Freescale Semiconductor
Application Note

AN2884
Rev. 1, 12/2004

LIN 2.0 Door Lock Slave

Based on the MC68HC908GR16 MCU and the LIN 2.0 Communication Protocol

by: Jiri Kuhn
8/16-bit Systems Engineering
Roznov pod Radhostem, Czech Republic

Introduction

Requirements for increased functionality inevitably lead to an increase in wiring requirements. Apart from the amount of copper required, the usable free space can be a limiting factor, impacting on development, especially within the car. Car doors are one example where this limitation applies. It is easy to imagine the amount of wiring needed within the car door if central door locking, power windows, power mirrors, and an audio system are fitted. All this requires extra wiring, which uses much of the strictly limited space.

One way to handle this situation is to use a bus oriented solution. Three main automotive industry qualified standards are in use today: high performance FlexRay networks; the widely used CAN; and the low cost LIN (used mainly at a sensor/actuator level). The door lock slave application described here, and the mirror unit implementation described in Reference [6]) are examples of LIN based solutions.

System Outline, VCT LTP 2.0

As AN2726 deals with implementing LIN 2.0 connectivity on Freescale MCUs (see Reference [3]), only the key connectivity related features are described here. It is recommended that the reader first study AN2726 (see Reference [3]).

freescale™ semiconductor

Cluster Topology, Node Names

The door lock application LIN cluster consists of one master node, four door lock units, and one trunk lock unit, as shown in Figure 1.

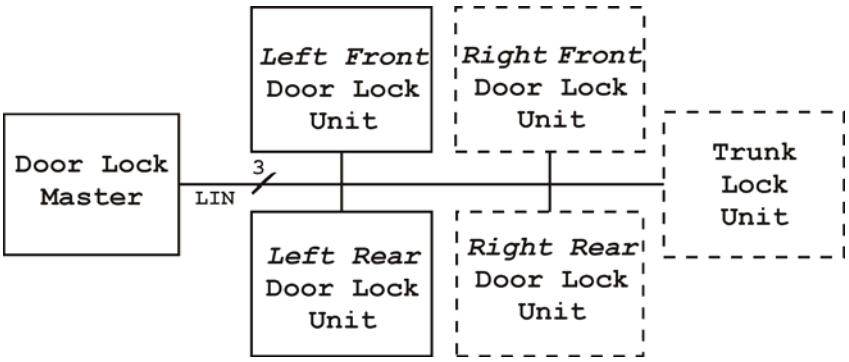

Figure 1. Door Lock Slave Application Cluster Topology

The door lock master node application is designed to operate with both left door units (front and rear). The remaining nodes (dashed blocks in Figure 1) are not implemented in the master node application; however, they are included in the cluster topology, together with relevant frame names.

The node naming convention and the node configured NAD (used as a parameter for the LIN diagnostic layer) can be seen in Table 1:

Table 1. Door Lock Slave Application Node Naming Convention

Node description	Master node	Left front door lock unit	Right front door lock unit	Left rear door lock unit	Right rear door lock unit	Trunk lock unit
Node name	lock_master	lock_f_l[1]	lock_f_r	lock_r_l[1]	lock_r_r	lock_trunk
Configured NAD	N/A	0x10	0x11	0x12	0x13	0x14

NOTES:
1. l is lowercase L

Message Strategy

The message strategy of the door lock slave application cluster can be seen in Table 2.

Table 2. Door Lock Slave Application Messaging Strategy

Node Name for Signal Provider	ID [0..5]	LIN Frame ID	Frame Name	Frame Description	Signal Name	Signal Description	Signal Initial Value	Which nodes are reading this response data? LOCK_MASTER	LOCK_F_L	LOCK_F_R	LOCK_R_L	LOCK_R_R	LOCK_TRUNK	Comments
<published_by>	<frame_id> (0 to 63)		<frame_name>	-	<signal_name>	-	<init_value>	<subscribed_by>						
LOCK_MASTER	0x10	0x4000	LOCK_DOOR_CMD	Lock/Unlock door servo	LOCK_DOOR	Lock/Unlock door	1		X	X	X	X		Initial value is set to 1 (Lock) to prevent unlock after the battery disconnection.
LOCK_MASTER	0x11	0x4001	LOCK_TRUNK_CMD	Lock/Unlock trunk servo	LOCK_TRUNK	Lock/Unlock trunk	1						X	Initial value is set to 1 (Lock) to prevent unlock after the battery disconnection.
LOCK_F_L	0x12	0x4002	LOCK_DOOR_FL_STATUS	Left front door lock status	LOCK_FL_ERROR	Error occurred on left front door	0	X						Set to 1 if error occurred from last lock move.
					LOCK_FL_SWITCH	Left front door switch status	0	X						Initial value set to 0 (Lock) to prevent unlock after the battery disconnection.
					LOCK_FL_SYS_ERROR	Left front door system status	0	X						Set to 1 if error occurred on the LIN layer.
LOCK_F_R	0x13	0x4003	LOCK_DOOR_FR_STATUS	Right front door lock status	LOCK_FR_ERROR	Error occurred on right front door	0	X						Set to 1 if error occurred from last lock move.
					LOCK_FR_SWITCH	Right front door switch status	0	X						Initial value set to 0 (Lock) to prevent unlock after the battery disconnection.
					LOCK_FR_SYS_ERROR	Right front door system status	0	X						Set to 1 if error occurred on the LIN layer.
LOCK_R_L	0x14	0x4004	LOCK_DOOR_RL_STATUS	Left rear door lock status	LOCK_RL_ERROR	Error occurred on left rear door	0	X						Set to 1 if error occurred from last lock move.
					LOCK_RL_SYS_ERROR	Left rear door system status	0	X						Set to 1 if error occurred on the LIN layer.
LOCK_R_R	0x15	0x4005	LOCK_DOOR_RR_STATUS	Right rear door lock status	LOCK_RR_ERROR	Error occurred on right rear door	0	X						Set to 1 if error occurred from last lock move.
					LOCK_RR_SYS_ERROR	Right rear door system status	0	X						Set to 1 if error occurred on the LIN layer.
LOCK_TRUNK	0x16	0x4006	LOCK_TRUNK_STATUS	Trunk status	LOCK_TRUNK_ERROR	Error occurred on trunk lock	0	X						Set to 1 if error occurred from last lock move.
					LOCK_TRUNK_SYS_ERROR	Trunk system status	0	X						Set to 1 if error occurred on the LIN layer.
LOCK_MASTER	0x3C	N/A	MasterReq	LIN Master Request Command			0x00	X	X	X	X	X	X	Data 0x00-0x7F reserved by LIN spec. 0x00= Sleep
-	0x3D	N/A	SlaveResp	LIN Slave Response Command	-	-	-	X						
-	0x3E	N/A	<reserved>	Reserved for user-defined extended frame	-	-	-							
-	0x3F	N/A	<LIN reserved>	Reserved for future extended LIN version	-	-	-							

In addition to the diagnostic frames, the master node sends two frames, *lock_door_cmd* and *lock_trunk_cmd*, with the lock/unlock command for the door and trunk units. The slave node response

varies according to the node's functionality. The rear door units and the trunk provide the master node with the system error bit information[1], together with error feedback, used to inform the master node that an error occurred when the lock last moved. The front door units, in addition to this information, send the door lock switch status to the master node; this is used by the master node application to generate the lock/unlock commands.

Schedule Tables

There are three schedule tables defined in the application.

The first is the *sch_conflict_resolving* schedule table, where the frame IDs are assigned to the nodes. An example of changing the NAD is shown (from value *0x09* to *0x10* for a node with supplier ID *0x0004* and function ID *0x0010*).

The second is the *normal_mode* schedule table. This is used in normal running mode to provide the master node with the status of the front door units switches and the door lock units error status, and to supply the slave nodes with the lock/unlock commands. Because the lock/unlock command is distributed throughout the *normal_mode* schedule table consecutively, it is the responsibility of the slave node application to move the lock unit only when the lock/unlock command is changed.

The third is the *low_power_mode* schedule table, which is designed to decrease the system power consumption. More details about the application low power mode can be found in Low Power Mode on page 11 section of this document.

1. The system error bit is set whenever a frame received or transmitted by the node contains an error in the response field.

Hardware

The slave node hardware concept is shown in Figure 2.

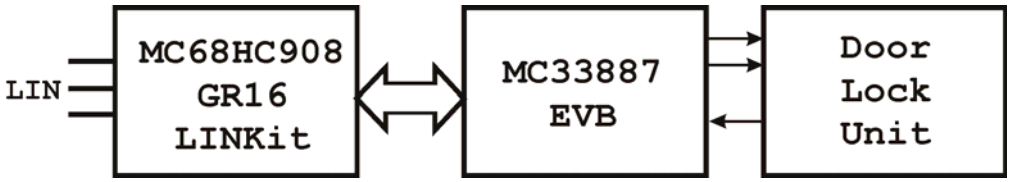

Figure 2. Slave Nodes Hardware Concept

The slave node consists of three main parts:

- MC68HC908GR60A LINKit board fitted with the MC68HC908GR16 MCU[1];
- MC33887 evaluation board, representing the high-power side of the door lock unit;
- The door lock unit itself.

MC33887

The MC33887 (see Reference [5]) is an example of the wide portfolio of Freescale monolithic H-bridge power IC. The key features of the MC33887 H-bridge are as follows.

- Capability to control inductive loads with continuous DC load currents up to 5.0 A
- Load current feedback
- Over current and over temperature protection
- Output short circuit protection and shutdown
- Active current limiting via internal constant OFF-time PWM
- Fault status reporting
- Sleep mode with current draw < 50μA

The MC33887 simplified internal block diagram can be seen in Figure 3:

1. As the MC68HC908GR60A and the MC68HC908GR16 MCUs are pin compatible, there is no need to change the LINKits board (other than replacing the MCU).

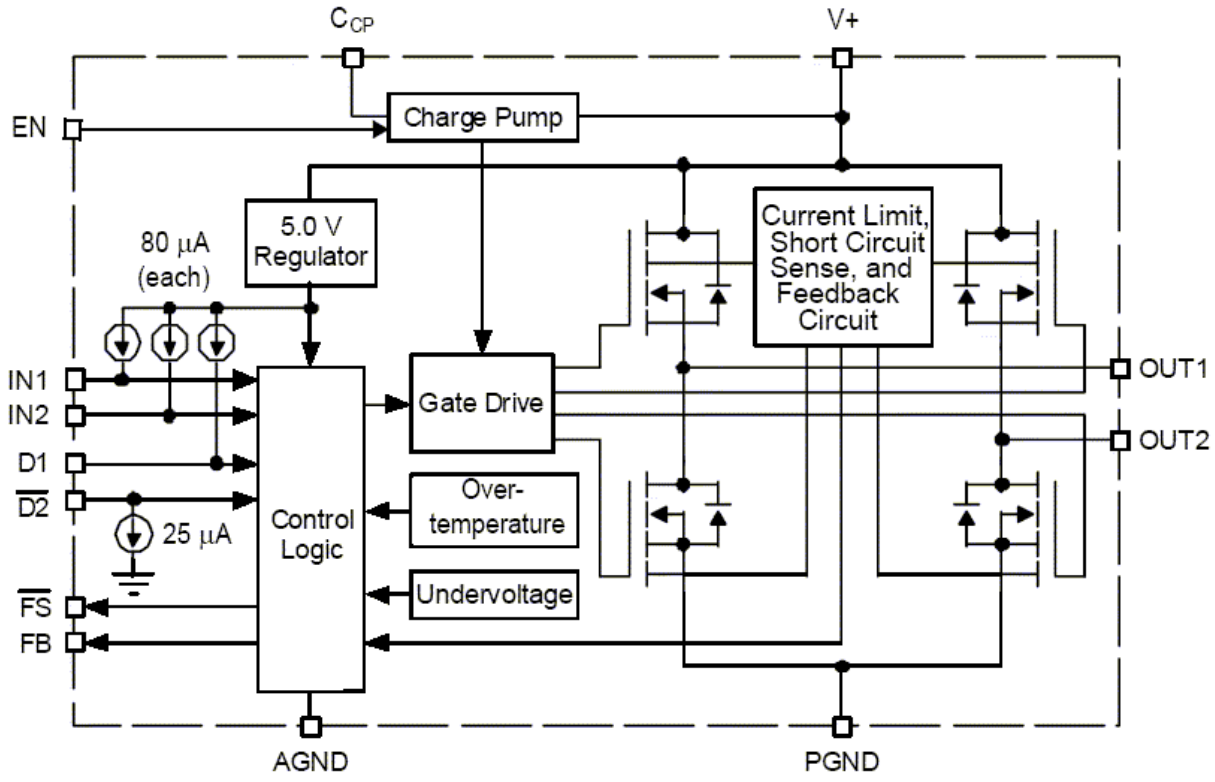

Figure 3: MC33887 Simplified Internal Block Diagram

The MC33887 is connected to the MCU via four input pins. Two of them (*IN1* and *IN2*) are used to control the MC33887 power output pins (*OUT1* and *OUT2*), and two (*D1* and $\overline{D2}$) are used to introduce the output power pins to tristate disable. The MC33887 load current feedback is connected to the MCU A/D converter channel and is used to detect the backstop position of the door lock unit.

Door Lock Units

To develop the application, door lock units based on a DC brush motor are used. However, the application can be easily modified to control solenoid door lock units.

In general terms, there are two kinds of door lock units:

- Rear door units, fitted with a DC brush motor (or any kind of actuator);
- Front door units, fitted with a DC brush motor and a switch detecting the lock movement. Information from the switch is used to decide if the change of state of the door locks was requested (e.g. after the car key turns in the door lock). A request to lock/unlock the car door can be replaced by a command from the remote key receiver or immobilizer.

The construction of the door lock unit consists of a mechanical changer, converting the rotation of the DC brush motor to the sliding movement of the connection rod with backstops. When the mechanical changer hits the backstop, the DC motor current increases. This situation is used to detect the end stop position of the door lock, to stop driving the DC motor. It is recommended to measure the steady state current of

the DC brush motor winding (after the current transient phenomenon caused by the DC brush motor startup).

MC68HC908GR60A LINKit Boards

As the aim was to use the MC68HC908GR16 MCU, which is pin compatible with the MC68HC908GR60A, and the GR16 LINKits boards were not available at the time of development, the GR60A LINKits boards were fitted with the GR16 MCUs. More details about the LINKit family can be found in Reference [4].

A simplified circuit connecting the GR16 LINKits board, the MC33887, and the door lock unit for the rear car door is shown in Figure 4.

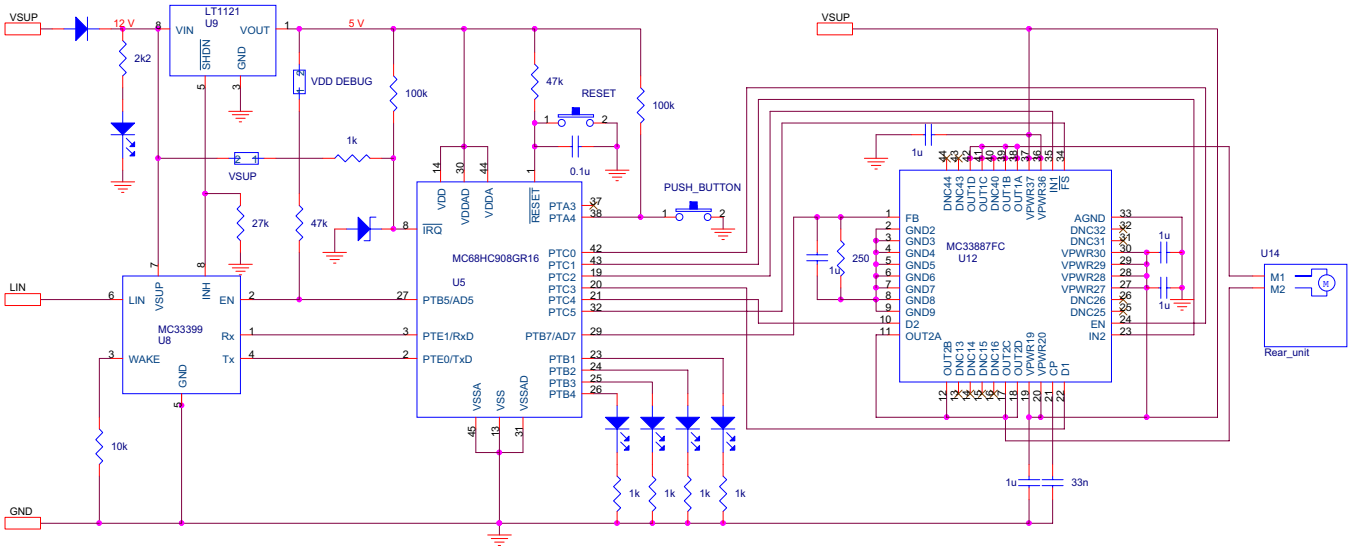

Figure 4. Simplified Rear Doors Lock System Circuit

The PTC port is used to connect the GR16 LINKits board and the MC33887 EVB. To monitor the MC33887 H-bridge output current, an external resistor is used to convert the current feedback to a proportional voltage, which is measured via channel 7 of the MCU A/D converter.

To enable cluster wake-up after a front door lock switch change of state, the circuit of the GR16 LINKits board, the MC33887, and the door lock unit for the front car door was slightly modified, as shown in Figure 5.

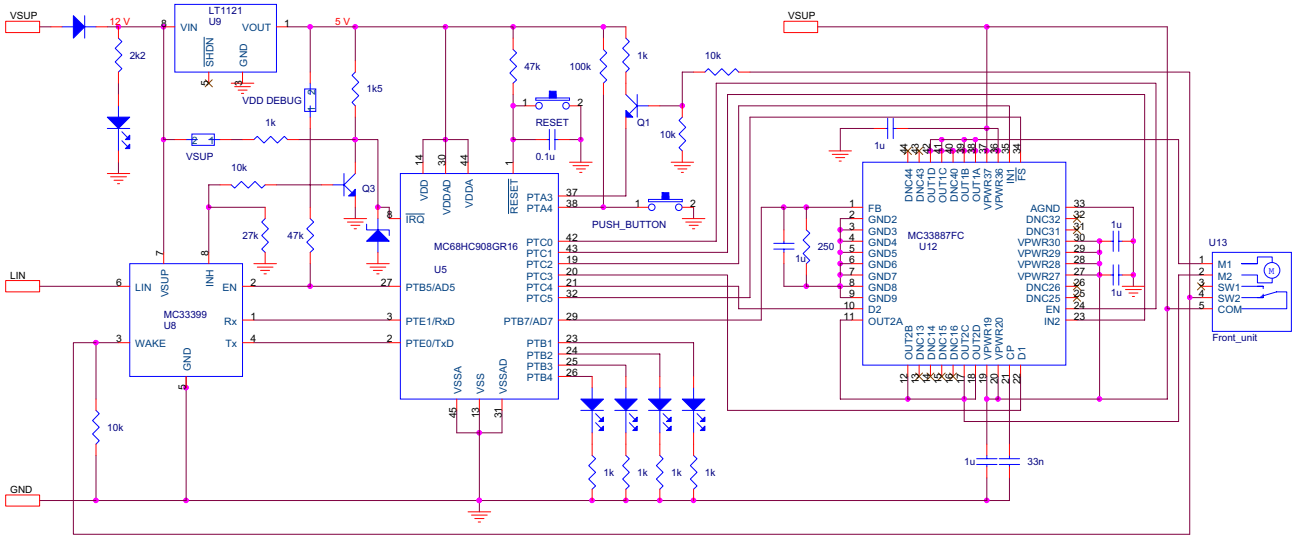

Figure 5. Simplified Front Doors Lock System

As the front door lock unit contains a 3-pin lock switch, one pin is connected to the wake-up pin of the MC33399 LIN interface, to enable wake-up of the LIN cluster via Q3 and the IRQ input of the GR16 MCU. To detect the lock switch position during the normal running mode, the lock switch is connected also to the PTA3 pin of the GR16 PTA port[1].

NOTE

Originally the LINKits boards are fitted with the MC33399 LIN physical interface. As the new, pin compatible, MC33661 LIN physical interface with many improvements (especially the slew rate control and low EMC emission) is available now, the MC33661 can be used instead of the MC33399.
If the MC33399 is replaced with the MC33661 on the LINKits board, a 10 kΩ series resistor should be added between the INH pin of the MC33661 and the SHDN pin of the LT1121 voltage regulator. More about the MC33661/MC33399 replacement can be found in Reference [7].

1. As the MC33399 wake-up pin threshold is typically 0.44 V_{SUP} (0.57 V_{SUP}, for low to high transition), the Q1 transistor is used to convert the voltages to TTL levels.

Software

The slave node application software is almost identical for the front and rear door units, the difference being that sending the lock switch status applies only to the front door unit.

Board Configuration File *board.h*

To enable easy configuration changes, all hardware dependent data are concentrated in the *board.h* header file. The following information can be found in this file.

- Low power mode used (see Low Power Mode on page 11 for more details)
- MCU port where the MC33887 H-bridge is connected
- Position of the lock switch pin (for the front door unit)
- A/D limit for the current end stop position detection
- Definition of the delay for repeated A/D reading (to suppress the starting current)
- Position of the MC33399 LIN interface enable pin

Main Program Structure

The main program flow chart can be seen in Figure 6.

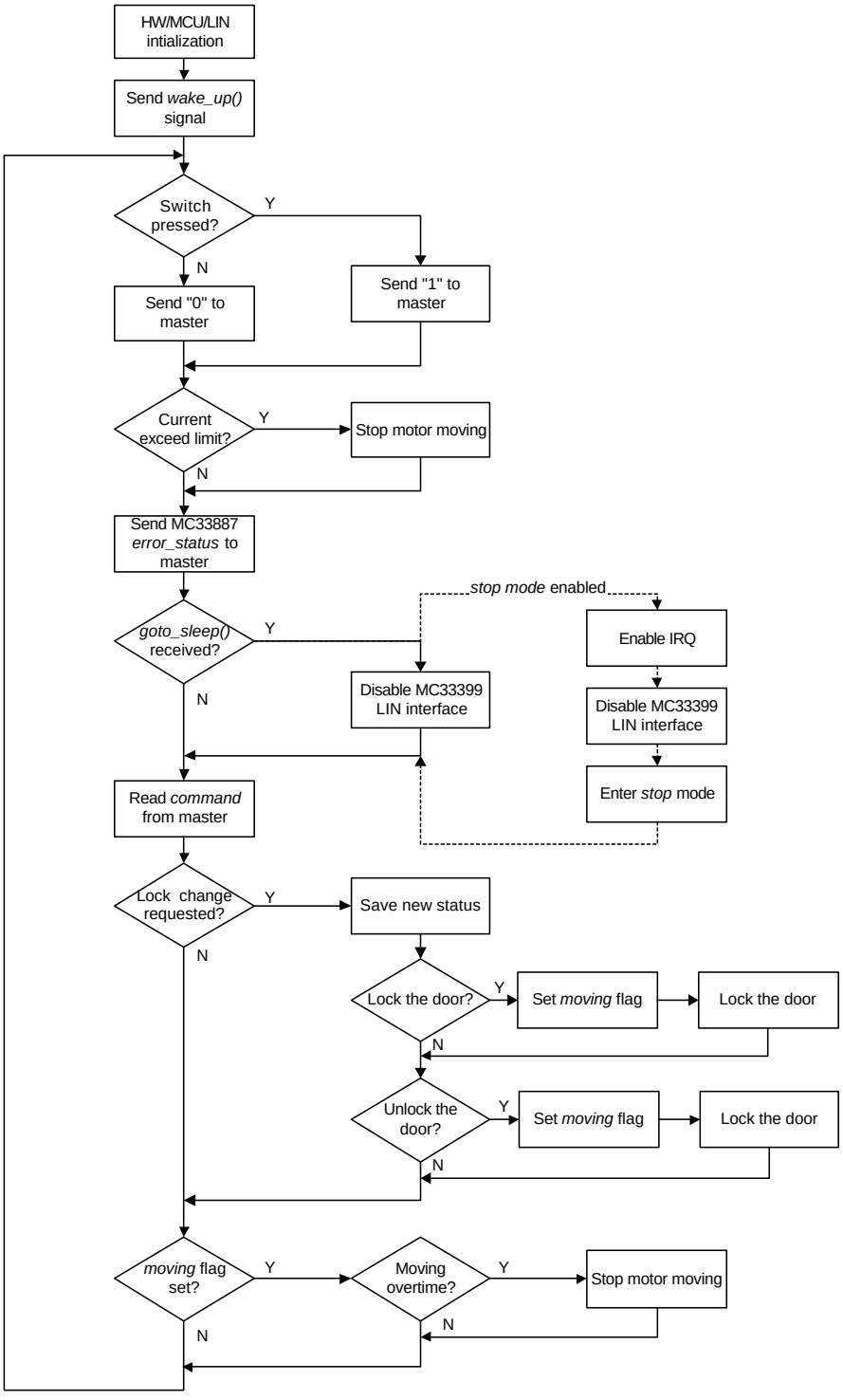

Figure 6: Main Program Flow Chart

After all necessary hardware initialization (of the MC33399 LIN interface and the MC33887 H-bridge), the LIN driver is initialized in the *main()* function. The initialization procedure for the LIN 2.0 driver is described in Reference [3]. After initialization, the forever loop is entered. The following functions are periodically called within it.

- *PET_WATCHDOG()*, which serves the watch dog counter
- *SendStatus()*, sending the status of the lock switch to the master node[1]
- *CheckVoltage()*, detecting the backstop via the MC33887 H-bridge current feedback
- *CheckError()*, providing the master node with information about issues on the MC33887 H-bridge (shortcut on the output, over temperature, etc.)
- *CheckLowPower()*, taking care of the system low power mode
- *MotorMng()*, driving the lock actuator

The most important function in the main forever loop is *MotorMng()*. As the information on the required lock position is distributed during the normal running mode consecutively, after the *lock_door* frame is read the information of the lock/unlock command is compared to the previous one. If the current lock/unlock command is different from the previous, an appropriate actuator action call is issued and the time of the call is memorized. If the actuator movement is not stopped within the *LOCK_OVER_TIME* time period by the *CheckVoltage()* function (via the backstop current accrual), it is stopped after the *LOCK_OVER_TIME* time period, and the error flag is set. After a transmission, this error flag can be used by the master node application to detect any malfunction of the door lock actuator or door lock.

Low Power Mode

To reduce the system power consumption, low power modes are included in the slave node unit software. The cluster low power mode is driven by the master node application. When the master node does not detect any lock switch status change within a five second period, it sends the *goto_sleep* command to the LIN bus (details of the LIN low power modes can be found in Reference [3]).

To detect the *goto_sleep* command on the slave node side, the *l_ifc_read_status_i1()* function is used:

```
if ((l_ifc_read_status_i1() & 0x0008)== L_SLEEP_REQUEST)     /* Sleep command received? */
{
    ....
}
```

Two typical low power modes are implemented in the slave unit software.

- System power-down
- MCU stop mode

System Power Off

The system power-down approach is shown in the rear door units. After a *goto_sleep* command reception, the MCU disables the MC33399 LIN interface. It causes the switchable voltage regulator to

1. Applicable only to the front door lock unit.

shut down (via the $\overline{\text{SHDN}}$ input of the LT1121 voltage regulator on the LINKits board). There are only two ways to wake up the system after power down: by a LIN wake-up via the LIN bus, or by a slave node reset.

MCU Stop Mode

The MCU stop mode is used in the front door units. After a *goto_sleep* command reception, the external IRQ is enabled, the MC33399 LIN interface is disabled, and the *STOP* instruction is executed. Since the bonding between the MC33399 LIN interface and the switchable voltage regulator is disconnected, it does not cause a system power down, but the $\overline{\text{IRQ}}$ input of the MCU is pulled high. If the status of the lock switch is changed while the MCU is in stop mode, the MC33399 INH output is enabled, and the GR16 $\overline{\text{IRQ}}$ pin is pulled low. This causes the GR16 to wake up from stop mode and, after initialization of the LIN driver, to transmit a LIN *wake_up* request.

As the *wake_up* request transmission is executed after every reset, the LIN 2.0 specification is violated[1].

References

1. MC68HC908GR16 Data sheet,
   Freescale document number: MC68HC908GR16/D
2. LIN Specification Package, Revision 2.0, 23 September 2003,
   www.lin-subbus.org
3. *LIN 2.0 connectivity on Freescale 8/16bit MCUs using Volcano LTP,*
   Freescale document number: AN2767
4. *LINKits LIN Evaluation Boards*,
   Freescale document number: AN2573
5. *MC33887 Data sheet*,
   Freescale document number: MC33887/D
6. LIN 2.0 Mirror Unit Slave,
   Freescale document number: AN2885
7. MC68HC908QB8 LIN kits Auto-addressing LIN Slave Node,
   Freescale document number: AN2760

1. According to the LIN 2.0 specification, the *wake_up* request should be transmitted only when the cluster was previously in sleep mode.

Acronyms

API	Application Program Interface
CAN	Controller Area Network
CFG	Target Hardware Configuration File
ECU	Electronic Control Unit
ID	IDentifier
ISR	Interrupt Service Routine
LCFG	LIN Configuration Tools
LIN	Local Interconnect Network
LTP	LIN Target Package
MCU	Microcontroller Unit
NAD	Node Address for Diagnostic
PRV	Private File
SBC	System Basis Chip
SCI	Serial Communications Interface
UART	Universal Asynchronous Receiver and Transmitter
VCT	Volcano Communications Technologies
UART	Universal Asynchronous Receiver and Transmitter
VCT	Volcano Communications Technologies

How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

AN2884
Rev. 1, 12/2004

freescale™
semiconductor