Freescale Semiconductor, Inc.
Application Note

Document Number: AN5201
Rev. 0, 09/2015

Integrating the LIN driver with BLDC sensorless motor controller

In the S12ZVM128 device

By: Agustin Diaz

Contents

1. Introduction

This application note provides a guideline to integrate the LIN driver with the BLDC sensorless motor controller software provided with the S12ZVM128EVB. This application note does not focus on explaining the LIN protocol or Freescale's driver configuration. For more information regarding this topic please visit www.freescale.com and search out for LIN application note AN5122.

2. The LIN Physical Layer

The LIN transmitter is a low-side MOSFET with current limitation and overcurrent transmitter shutdown. A selectable internal pull-up resistor with a serial diode structure is integrated, so no external pull-up components are required for the application in a slave node. The fall time from recessive to dominant and the rise time from dominant to recessive is selectable and controlled to guarantee communication quality and reduce EMC emissions. The symmetry between both slopes is guaranteed.

The LIN (Local Interconnect Network) bus pin provides a physical layer for single-wire communication in automotive applications. The LIN Physical Layer is designed to meet the LIN Physical Layer 2.2 specification from LIN consortium.

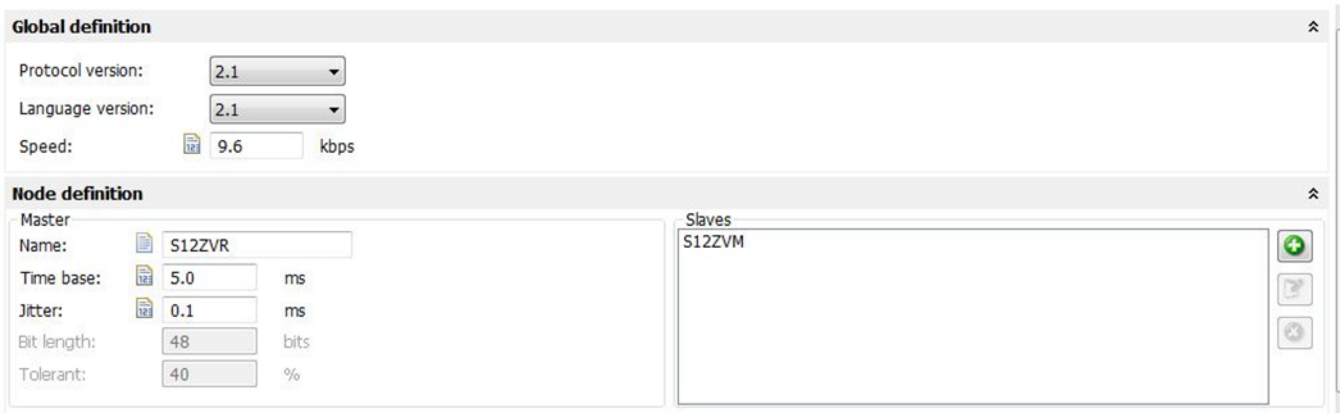
The following is a summary of the most notable LIN Physical Layer features:

- Compliant with LIN Physical Layer 2.2 specification.
- Compliant with the SAE J2602-2 LIN standard.
- Standby mode with glitch-filtered wake-up.
- Slew rate selection optimized for the baud rates: 10.4 kbit/s, 20 kbit/s and Fast Mode (up to
- 250 kbit/s).
- Switchable 34 kOhms/330 k Ohms pullup resistors.
- Current limitation for LIN Bus pin falling edge.
- Overcurrent protection.
- LIN TxD-dominant timeout feature monitoring the LPTxD signal.
- Automatic transmitter shutdown in case of an overcurrent or TxD-dominant timeout.
- Fulfills the OEM “Hardware Requirements for LIN (CAN and FlexRay) Interfaces in Automotive Applications” v1.3

3. LIN Configuration

In this application note a S12VR32 was selected as master and the S12ZVML128 functions as the slave. The parameters configured in the LDF and NPF files are shown in the following images. For more information about the definition of the parameters shown please review the LIN application note available at www.freescale.com or the LIN specification document.

The names chosen in this example are not absolute and can be defined by the user.



Global definition

Protocol version:	2.1	
Language version:	2.1	
Speed:	9.6	kbps

Node definition

Master

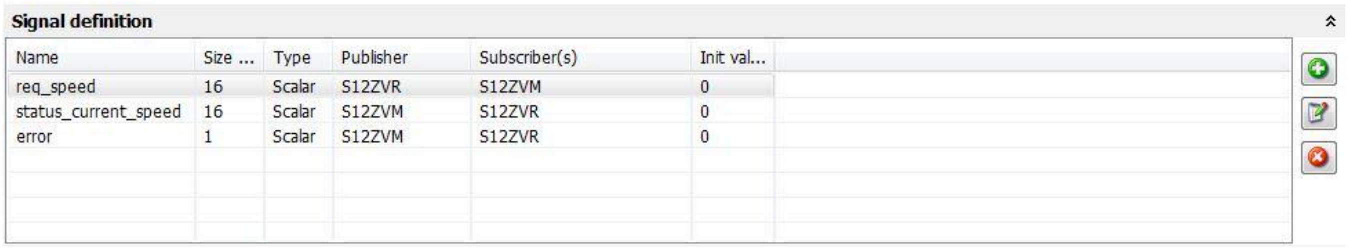
Name:	S12ZVR	
Time base:	5.0	ms
Jitter:	0.1	ms
Bit length:	48	bits
Tolerant:	40	%

Slaves

S12ZVM

Figure 1. LDF Global definition and Node definition

In the example provided in this application note 3 signals were defined. One to send the speed to the slave, one to read the current speed of the slave and one for the error handling.

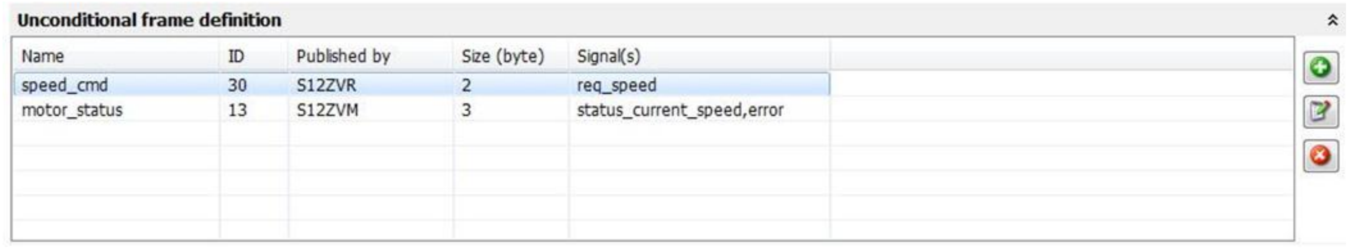


Signal definition

Name	Size ...	Type	Publisher	Subscriber(s)	Init val...
req_speed	16	Scalar	S12ZVR	S12ZVM	0
status_current_speed	16	Scalar	S12ZVM	S12ZVR	0
error	1	Scalar	S12ZVM	S12ZVR	0

Figure 2. LDF Signal definition

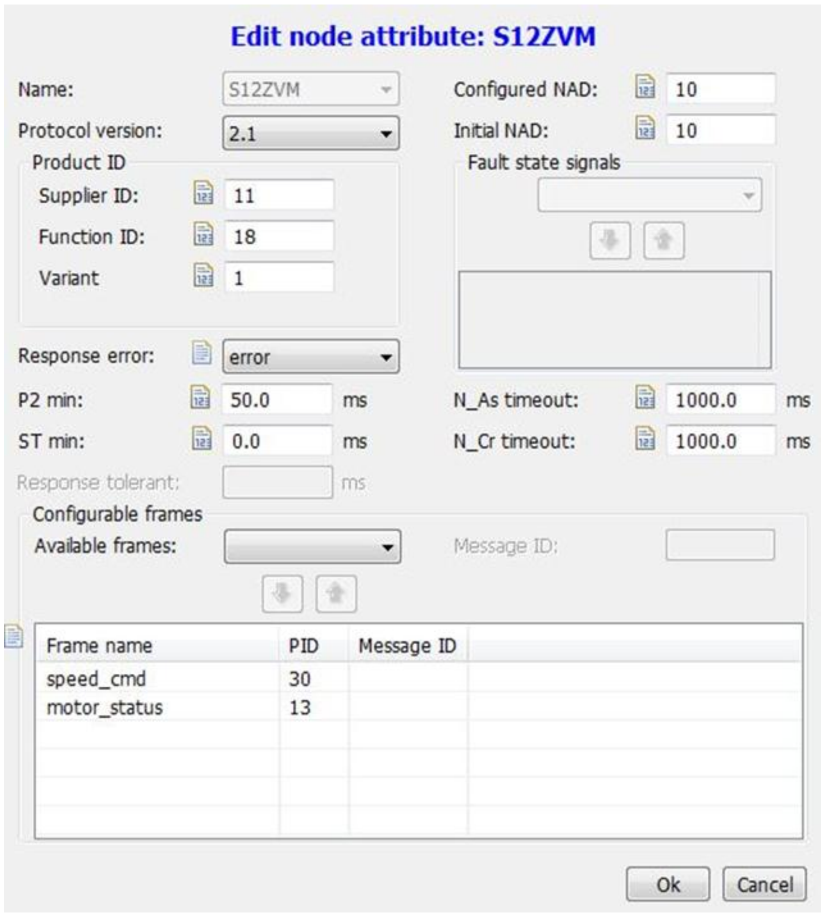
Two unconditional frames where defined. One to carry the required speed and the other one to carry the current speed of the slave and the error signal.



Unconditional frame definition

Name	ID	Published by	Size (byte)	Signal(s)
speed_cmd	30	S12ZVR	2	req_speed
motor_status	13	S12ZVM	3	status_current_speed,error

Figure 3. LDF Frames definition



Edit node attribute: S12ZVM

Name:	S12ZVM	Configured NAD:	10
Protocol version:	2.1	Initial NAD:	10

Product ID

Supplier ID:	11
Function ID:	18
Variant	1

Fault state signals

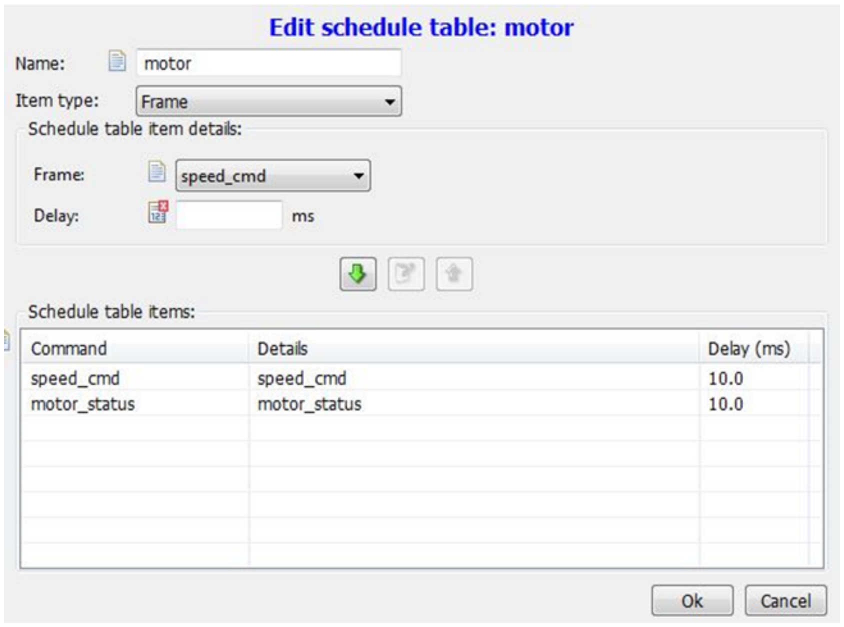
Response error:	error		
P2 min:	50.0 ms	N_As timeout:	1000.0 ms
ST min:	0.0 ms	N_Cr timeout:	1000.0 ms
Response tolerant:	ms		

Configurable frames

Available frames: | Message ID:

Frame name	PID	Message ID
speed_cmd	30	
motor_status	13	

Figure 4. LDF Node attribute configuration



Edit schedule table: motor

Name:	motor
Item type:	Frame

Schedule table item details:

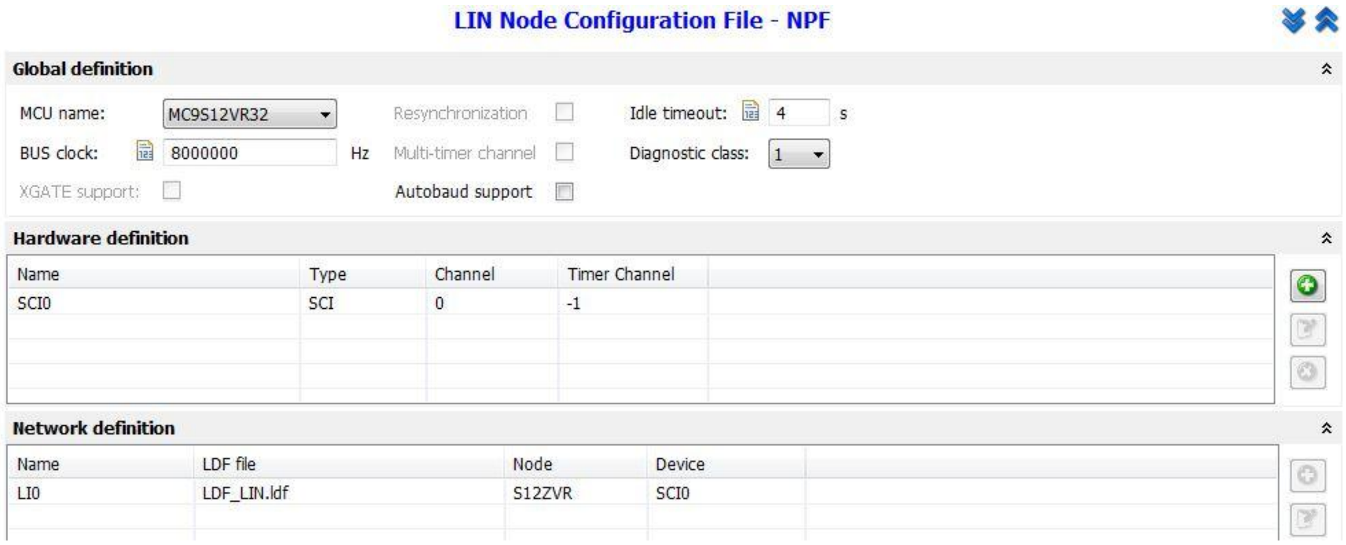
Frame:	speed_cmd
Delay:	ms

Schedule table items:

Command	Details	Delay (ms)
speed_cmd	speed_cmd	10.0
motor_status	motor_status	10.0

Figure 5. LDF Schedule configuration

The configuration for the nodes are the following:



LIN Node Configuration File - NPF

Global definition

MCU name:	MC9S12VR32	Resynchronization	☐	Idle timeout:	4 s
BUS clock:	8000000 Hz	Multi-timer channel	☐	Diagnostic class:	1
XGATE support:	☐	Autobaud support	☐		

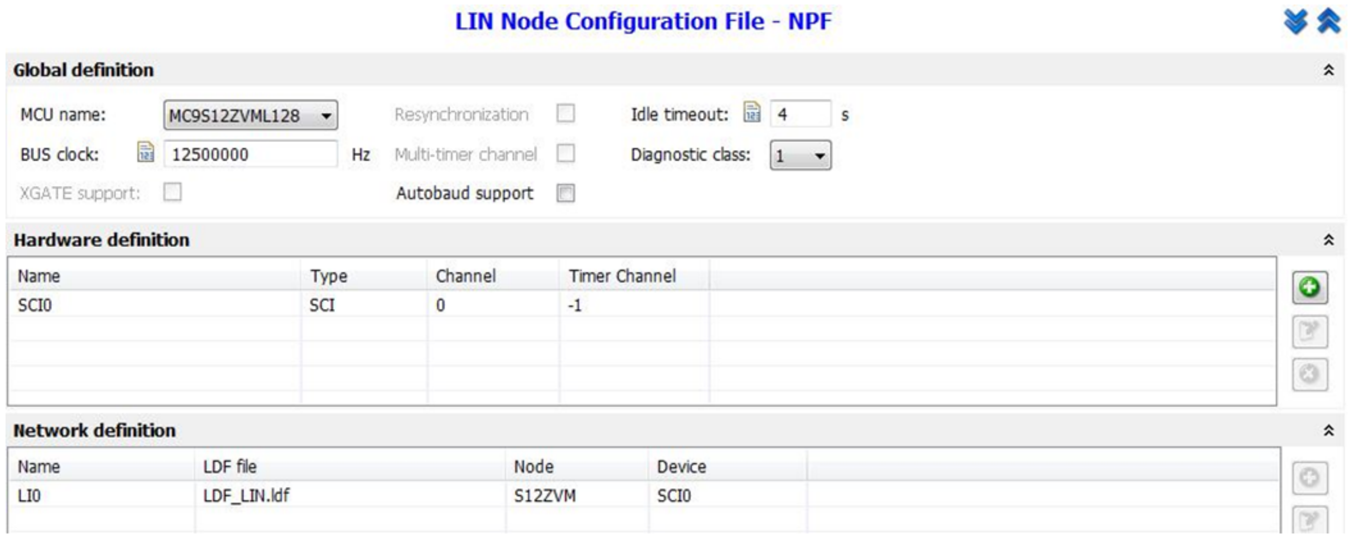
Hardware definition

Name	Type	Channel	Timer Channel
SCI0	SCI	0	-1

Network definition

Name	LDF file	Node	Device
LI0	LDF_LIN.ldf	S12ZVR	SCI0

Figure 6. NPF Master node configuration



LIN Node Configuration File - NPF

Global definition

MCU name:	MC9S12ZVML128	Resynchronization	☐	Idle timeout:	4 s
BUS clock:	12500000 Hz	Multi-timer channel	☐	Diagnostic class:	1
XGATE support:	☐	Autobaud support	☐		

Hardware definition

Name	Type	Channel	Timer Channel
SCI0	SCI	0	-1

Network definition

Name	LDF file	Node	Device
LI0	LDF_LIN.ldf	S12ZVM	SCI0

Figure 7. NPF Slave node configuration

NOTE

It is important that the BUS clock on each device matches the BUS clock at which each one is configured.

4. Example Overview

In the example included in this application note LIN communication was implemented between a S12ZVR32EVB which will be the master and a S12ZVM128EVB which will be the slave.

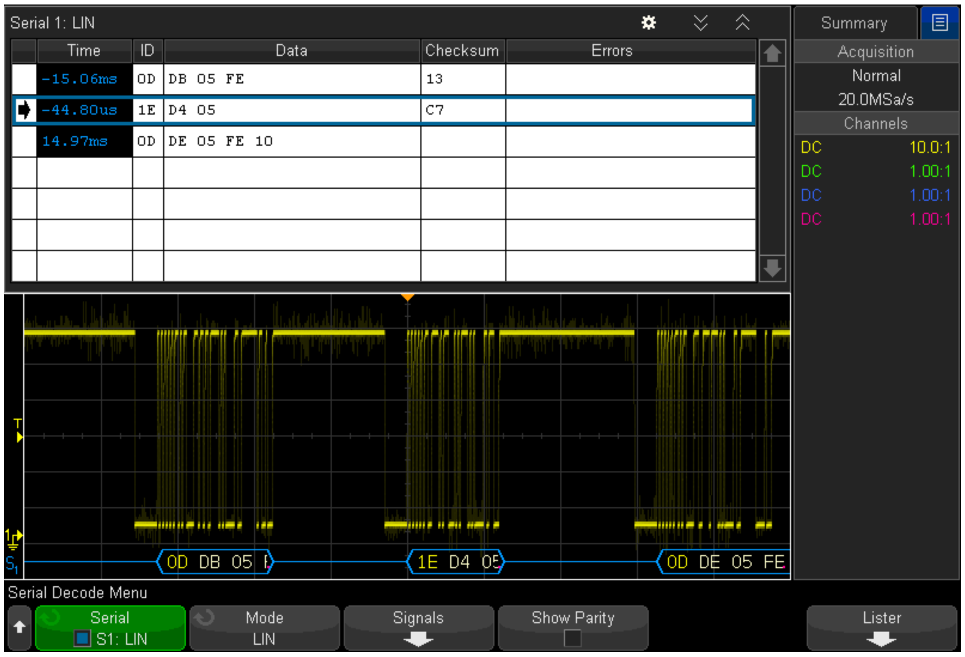
The example illustrates the implementation of LIN in the BLDC motor control algorithm included in the S12ZVM128EVB documentation. Through the LIN bus, the master sends the speed at which the motor should be running. The slave replies back with the current speed of the motor. There is also an error signal in case a problem is presented in the transmission.

The master S12VR32EVB sends three different speed references to the slave with a three position switch. The speed reference is represented as a 16 bit variable that the control algorithm transform into RPM. The first one is the minimum supported by the algorithm (1490 in the 16 bit variable). The second one is another speed selected by the user, for this example a 3000 value was selected for the 16 bit variable. The third one makes the motor slowly accelerate from the minimum velocity to a selected velocity, when it reaches the established velocity the motor slowly decelerate to minimum speed and starts to accelerate again. The speed reference received by the slave is stored in a 16 bit variable called speed. The current speed of the motor is stored in a variable called actual Speed and then send to the master.

The slave receives the speed sent by the master, implements it on the motor and sends the speed at which the motor is running to the master.

Images of the LIN transmission and the behavior of the motor are shown below. The data selected by the cursor is the required velocity and the one with ID 0D is the current velocity.

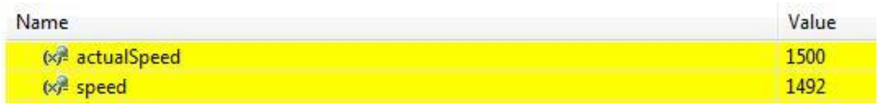
4.1. Minimum Speed Reference


Serial 1: LIN

Time	ID	Data	Checksum	Errors
-15.06ms	0D	DB 05 FE	13	
-44.80us	1E	D4 05	C7	
14.97ms	0D	DE 05 FE 10		

Figure 8. LIN transmission of minimum speed reference

In the following image the used variables in the slave are shown. Small variation in velocities may be presented due to acceptable errors in the control algorithm. In this example the master request that the 16 bit variable sets to a value of 1492. The algorithm control sets it to 1500. Errors of +/- 10 are acceptable.



Name	Value
actualSpeed	1500
speed	1492

Figure 9. CW variables of minimum speed reference

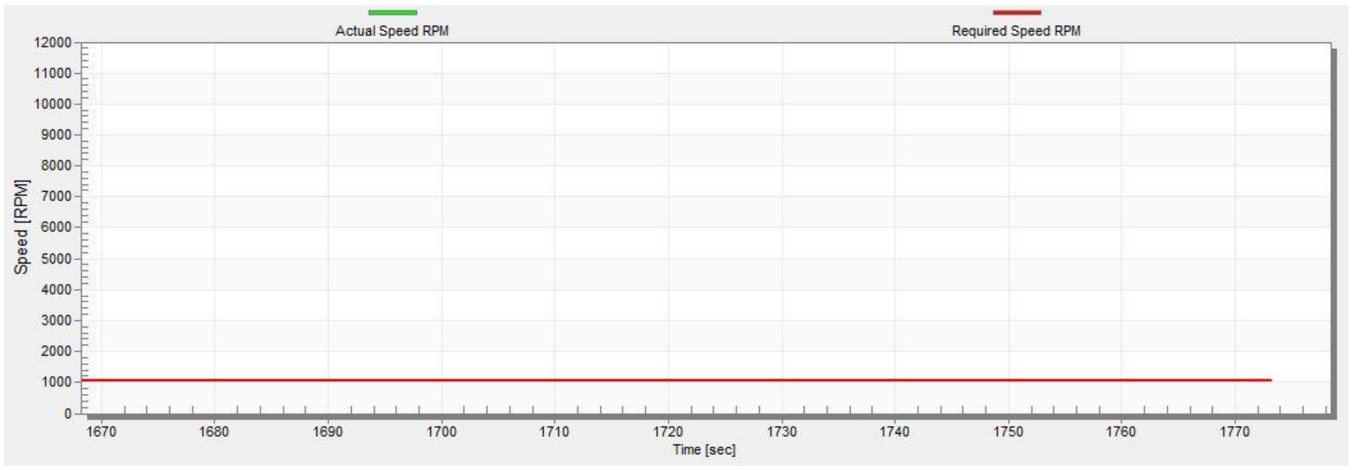
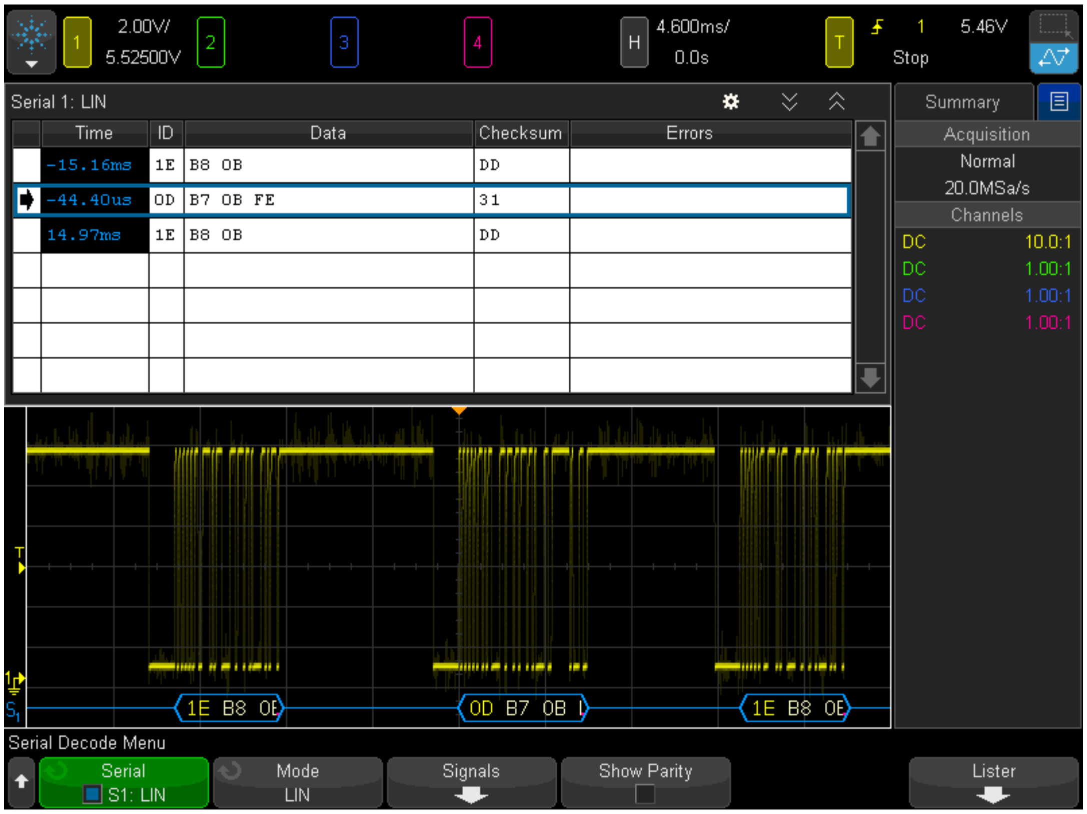


Figure 10. FreeMaster of minimum speed reference

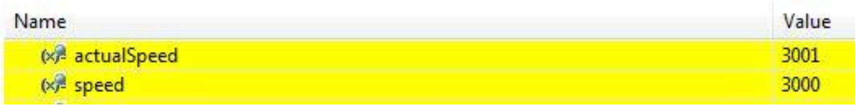
4.2. User Defined Speed Reference



Serial 1: LIN

	Time	ID	Data	Checksum	Errors
	-15.16ms	1E	B8 0B	DD	
➧	-44.40us	0D	B7 0B FE	31	
	14.97ms	1E	B8 0B	DD	

Figure 11. LIN transmission of established speed reference



Name	Value
actualSpeed	3001
speed	3000

Figure 12. CW variables of established speed reference

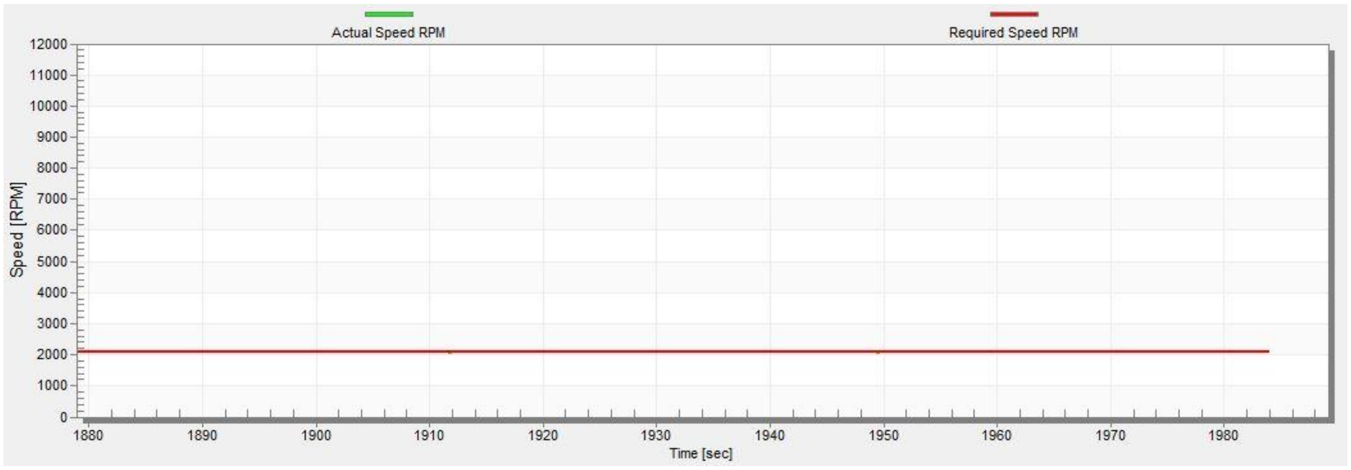


Figure 13. FreeMaster of established speed reference

4.3. Up and Down Speed Reference

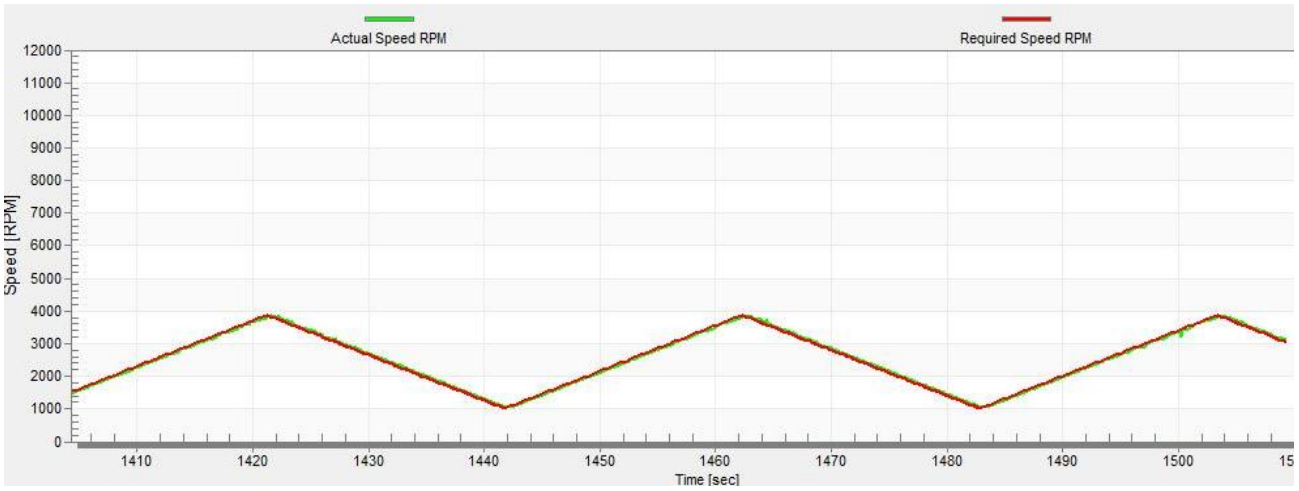


Figure 14. FreeMaster of accelerate and decelerate sequence

5. Important Considerations

Due to the BLDC motor control algorithm is extremely time constrained to integrate both programs special considerations should be taken into account.

- Interruption definition
- Vector interruption allocation in memory
- Resources distribution

As the control algorithm and the LIN driver share interrupts vectors trying to merge them results in compiling errors. In the example provided to handle that problem all the interruptions were handled with the interruption vectors described in the mc9s12zvml128.h file. Also in the S12zvm.prm located in src/S12ZVM_system/startup_CW all the references to the vector.c file or to any interrupt vector should be either deleted or commented. This need to be done because the control algorithm locate the interrupt vectors in certain memory locations, which overlap with the vectors defined in the LIN driver and cause errors. The repeated interrupt vectors defined in the LIN driver were the SCI0 and the timer0 channel 2. For better understanding check the example provided in this application note.

Another consideration is the distribution of resources. In order for the control algorithm to work properly certain tasks should be realized immediately after requested. For example, the zero-cross and commutation timing calculations are essential for the motor to work. The LIN driver adds more tasks to be handled by the MCU, which can cause the motor to stop working correctly. In order to solve this problems the timeout interruption of the LIN driver should be disabled. In the lin_lld_timerv.c file located at src/LIN/LIN_driver/bsp/SCI the configuration on the timer for the S12ZVML128 should be located and interruption for the channel 2 disabled (TIM0TIE_C2I=0).

Also in the lin_lld_sci.c file located in src/LIN/LIN_driver/bsp/SCI the lin_lld_sci_isr function should be located. This function defines what to do when the LIN driver SCI interruption is detected. In order to allow the control algorithm to handle its own interruptions an *asm(CLI)* instruction should wrote down after clearing the interruption flag. This instruction enable all the interruptions requests which allow the motor to keep in check the zero crosses in case one occurred while the driver is handling the SCI interruption.

6. Attachments

This application note includes the example software explained in previous sections. The software contains both, the program of the S12ZVM as the slave and the program of the S12ZVR which function as the master.

The .ldf and .npf archives used to configure the network are located in the S12ZVM project.

The project of the S12ZVM contains the FreeMaster file used to monitor the speed of the motor.

7. References

- http://cache.freescale.com/files/microcontrollers/doc/app_note/AN5122.pdf
- http://www.freescale.com/webapp/sps/site/overview.jsp?code=IFATOLIN

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5201
Rev. 0
09/2015