NXP Semiconductors
Application Note

Document Number: AN5389
Rev. 0, 02/2017

S12Z MagniV LIN Bootloader

by: Agustin Diaz

Contents

1 Introduction

The LIN protocol is a 1-wire serial protocol and uses the UART format to send the data. However, LIN protocol is a master-slave communication. The master is the one that controls and initiates all communication. It can be connected to different slaves. With the growth of memory size in automotive MCUs the implementation of a bootloader has become more common and as LIN is one of the most widely used automotive communication protocols. It is essential that a bootloader is able to receive a new version of software through LIN.

This application note focus in the implementation of a bootloader that receives the new software to program through LIN. The bootloader that can be found in this application note is based in the application note AN4723 software. It uses the application note's software routines for memory allocation and structure of the program. The main changes remain in the communications layer. Therefore, this document will not provide and extended description of the memory allocation routines as it has been already described in AN4723.

2 Overview

Master node

LIN is a single wire communication that uses the UART format to send data bytes. LIN is a master-slave communication. Every communication is initiated by the master and can be answered by one or multiple slaves. However, slaves cannot communicate information until master has requested them to do it. LIN groups all data to communicate in different frames. Each frame may be linked to one or multiple slaves. Every frame is divided in two parts. The header and the response. The header is always send by the master and contains the ID of that frame. The response field contains the data, it can be either send by the master of the slave.

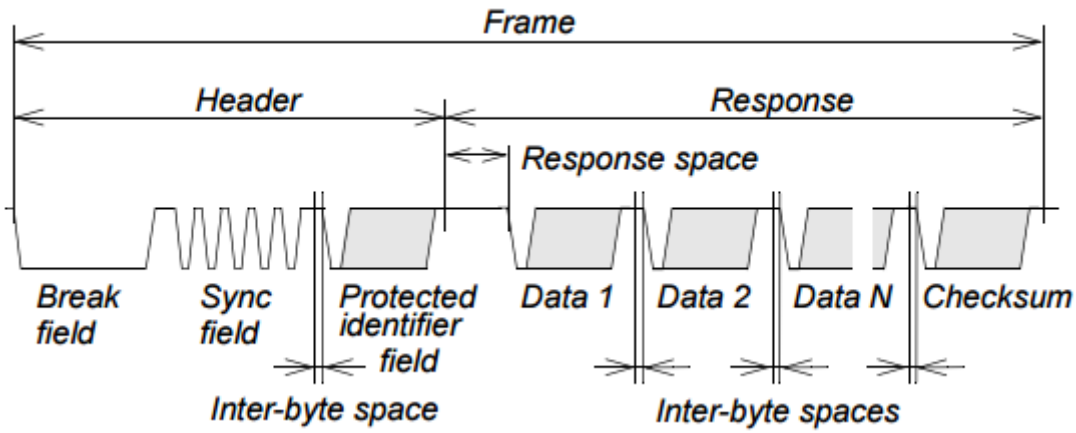

Figure 1. Frame structure

The implementation of the bootloader resides on the slave node. Therefore, the content of this application note will heavily focus on the slave node. However, in order to be able to use the graphical interface provided in AN4723 to load the new software a Master node was implemented. This node translates the serial communication send by the computer to LIN frames and send it to the slave through the physical layer integrated in MagniV devices as can be observed in the following diagram:

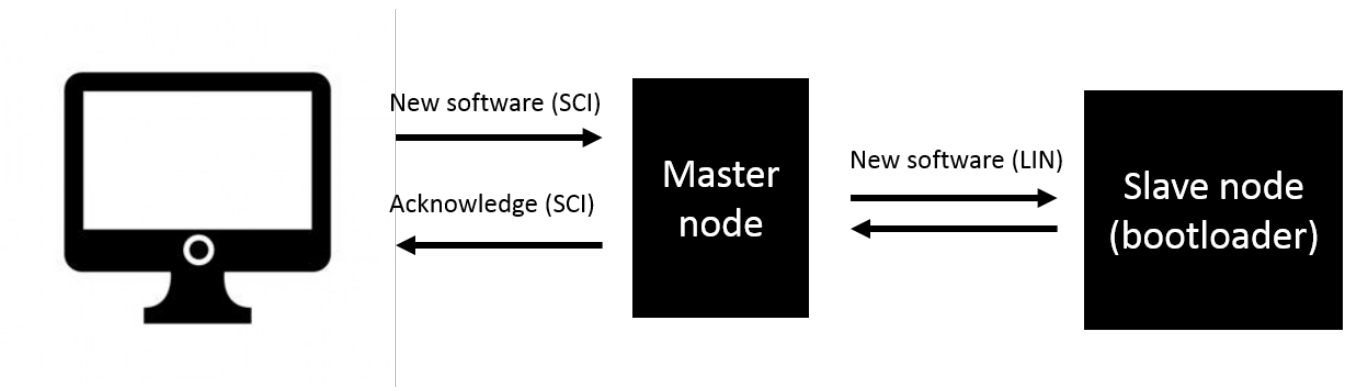

Figure 2. Application diagram

3 Master node

In order to test the functionality of the bootloader and use the graphical interface of AN4723 to load a new firmware a Master node was implemented. This node translates the data send by the computer to LIN and send it through the LIN physical layer available in MagniV devices.

The computer receives an S-record file and send it through SCI to the MCU. The S-record file is send line by line and then waits for the MCU to tell the PC to send another line. In order to be compliant with the LIN standard, the master node receives all the line but divide it in different frames with a maximum length of 8 bytes in the data field. This means that it will sent a maximum of 8 bytes per frame until it has send all the data of the S-record file received from the computer.

As the LIN standard does not fully specified how a bootloader must be implemented. For this application it was chosen to use two unconditional frames for sending and receiving the information. One frame is used to send all the data received by the PC and the other is used to acknowledge the master node that the slave has received and save the data in flash memory.

In order to make it easier for the user to select the ID of each of those frames.

Configuration macros were defined in the code to select the ID of the transmission frame and the reception frame.In the software attached to this application note the transmission and reception IDs frames were 0x31 and 0x32 respectively.

The configuration macros also allows the user to change the baud rate of the LIN transmission. This macro can be found in SCI.h file.

The application note was tested at 9600 baud and 19200 baud as it is the maximum allowable speed defined by the LIN standard.

4 Bootloader node

As it was mentioned above this bootloader is based in the software provided by AN4723. This section will focus on the changes done in the communication layer and how the LIN communication was enabled. As MagniV devices has limited memory, the bootloader should remain as small as possible. Therefore, for this example no LIN stack was used nor interruptions, all LIN communication was enabled by polling any transmission and reception.

As in the reference code of AN4723 the bootloader waits a period of time for any reception issued by the master. If no data is detected and timeout period finishes then the code will try jump to any application that was saved before. However, if the slave receive any frame with the same ID reserved for reception of data. It will start saving the data transmitted by the master. The slave expects to receive the S-record file line by line. Sending acknowledge to the master every time it has finish writing the data into flash. The master should send the S-record file data in frames of 8 bytes. According to the S-record format the first transmitted byte contains the number of data that will be transferred.

The LIN bootloader example contained within this application note have the following configuration parameters in the ***Comms.h*** file:

```
#define BAUD_RATE                       19200
#define LIN_MAX_DATA_BYTES              0x08
#define BOOTLOADER_TRANSMIT_FRAME_ID    0x31
#define BOOTLOADER_RECEIVE_FRAME_ID     0X32
```

- The BAUD_RATE parameter stablish the rate at which data will be send. The bootloader has been tested for 9600 baud and 19200 baud as it is the maximum rate allowed by LIN standard.
- The LIN_MAX_DATA_BYTES parameter stablish the number of data that will be send in one frame, to make the bootloader as fast as possible it has been set to 8 bytes.
- The BOOTLOADER_TRANSMIT_FRAME_ID parameter stablish the ID of the frame that will carry out the data from the master to the slave. This frame is published by the master and the slave only listen and save the data to later store it in flash.
- The BOOTLOADER_RECEIVE_FRAME_ID parameter stablish the ID of the frame that will carry out the response from the slave. The slave respond every time it has finished saving the data in flash and master sends issue the header of a frame with the specified ID.

The following images illustrate the functionality of the bootloader.

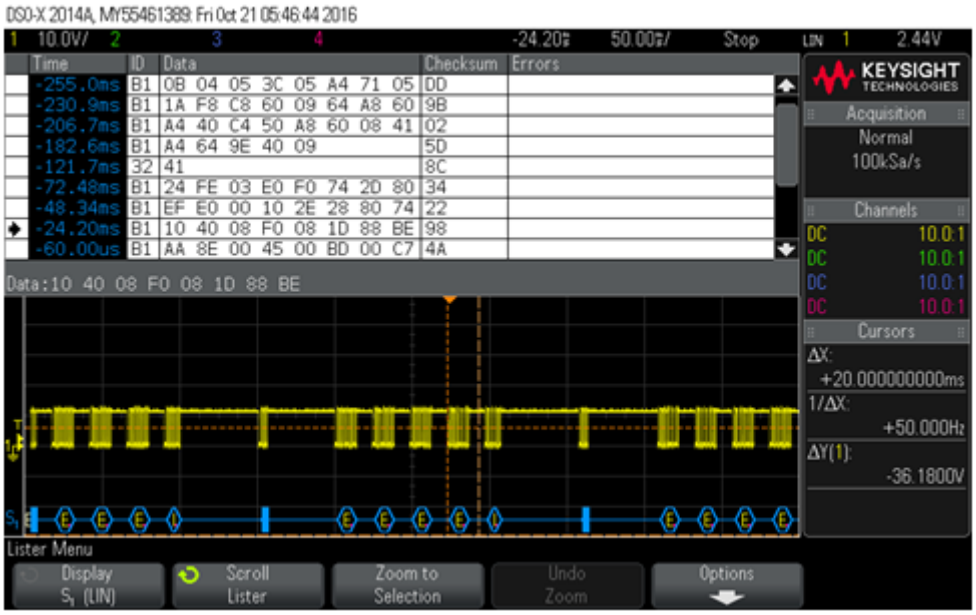

Figure 3. Transmission of data

The Figure 3 illustrate the transmission process. First the frames with PID of 0xB1 (ID of 0x31) contains the new program data. For this example the transmit ID was set to 0x31. The first byte send in this transactions represent the number of bytes to be transmitted as it was explained above and in AN4724. In this case the number of bytes per line is 0x24 (36 decimal). Therefore, to send the 36 bytes of data, 5 different frames are required. After sending all 36 bytes of data the master issue a receive frame with enough time for the slave to finish writing all received data in to flash.

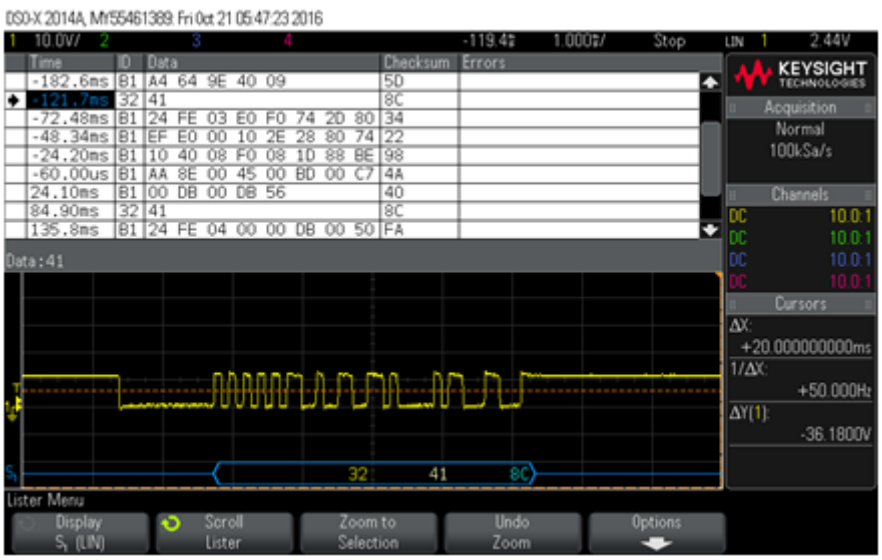

Figure 4. Transmitting acknowledge signal

The Figure 4 illustrate the response of the slave to notify the master that it had finish writing the new program data in to flash and that it is ready to receive another line of the S-record file. This response has a different ID. This ID correspond to the BOOTLOADER_RECEIVE_FRAME_ID for this example it was set to 0x32. As in the UART bootloader of AN4723, an ‘OK’ characters is defined to tell the master that data was correctly received and values were stored in flash. This value is then send by the master to the computer to ask for another line of the S-record file.

5 Application to load

As in the AN4723 software, the user must select the start address where the application that is being loaded starts. The software uses this address in order to jump and execute the application. It is very important that this address is correctly set otherwise the application will not be run correctly and undesired behavior may happen.

In order to check the entry point of the application look at the MAP file and look for the startup function. If the application to load was created using CodeWarrior the map file can be found in the FLASH folder. Once open look for the STARTUP SECTION. It should look like this one:

```
*****************************************************************************************************
STARTUP SECTION
-----------------------------------------------------------------------------------------------------
Entry point: 0xFE002E (_Startup)
_startupData is allocated at 0xFE003E and uses 20 Bytes
extern struct _tagStartup {
  unsigned nofZeroOut     1
  _Range   pZeroOut       0x1100     33
  _Copy    *toCopyDownBeg 0xFE0392
  int      nofInitBodies  0
  _Cpp     *initBodies    0xFE0059
  int      nofFiniBodies  0
  _Cpp     *finiBodies    0xFE0059
} _startupData;

*****************************************************************************************************
```

Figure 5. Startup section in MAP file

The entry point for this specific application is 0xFE002E, this should be the specified address in the bootloader.

6 Attachments

This application note comes with three software projects:Master node:

- This project is implemented in a S12ZVL128 evaluation boardBootloader node
- (S12ZVL128)Bootloader node
- (S12VML128)

The application note also includes one S-record file example for each of the bootloader node.

7 Reference

- Package, LIN Specification. "Revision 2.2A." LIN consortium (2010).
- Inzunza, Arturo AN4723 : S12Z MagniV Bootloader. Rev 0. July, 2013.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number AN5389
Revision 0, 02/2017