface is implemented on the Atmel SAM C21 Xplained Pro evaluation kit.

Figure 1-1. LIN Interface Between the SAM C21 and the LIN Bus



LIN Bus transceivers are readily available from a number of semiconductor manufacturers. The Atmel ATA663211 LIN Bus transceiver is preferred.

2 LIN BUS General

2.1 Protocol

The LIN Bus contains one master device and multiple slave devices. The master controls all communications over the bus and is responsible for maintaining regular communications with each slave device. Slave devices listen for and respond to messages from the master. Each slave device is assigned a unique message identifier (ID). When a slave receives a message from the master containing its unique ID, the slave will respond to the host with a message consisting of one to eight bytes of data and an 8-bit checksum.

Messages sent by the master to a slave consist of three fields: break, sync, and message ID. Once received by the slave device, it may take internal action or respond to the master. Decisions as to what action a slave will take to a particular message identifier are determined during development of the system.

2.2 LIN Message Frame

MESSAGE HEADER			MESSAGE RESPONSE	
BREAK	SYNC(0x55)	IDENTIFIER	DATA	CHECKSUM

2.2.1 BREAK

Every LIN frame begins with a break, which serves as a start-of-frame notice to all nodes on the bus.

2.2.2 SYNC

The sync field allows slave devices that perform automatic baud rate detection to measure the period of the baud rate and adjust their internal data rates to synchronize with the bus.

2.2.3 IDENTIFIER (ID)

The ID field consists of a 6-bit message ID and a 2-bit parity field. The ID denotes a specific message address but not the destination.

2.2.4 DATA

Upon receipt and identification of the ID, one slave will respond with one to eight bytes of data and an 8-bit checksum.

2.2.5 CHECKSUM

The checksum is calculated and sent as the last byte of the response. There are two checksum algorithms defined by the LIN Bus specification: Classic and enhanced.

The classic checksum is calculated by summing the data bytes alone and the enhanced checksum is calculated by summing and data bytes along with the protected ID.

2.3 SAM C21 LIN Master

LIN master is available with the following configuration:

- LIN master format (CTRLA.FORM = 0x02)
- Asynchronous Mode (CTRLA.CMODE = 0)
- 16x sample rate using fractional baud rate generation (CTRLA.SAMPR = 1)

Using the LIN command field (CTRLB.LINCMD), the complete header can be automatically transmitted, or software can control transmission of the various header components.

When CTRLB.LINCMD = 0x01, software controls transmission of the LIN header. In this case, software uses the following sequence:

- CTRLB.LINCMD is written with 0x01
- Data register is written to 0x00 – this triggers transmission of the break field by hardware

Note: Writing the data register with any other value will also result in the transmission of the break field by hardware.

- Data register is written with 0x55 – the 0x55 sync value is transmitted
- Data register is written with the identifier – the identifier is transmitted

When CTRLB.LINCMD is written with 0x02, hardware controls the transmission of the LIN header. In this case, software uses the following sequence:

- CTRLB.LINCMD is written with 0x02
- Data register is written with the identifier – this triggers transmission of the complete header by hardware. First the break field is transmitted. Next the sync field is transmitted, and finally the identifier is transmitted.

In LIN master mode, the length of the break field is programmable using the break length field (CTRLC.BRKLEN). When the LIN header command is used (CTRLB.LINCMD = 0x02), the delay between break and sync fields, in addition the delay between sync and ID fields are configurable using the header delay field (CTRLC.HDRDLY). When manual transmission is used (CTRLB.LINCMD = 0x01), software controls the delay between break and sync.

3 Firmware

3.1 LIN SERCOM Configuration

In order to connect the SAM C21 controller TX and RX pins to the LIN Bus transceiver the pin Mux and correct I/O pins need to be set. In addition, an I/O port must be configured as an output to drive the LIN transceiver's enable input.

The following function configures the transceiver enable pin, baud rate and SERCOM TX/RX pins, LIN Bus mode (slave/master) and also sets up the callback functions for transmission, reception, and error handling:

```
static void configure_usart_lin(void)
{

        struct port_config pin_conf;
        port_get_config_defaults(&pin_conf);
        pin_conf.direction   = PORT_PIN_DIR_OUTPUT;
        port_pin_set_config(LIN_EN_PIN, &pin_conf);

        /* Enable LIN Transceiver */
        port_pin_set_output_level(LIN_EN_PIN, 1);

        struct usart_config config_lin;
        usart_get_config_defaults(&config_lin);

        /* LIN frame format*/
        config_lin.lin_node = CONF_LIN_NODE_TYPE;
        config_lin.transfer_mode = USART_TRANSFER_ASYNCHRONOUSLY;
        config_lin.sample_rate = USART_SAMPLE_RATE_16X_FRACTIONAL;

        config_lin.baudrate = 115200;
        config_lin.mux_setting = LIN_USART_SERCOM_MUX_SETTING;
        config_lin.pinmux_pad0 = LIN_USART_SERCOM_PINMUX_PAD0;
        config_lin.pinmux_pad1 = LIN_USART_SERCOM_PINMUX_PAD1;
        config_lin.pinmux_pad2 = LIN_USART_SERCOM_PINMUX_PAD2;
        config_lin.pinmux_pad3 = LIN_USART_SERCOM_PINMUX_PAD3;

        /* Disable receiver and transmitter */
        config_lin.receiver_enable   = false;
        config_lin.transmitter_enable = false;
```

```
    if (CONF_LIN_NODE_TYPE == LIN_SLAVE_NODE) {
        config_lin.lin_slave_enable = true;
    }

    while (usart_init(&lin_instance,
        LIN_USART_MODULE, &config_lin) != STATUS_OK) {
    }

    usart_enable(&lin_instance);

    usart_register_callback(&lin_instance,
        lin_read_callback, USART_CALLBACK_BUFFER_RECEIVED);
    usart_enable_callback(&lin_instance, USART_CALLBACK_BUFFER_RECEIVED);
      usart_register_callback(&lin_instance, lin_read_error_callback, USART_CALLBACK_ERROR);
    usart_enable_callback(&lin_instance, USART_CALLBACK_ERROR);
    system_interrupt_enable_global();
}
```

3.2 LIN Message Reception

The following code enables the reading of messages in both master and slave modes:

```
    if (CONF_LIN_NODE_TYPE == LIN_MASTER_NODE) {
        /*   LIN in Master Mode */
        if (lin_master_transmission_status(&lin_instance)) {
            usart_enable_transceiver(&lin_instance,USART_TRANSCEIVER_TX);
            lin_master_send_cmd(&lin_instance,LIN_MASTER_AUTO_TRANSMIT_CMD);
            usart_write_wait(&lin_instance,LIN_ID_FIELD_VALUE);
            usart_enable_transceiver(&lin_instance,USART_TRANSCEIVER_RX);
            while(1) {
                usart_read_buffer_job(&lin_instance,
                (uint8_t *)rx_buffer, 5);
            }
        }
    } else {
        /*   LIN in Slave Mode */
        usart_enable_transceiver(&lin_instance,USART_TRANSCEIVER_RX);
        while(1) {
            usart_read_buffer_job(&lin_instance,
            (uint8_t *)rx_buffer, 1);
```

```
        }
    }
```

In the ASF project's "conf-lin.h" file the following define is used to configure the firmware as a master or slave device:

```
#define CONF_LIN_NODE_TYPE LIN_MASTER_NODE
```

The project's default is LIN_MASTER_MODE. This should be changed to LIN_SLAVE_MODE to build the project for a slave device.

3.3 LIN Message Transmission

The following code handles the reception of a self-generated LIN message by the master, and sets up the response if running on a LIN slave:

```
static void lin_read_callback(const struct usart_module *const usart_module)
{

    uint8_t i = 0;

    if (CONF_LIN_NODE_TYPE == LIN_MASTER_NODE) {
        for(i = 0; i < LIN_DATA_LEN; i++){
            if(rx_buffer[i] != tx_buffer[i]) {
                /* data error – perform error processing */
                break;
            }
        }
    } else if (CONF_LIN_NODE_TYPE == LIN_SLAVE_NODE) {
        if(rx_buffer[0] == LIN_ID_FIELD_VALUE) {
            usart_enable_transceiver(&lin_instance,USART_TRANSCEIVER_TX);
            printf("Receive ID field from mater: OK \r\n");
            usart_write_buffer_job(&lin_instance,
                (uint8_t *)tx_buffer, LIN_DATA_LEN);
        }
    }
}
```

4 ASF LIN Bus Project

In order to test the project, two SAM C21 Xplained Pro boards will need to be connected together using the LIN Bus header. Shunts for both the Master Mode Pull-up and LIN VCC Power should be installed on the Xplained Pro boards.

Two projects will have to be built; one configured as master mode and the other configured as slave mode. To build the master project, make sure the #define CONF_LIN_NODE_TYPE is set to LIN_MASTER_MODE in “conf_lin.h”. Similarly, set the #define CONF_LIN_NODE_TYPE as LIN_SLAVE_MODE for the slave. Once the projects have been built, program one SAM C21 Xplained Pro with the master mode project, and the other with the slave mode project.

Plug both boards in to the computer’s USB port. Using a terminal program such as TeraTerm, create two terminal windows and set the serial port for eight data bit, one stop bit, no parity, and with a baud rate of 115200. Once connected to the terminal, each board should be reset using the reset pushbutton. The master terminal window will display “LIN Works in Master Mode”, and the slave terminal window will display “LIN Works in Slave Mode”. Pressing the reset pushbutton on the master board results in a display of “Slave response: OK” on the master mode terminal window, and the slave terminal window displaying “Receive ID field from master: OK”. The ASF based code is now ready, and it is possible to start developing LIN master and slave applications.

5 Summary

The firmware used in this application note is taken from the LIN Bus Example application provided by Atmel Studio’s ASF extension. For more information on LIN Bus refer to the latest LIN Bus specification.

6 Revision History

Doc Rev.	Date	Comments
42470A	06/2015	Initial document release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**