Freescale Semiconductor
Application Note

AN2885
Rev. 0, 11/2004

LIN 2.0 Mirror Unit Slave

Based on the MC68HC908EY16 MCU and the LIN 2.0 Communication Protocol

by: Petr Cholasta
8/16-bit Systems Engineering
Roznov pod Radhostem, Czech Republic

Introduction

The main purpose of this application note is to describe and demonstrate the use of the 8-bit MC68HC908EY16 MCU in a car door mirror application.

The other goal is to introduce the Volcano LIN Target Package (VCT LTP 2.0) as a tool for the implementation of a LIN 2.0 enabled LIN node (see Reference [1]).

The VCT LTP 2.0 represents a new approach to the development and generation of the LIN cluster. The benefit of using this software package is the high level approach to creating and rebuilding a LIN cluster. This contributes to the lucidity of the LIN cluster generation process. For a general overview on implementation and use of this software package see AN2767 Reference [2].

The theme of this Application note is mainly focused on the demonstration of the capabilities and performance of the MC68HC908EY16 MCU used in the LIN 2.0 enabled design. The mirror application was chosen as a typical application of the MCU.

The complete application software for the left slave unit (AN2885SW) can be downloaded from the Freescale web site at http://www.freescale.com.

freescale™
semiconductor

General Description

This application note describes the LIN mirror unit slave as a part of the complete mirror system. This system is primarily developed for the automotive industry, although this solution may be suitable for other applications where it is necessary to control a target device via the LIN bus.

System Outline

Figure 1 shows the mirror system concept. The LIN master unit controls both the left and right slave units via the LIN bus, according to the commands on the superior bus (e.g. CAN) or to the signals from the keyboard. Each slave unit controls a dedicated mirror unit, and reports back its state to the master unit via the LIN bus.

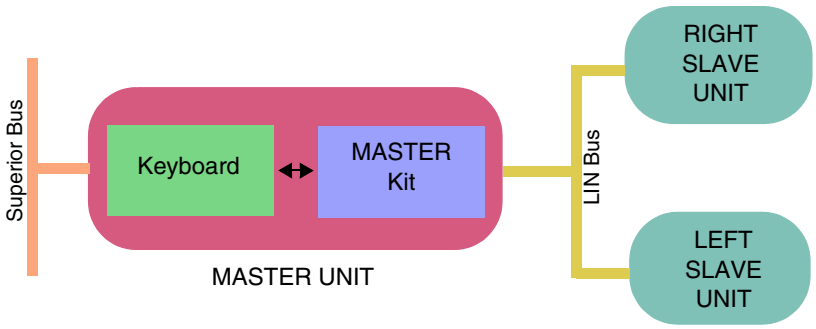

Figure 1. Mirror System Outline

Slave Unit Features

The slave unit is capable of controlling a mirror unit according to the master commands and reporting its current status back to the master.

Slave unit capability:

- tilt mirror
- detect an obstruction when moving the mirror
- provide information on errors occurred, when the mirror last moved
- provide information on temperature
- control the on-board LED (as an example of general I/O control)

Part of this application note is the software package for the left slave unit implementation. However, this software package could be easily rebuilt to implement the right slave unit.

Slave Concept

The slave hardware board implementation is strictly modular; it is built on general purpose EVBs provided by Freescale. In this case, the slave board consists of two development boards. The first is the EY16 LINkit, consisting of MCU MC68HC908EY16, LIN physical interface MC33399, voltage regulator LT1121, and a programming / debug interface. The second is the dual H-bridge MC33922 EVB (see Figure 2).

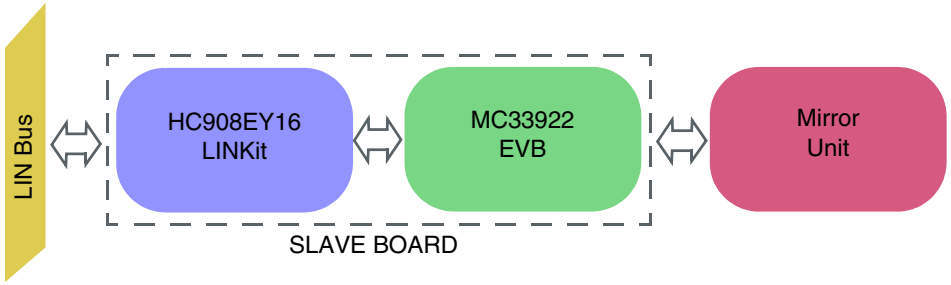

Figure 2. Slave Unit Concept

NOTE

Highly integrated single-package solution MM908E625 Reference [8], a member of the Freescale IDC family, is recommended for use in a LIN mirror slave application. It incorporates a MC68HC908EY16 MCU, LIN physical interface, voltage regulator, four half bridges, and one high side switch, all in one 54 lead plastic case.
For more information on IDC products, see Reference [9].

Mirror Unit

This application allows the control of two mirror unit systems, which are currently used for this kind of application. The first one is a two motors mirror unit, where each movement axis is controlled by one dedicated DC motor. The second system uses only one DC motor and an electromagnet. The electromagnet handles the clutch, which connects the DC motor to either the ahead/back, or the up/down movement mechanism.

As both applications use the same hardware platform, versatility is achieved by the software modification.

Freescale Components Used

Microcontroller Unit MC68HC908EY16

Features:

- Standard features of CPU08
- 8 MHz internal bus frequency at 5V
- Internal oscillator with no external components
  - Software selectable bus frequencies
  - 25 percent accuracy with a trimming capability of better than 1 percent
  - Clock monitor
  - Option to allow use of external clock source or external crystal / ceramic resonator
- 15,872 bytes of on-chip FLASH memory with in circuit programming
- 512 bytes of on-chip RAM
- Internal clock generator module (ICG)
- Two 16-bit, 2-channel timer (TIMA and TIMB) interface modules with selectable input capture, output compare, and pulse-width modulation (PWM) capability on each channel
- 8-channel, 10-bit successive approximation analog-to-digital converter (ADC)
- Enhanced serial communications interface (ESCI)
- Serial peripheral interface (SPI)
- Timebase module (TBM)
- Low voltage inhibit module (LVI)
- 5-bit keyboard interrupt
- 24 general I/O pins
- External asynchronous interrupt pin (/IRQ)
- System protection features:
  - Optional computer operating properly (COP) reset
  - Ilegal opcode detection with reset
  - Ilegal address detection with reset
- Standard low-power modes of operation:
  - Wait mode
  - Stop mode

The inner structure is shown in Figure 3. For more information see Reference [5].

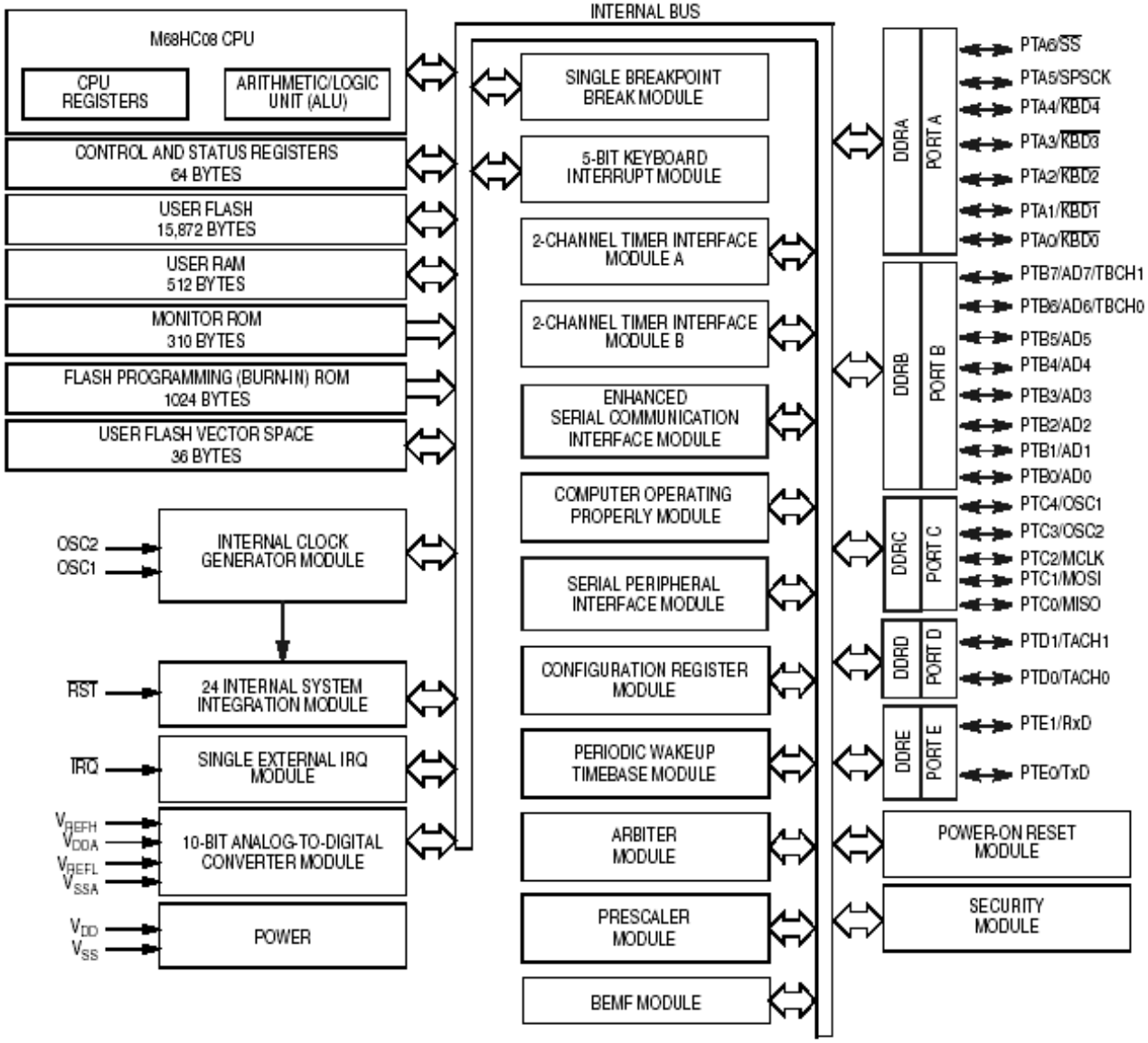

Figure 3. MC68HC908EY16 MCU Block Diagram

LIN Physical Interface MC33399

This component (depicted in Figure 4) is designed for use in master and slave LIN nodes as a bus voltage converter with an implemented bus wake-up capability (see Reference [6]).

Device features:

- Communication speed up to 20 kbps
- Interfaces to the MCU with CMOS compatible I/O pins
- Two operational modes: Normal and Sleep
- Very low standby current of 20 µA during Sleep mode
- An unpowered node does not disturb the LIN network
- Wake up capability from the LIN bus, or MCU, or by high voltage on the wake-up pin
- Controls an external voltage regulator
- High EMC immunity

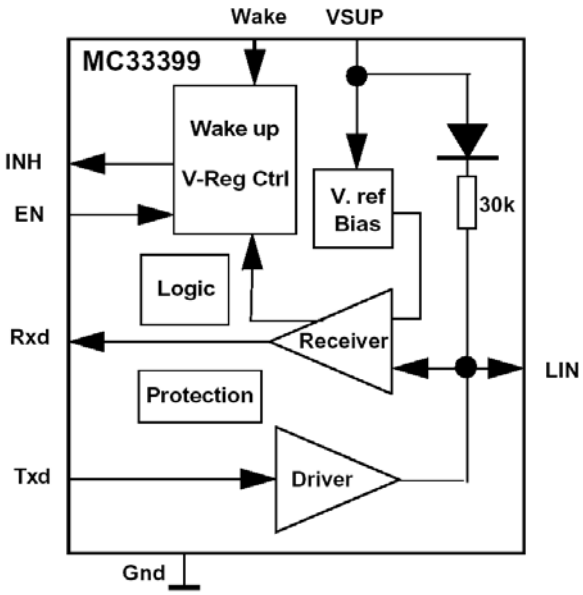

Figure 4. MC33399 (LIN Physical Interface) Block Diagram

NOTE

The new eLIN physical interface MC33661 (Reference [7]) fully replaces the MC33399 described above[1]. With a signal slew rate selection option, active bus signal shaping, and a special mode for operating above 100 kbps for testing and programming, it provides excellent EMC behavior and capability for programming MCU memory via LIN bus.

1. On the LINkit board, replacing the MC33399 with the MC33661 requires the addition of a 10 kΩ resistor between the MC33661 INH pin and LT1121 /SHDN pin, because of the MC33661 higher INH pin drive capability.

Dual H-bridge MC33922

The MC33922 is a dual H-bridge with load current feedback and short circuit protection (see Figure 5).

System features:

- Current recopy feature (load current control)
- Active current limiting via internal constant OFF-time PWM
- Output short circuit protection with shutdown
- Undervoltage, Overvoltage, Overtemperature and Overload reporting
- Normal and Sleep modes (in Sleep mode, current consumption is 50 µA per H-bridge)

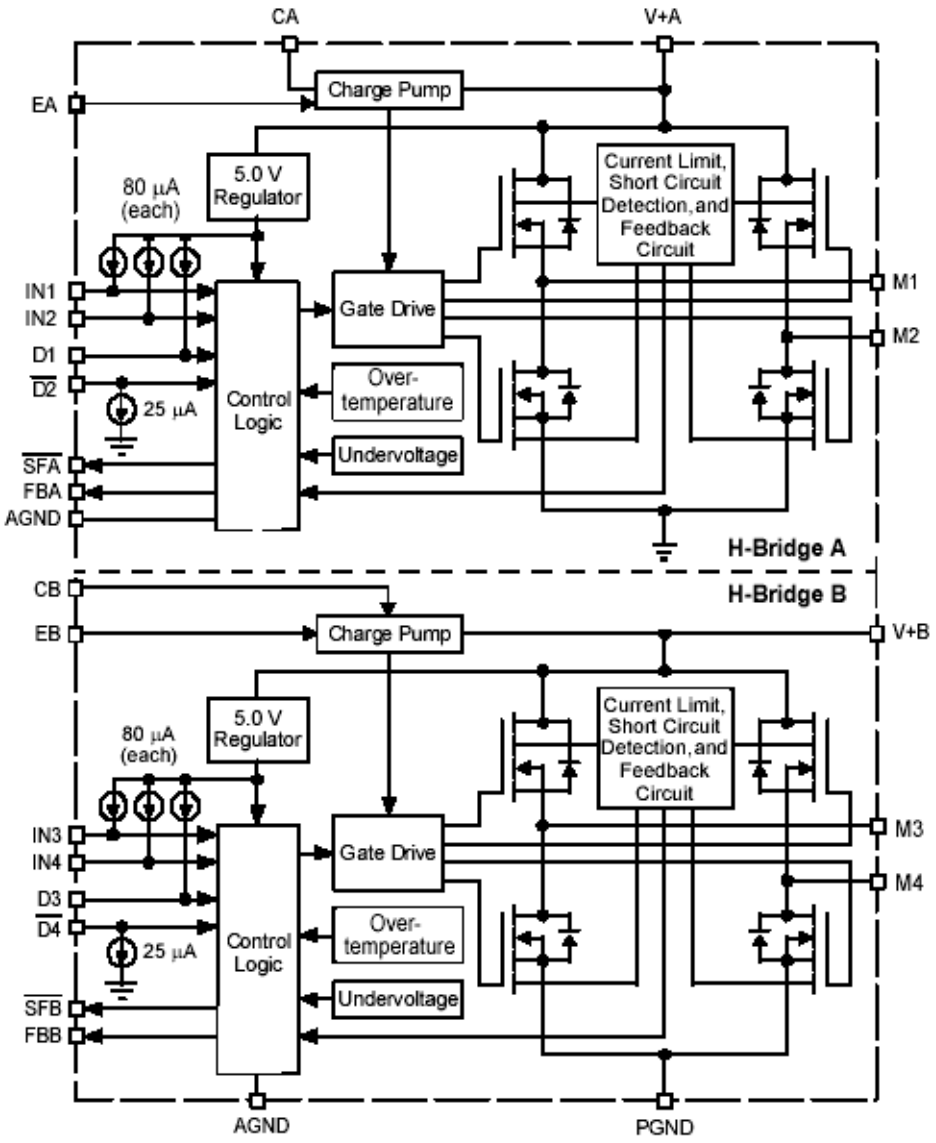

Figure 5. MC33922 (Dual H-bridge) Block Diagram

Hardware Description

Slave Unit Schematic

The hardware schematic is shown in Figure 6. It covers the connection of two development boards - MCU MC68HC908EY16 LINkit and the dual H-bridge MC33922 EVB (see Slave Concept on page 3). For schematic lucidity, only those EVB segments which are necessary for this application are included. For detailed information on used LINkit see Reference [3].

The temperature sensor MAX6611 was assembled in the prototyping area of the LINKit board[1]. This IC offers voltage reference[2] for the MCU ADC module and ambient dependent temperature output[3]. More information about temperature measurement using this type of sensor can be found in Reference [4].

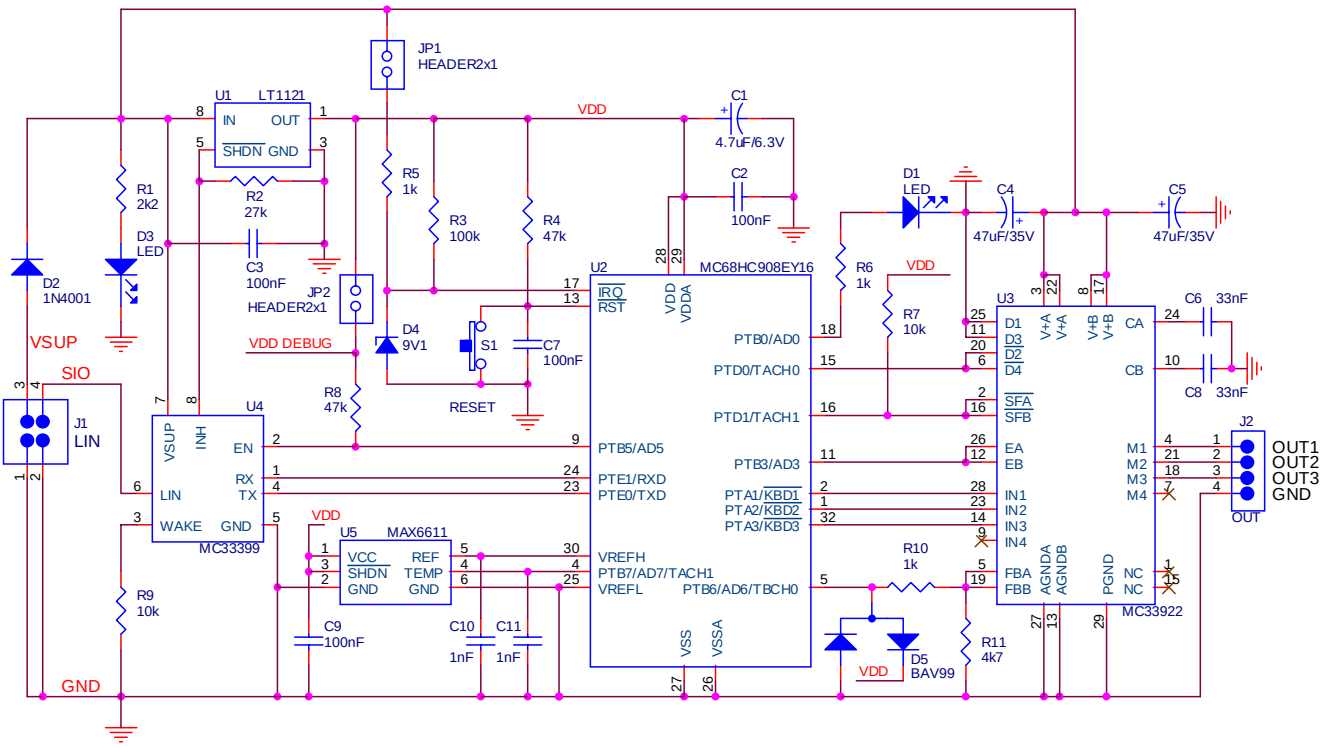

Figure 6. Slave Unit Schematic Diagram

NOTE

The MC68HC908EY16 schematic diagram shows only those MCU pins that are used in this application. The other MCU pins are hidden.

1. On the LINkit board, the MCU V_{REFH} pin is connected to the power supply voltage VDD. Assembling the temperature sensor MAX6611 requires cutting a track between the VDD and V_{REFH} MCU pins.
2. The voltage reference equals 4.096V.
3. The sensor output voltage slope is 16 mV/ °C. For temperature Ta = 0°C, the output voltage equals 1.2 V.

The value of the dual H-bridge MC33922 output current is sensed via resistor R11. Considering that the H-bridge output current to the sense pin current ratio is about 375, the maximum output current measurable by the ADC is:

$$I_{OUTMAX} = \frac{U_{REF}}{R11} \times C_R = \frac{4,096}{4700} \times 375 = 330mA \quad \text{(EQ 1)}$$

where:

- U_{REF} = MCU ADC reference pin voltage [V]
- C_R = dual H-bridge current recopy ratio [-]
- R11 = value of sense resistor [Ω]

In the case of a higher output current, resistor R10 and double diode D5 protect the MCU AD6 pin against damage from the high voltage.

The JP1 and JP2 jumpers are used only during programming and debugging of the MCU. For more information about the LINkit board MCU programming abilities, see Reference [3].

The MCU does not require a crystal for clock generation, as it has its own internal oscillator (ICG module). Trimming of the oscillator is done via bit time measurement, which is processed during a LIN frame synchronization field reception (see Reference [1]).

The LINkit board circuit configuration also allows the slave unit to enter the Sleep mode, when power consumption is minimal. This feature fulfils the LIN specification requirement of the supply power saving during no LIN bus activity (see Reference [1]).

Mirror Unit Description

The mirror unit is described in terms of:

- connection to the slave unit
- control by the slave unit

Mirror movement directions are shown in Figure 7.

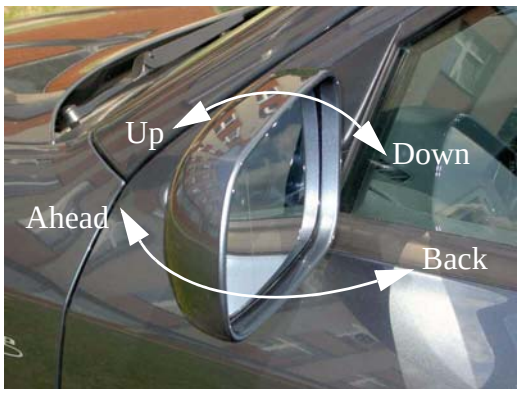

Figure 7. Mirror Movement Terminology

Two DC Motor Mirror Platform

In this configuration, the mirror position is controlled using two DC motors. Each motor moves the mirror along one axis.

Figure 8 shows how this type of mirror unit is connected to the slave unit. Inside the mirror unit there are two motor outputs connected together. Therefore, only three slave unit outputs can fully drive the mirror in all possible movement directions.

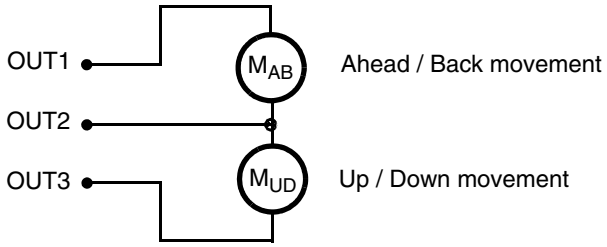

Figure 8. Two Motor Mirror Unit Connection

The direction of mirror movement depends on the state of the slave unit outputs. Table 1 specifies this relationship, and also covers current threshold limits for an error detection. They are defined as symbolic constants in the software (see File hw_platform.h on page 15) and their values are closely linked with the R11 resistance value (see (EQ 2)). These current limits are dependent on the DC motors' parameters and they can vary for different types of mirror unit.

The explanation of the various error states is based on the output current value measurement. No error is detected when the output current value lies within these limits.

Table 1. Relationship Between the Output Pattern and Mirror Movement(1)

Mirror movement	OUT1	OUT2	OUT3	Low current threshold	High current threshold
Ahead	1	0	0	50 mA	100 mA
Back	0	1	1		
Up	0	0	1		
Down	1	1	0		

NOTES:
1. Symbols in the columns OUTx mean: 1 - power supply voltage, 0 - power supply ground

Electromagnet and DC Motor Mirror Platform

In this case, one DC motor controls the mirror movement. The motor is connected via the clutch (controlled by the electromagnet) to the mechanism for either the ahead/back movement, or the up/down movement. Connection of this mirror unit to the slave unit board is shown in Figure 9.

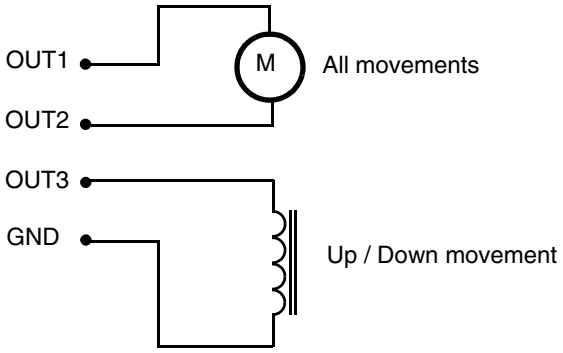

Figure 9. Electromagnet and Motor Mirror Unit Connection

Because an electromagnet is used, four wires are needed to control the mirror. However, only three dual H-bridge outputs are used as active control outputs (as is the case described in Two DC Motor Mirror Platform on page 10). The ground wire is present in a typical mirror unit (e.g. for heating or lighting connections). Therefore, from a hardware point of view, this method of mirror control seems to be equal to the previously described one.

Table 2 specifies the direction of mirror movement according to the state of slave unit outputs. Table 2 also covers current threshold limits for error detection. These limits are determined by the software (see File hw_platform.h on page 15) and they are closely linked to the R11 resistance value (see (EQ 2)). They are dependent on the DC motor and electromagnet parameters and they can vary for different types of mirror unit.

The explanation of the various error states is based on the output current value measurement. No error is detected when the output current value lies within these limits.

Table 2. Relationship Between the Output Pattern and Mirror Movement(1)

Mirror movement	OUT1	OUT2	OUT3	Low current threshold	High current threshold
Ahead	1	0	0	50 mA	100 mA
Back	0	1	0		
Up	1	0	1	200 mA	300 mA
Down	0	1	1		

NOTES:
1. Symbols in the columns OUTx mean: 1 - power supply high voltage, 0 - power supply ground.

Software Description

General Description

The project software can be separated into two main parts. The first part comprises the LIN connectivity related routines; the second part is the mirror application itself. Figure 10 shows how the project folders are arranged in the project directory.

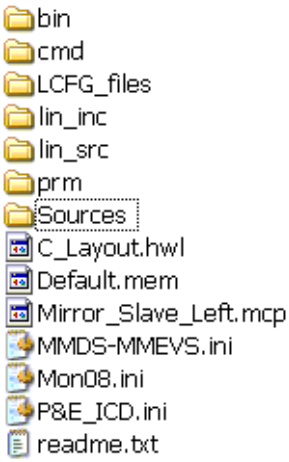

Figure 10. Project Folders

LIN Connectivity Related Routines

The LIN connectivity software covers all LIN communication (see Appendix A — Messaging Strategy on page 23). The mirror application controls LIN communication by the receiver and the transmitter signal command buffers and flags.

LIN connectivity is implemented using the VCT LTP 2.0[1] software package (see Reference [2]).

In the project directory, the VCT LTP 2.0 package is represented by folders as follows:

- LCFG_files
  - *Mirror_net.ldf* (mirror network LIN description file)
  - *slave_left.prv* (left slave private file)
  - *slave_right.prv* (right slave private file)
  - *uart.cfg* (includes MCU ESCI module parameters definition)
  - *build.bat* (batch file for generation of LIN node files (*l_gen.h* and *l_gen.c*))
- lin_inc (header files folder - includes generated *l_gen.h*)
- lin_src (source files folder - includes generated *l_gen.c*)

1. Release version LTP20_1_4_0.

Note that the *l_gen.h* and *l_gen.c* files present in the downloadable application (AN2885SW) are generated for the left slave mirror unit. However, for the right slave mirror unit, the *slave_right.prv* file is pre-created, so the *l_gen.h* and *l_gen.c* can be easily generated.

Mirror_net.ldf (mirror network LIN description file) includes a definition of the mirror application LIN cluster (see Mirror_net.ldf on page 24). In the case of this application, the network consists of one master controlling both the left and right slaves. The slaves control the dedicated mirror unit according to the master commands and report their status back to the master (detailed information can be found in the General Description on page 2).

The LIN description file contains the definitions of network signals, frames with corresponding identifiers, node attributes, and schedule tables (see Mirror_net.ldf on page 24). The schedule table determines the frame order to be processed. It also defines the period of the next frame processing call.

The transmitted and received LIN frames consist of a break field, synchronization field, identifier, data field, and checksum (see Reference [1]). All LIN network nodes (master and slaves) choose, according to the identifier, to either ignore or to process the received LIN frame. The identifier also determines whether the node has to read the frame data field or to transmit a data field. The frame data field consists of predefined signals (see *signals* in Mirror_net.ldf on page 24). The checksum which is the last byte in the frame, is calculated over the identifier and data field values.

Each LIN slave has its own defined node address (see NAD in the *node attributes* part of Mirror_net.ldf on page 24), which is used for diagnostics and configuration. In this application, NAD 0x45 (hex) was assigned to the left slave. If the network is powered-up, the master starts to send master request frames (see the *net_configuration* part of the schedule table in Mirror_net.ldf on page 24 for the cluster configuration). The master request frames consist of the slave NAD and configuration commands. During the LIN cluster configuration, identifiers are assigned to each slave node frame. This process allows the configuration of the required task for each slave node in the LIN network. After successful configuration, the master runs the *run_mode* schedule table (see Mirror_net.ldf on page 24), i.e. the master starts to control the mirror network left and right slaves.

The **build.bat** batch file generates the LIN connectivity node files *l_gen.h* and *l_gen.c*. These files are manually included in the Metrowerks CodeWarrior stationery *lin_inc* and *lin_src* project folders, and, with the *lin.lib* library, they create the LIN connectivity part of the node application.

The LIN connectivity node files are generated according to the parameters defined in **slave_left.prv** (left slave private file). These parameters are as follows (see also slave_left.prv on page 27).

- Define LIN network
- Define generated node
- Define node MCU UART module parameters
- Define communication flags

The main application code (*main.c* file) uses the communication flags to recognize that a new signal was received.

Mirror Application Software

The mirror application software controls the mirror unit according to the master node commands (see Appendix A — Messaging Strategy on page 23).

The functions provided are as follows.

- Moving the mirror (done by MC33922)
- Reading temperature (temperature sensor MAX6611)
- Controlling LED (on LINkit board marked as D8) — ON/OFF/FLASH
- Error and status reporting:
  - Mirror moving
  - Detect an obstruction when attempting to move mirror
  - No mirror unit connected
  - Outputs of MC33922 shorted
  - Dual H-bridge overtemperature
  - On-board LED is ON/OFF.

The application software is represented in the project directory by the folder:

- Sources
  - hw_platform.h (target dependent stuff)
  - main.h (definition of data types used in project)
  - main.c (application main control file)

Metrowerks CodeWarrior Project Stationery

Figure 11 shows the standard Metrowerks CodeWarrior project stationery structure.

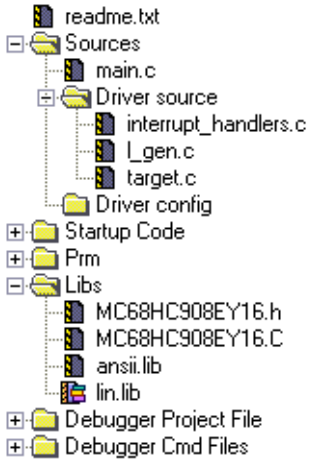

Figure 11. Metrowerks CodeWarrior Project Stationery

The stationery includes:

- LIN driver
  - interrupt_handlers.c — ESCI interrupt service routine call
  - l_gen.c — LIN message definitions (in this case generated for the left slave unit)
  - target.c — ICG clock trimming routine.
  - lin.lib — LIN library
- Mirror application
  - main.c — application control code,
  - MC68HC908EY16.h — EY16 register mapping, as a standard Metrowerks CodeWarrior file,
  - MC68HC908EY16.c — EY16 register type definition, as a standard Metrowerks CodeWarrior file.

Mirror Application Software Description

The mirror application software files and their impact on the whole project are described below.

File hw_platform.h

This file covers target dependent stuff. It acts as an interface between the target hardware and the application code (main.c file). The benefit of this approach is a straightforward method of porting the application to another target.

This file includes definitions of:

- controlled mirror unit type
- used MCU pin definition
- dual H-bridge load (motors, electromagnet of mirror unit) current thresholds
- patterns for setting of dual H-bridge outputs, when controlling mirror tilting
- flash LED time

The current threshold constants were calculated according to the equation (EQ 2). It specifies the relationship between the measured output current and its representative value after A/D conversion.

$$I_{OUT(AD)} = \frac{\frac{I_{OUT}}{C_R} \times R11}{\frac{U_{REF}}{255}} = \frac{\frac{I_{OUT}}{375} \times 4700}{\frac{4,096}{255}} = I_{OUT} \times 780 \qquad \text{(EQ 2)}$$

where:

- $I_{OUT(AD)}$ = unsigned 8-bit (0 - 255) current representative after the A/D conversion [-]
- I_{OUT} = output current [A]
- C_R = current recopy ratio of the dual H-bridge [-]
- R11 = value of sense resistor [Ω] (see Figure 6)
- U_{REF} = MCU ADC reference pin voltage [V]

File main.h

This file incudes definitions of types used in the main.c file.

File main.c

This file contains the main application code which uses the LIN connectivity software.

The main routine flow chart can be seen in Figure 12. The flow chart building blocks are described below.

The **Initialize MCU** block of the main routine flow chart (Figure 12) covers the initialization of the LIN connectivity software, the mirror application software, and the MCU peripherals (ICG, TBM, ADC, ESCI, I/O).

The peripherals used are configured as follows:

- ICG — clock generator, clock frequency 25.5 MHz +- 1%, bus frequency 6.37 MHz +- 1% (tolerance is valid after trimming, which is done by the VCT LTP 2.0)
- TBM — source of the periodic interrupt arising every 10 ms
- ADC
  - used for the A/D conversion of the mirror unit hardware platform current (necessary for current value checking)
  - for the temperature dependent voltage reading
- ESCI — LIN communication interrupt driven
- I/O — standard input/output ports for the dual H-bridge, LIN physical layer and voltage regulator control.

When the TBM and ESCI interrupts are enabled, one of the following interrupts can occur.

- TBM interrupt — In the body of the service routine, the following variables are decremented:
  - *inrush_cur_timeout* counter. When the voltage is applied across the motor, its current rises above maximum limits for a short period of time. To avoid false reporting while the motor is not fully running, it is necessary to stop checking the current. This variable establishes a 200 ms period of no current checking.
  - *flash_led_time* counter. If the flash LED command was received, the value loaded to this counter determines the period of the LED turn ON state. When this counter reaches the value “one”, the LED is turned OFF. The loaded counter value equates to a 30 ms period of LED ON state.
  - *enter_sleep_mode* counter, which counts the amount of time remaining before the device enters Sleep mode.
- ESCI interrupt:
  - runs LIN driver
  - refreshes *enter_sleep_mode* counter. This feature is based on LIN specification Reference [1], which specifies that, after four seconds of no LIN bus activity, the device must enter Sleep mode.

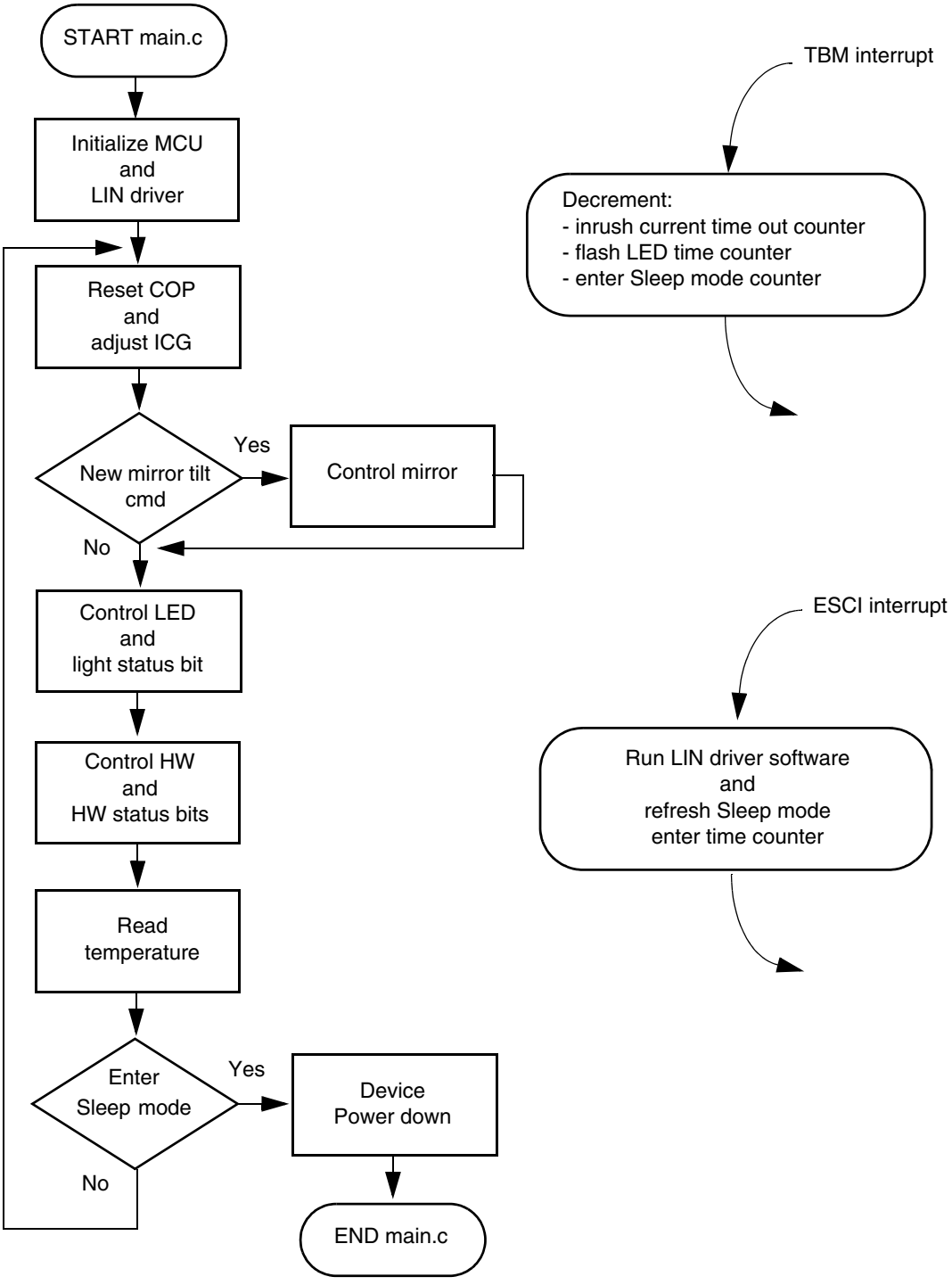

Figure 12. Main Routine Flow Chart

The **reset COP and adjust ICG** block of the main routine flow chart (Figure 12) covers:

- COP feeding
- ICG adjustment, which is done by the VCT LTP 2.0. The trimming routine is based on bit time measurement during LIN frame synchronization field reception.

The **new mirror tilt cmd** condition block branches the flow chart into two parts. If any new tilt command was received, different to the previous one, then apply this new command to the mirror unit. Otherwise do nothing.

The **control mirror** block includes control of the mirror movement mechanism in terms of:

- run the movement mechanism
- stop the mechanism moving.

The **control LED and light status** bit covers LED control according to the actually received light command. The capabilities of device LED control are:

- turn ON/OFF LED
- flash LED. In this case the time period of LED flash is determined by the *flash_led_time* counter.

The light status bit of the status frame is set, according to the current LED state.

The **control hardware and hardware status bits** block substitute the function that reads the motor / electromagnet current value. Then it checks if the measured current value is within predefined limits. If the current is out of those limits, it stops the mechanism moving, and reports one of the following errors:

- obstruction detected — motor current is higher than the maximal limit, represented in code via *run_current_threshold_high* symbolic constant
- mirror unit is not connected — motor current is lower than the minimal limit determined in code via *run_current_threshold_low* symbolic constant
- output short circuit, or the dual H-bridge overtemperature — detect output hard short, or overtemperature of dual H-bridge by reading its fault pin

If the current value is between the limits, then it provides information about the mirror state, i.e. moving or not moving.

All states mentioned above are reported to the master using the hardware status bits.

The **read temperature** block covers the process of the A/D ambient temperature dependent voltage conversion and its representative storage in the LIN software transmit buffer. The voltage source is temperature sensor MAX6611 (see schematic in Figure 6). The relationship between the ambient temperature and its unsigned 8-bit representative (0 - 255), is fully described by the following formula:

$$Ta = Ta_{AD} - 75 \quad \text{(EQ 3)}$$

where:

- Ta = ambient temperature in [°C]
- Ta_{AD} = Ta equivalent voltage value after the A/D conversion [-] (range from 0 to 255)

The **enter Sleep mode** condition block of the main routine flow chart (see Figure 12) acts in the program as the Sleep mode activator. The Sleep mode is entered if either Sleep command was received, or four seconds of no LIN bus activity has expired. In Sleep mode, the device is powered down and it can be woken up by a LIN wake-up command (see Reference [1]).

Conclusion

This application note describes the use of the MC68HC908EY16 in a mirror application. The primary aim of the application is to show the performance and capabilities of the MC68HC908EY16 MCU. A secondary aim is to demonstrate LIN 2.0 connectivity, which, in this case, is implemented in the node using the Volcano Technology LIN Target Package 2.0 (VCT LTP 2.0)[1]. The complete left slave mirror application software (AN2885SW) can be downloaded from http:www.freescale.com.

The mirror application MCU memory usage is shown in Table 3.

Table 3. Particular Code Sizes

MCU memory type	MCU memory size	LTP 2.0 software occupies	Mirror application software occupies	Totally occupied area	Free space	Free space
FLASH	15,872 bytes	2817 bytes	913 bytes	3730 bytes	12,142 bytes	76%
RAM	512 bytes	49 bytes	16 bytes	65 bytes	447 bytes	87%

Total VCT LTP 2.0 memory consumption is dependent on which capabilities of the package are used. Memory consumption also increases with the number of LIN communication frames used. The mirror application uses the node configuration option, which is mandatory for LIN 2.0 Reference [1], and LIN connectivity without the diagnostics layer usage (see LIN Connectivity Related Routines on page 12).

The mirror application uses the following MCU peripherals.

- 2 of 8 A/D converter channels
- 12 of 24 I/O MCU pins
- TBM module
- IGC module
- ESCI module

The rest of the MCU peripherals are free and can be used by the user for other application purposes (see the MCU block diagram in Figure 3).

The MCU core performance can be represented by the time it takes to perform one main loop cycle (Figure 12). In the case of no ESCI interrupts, the one loop code service takes approximately 80 µs. In the case of an ESCI interrupt arising, LIN reception/transmission is required, and the total time rises to 200 µs.

1. Implemented VCT LTP 2.0 release version LTP20_1_4_0.

References

1. LIN Specification Package, Revision 2.0, 23 September 2003, LIN consortium
2. AN2767: LIN 2.0 connectivity on Freescale 8/16-bit MCUs using Volcano LTP, Freescale Semiconductor data sheet
3. AN2573: LINkits Evaluation Boards, Revision 1.0, 11/2003, Freescale Semiconductor data sheet
4. AN2623: LIN Temperature Sensor Using the MC68HC908QT/QY MCU, 11/2003, Freescale Semiconductor data sheet
5. MC68HC908EY16/D: MC68HC908EY16 Advance Information, Revision 4.0, 2/2003, Freescale Semiconductor data sheet
6. MC33399/D: MC33399 Automotive LIN Physical Interface, Revision 4.0, 8/2001, Freescale Semiconductor data sheet
7. APDPBA1: MC33661 eLIN Transceiver, 3/2003, Freescale Semiconductor Product Bulletin
8. MM908E625/D: MM908E625 Integrated Quad Half H-bridge with Power Supply, Embedded HC08 MCU and LIN Serial Communication, Revision 2.0, 09/2003, Freescale Semiconductor data sheet
9. Freescale Semiconductor web pages of Analog Product Division, Embedded MCU + Power: http://www.freescale.com/webapp/sps/site/taxonomy.jsp?nodeId=0143598784

Acronyms

A/D	Analog to Digital
ADC	Analog to Digital Converter
CAN	Controller Area Network
COP	Computer Operating Properly
eLIN	Enhanced LIN
ESCI	Enhanced Serial Communication Interface
EVB	Evaluation Board
IC	Integrated Circuit
ICG	Internal Clock Generator module
IDC	Intelligent Distributed Control
I/O	Input/Output ports
LED	Light Emitting Diode
LIN	Local Interconnect Network
LINkit	EVB for LIN development
LTP	LIN Target Package
MCU	Microcontroller Unit
TBM	Timebase Module
VCT	Volcano Communications Technology

Appendix A — Messaging Strategy

Node Name for Signal Provider	ID [0..5]	LIN ID Field (w/ parity)	Frame Name	Frame Description	Frame Size (Bytes)	Sig #	Signal Name	Signal Description	Signal Length (bits)	Signal Start Bit	Signal Initial Value	Raw Value Range
MIRROR_MASTER	0x30	0xF0	MIRROR_R_CMD	Mirror right command	1	0	MIRROR_R_T_CMD	Mirror right tilt command	4	0	0	0 - STOP MIRROR TILTING 1 - AHEAD MIRROR TILTING 2 - BACK MIRROR TILTING 4 - UP MIRROR TILTING 8 - DOWN MIRROR TILTING
						1	MIRROR_R_L_CMD	Mirror right light command	2	4	0	0 - LIGHT OFF 1 - LIGHT ON 2 - LIGHT FLASH
MIRROR_R	0x31	0xB1	MIRROR_R_STATUS	Mirror right status	1	0	MIRROR_R_HW_ST	Mirror right hardware status	4	0	0	0 - CURRENTLY NONE REPORT 1 - MOVING THE MIRROR 2 - DURING MIRROR TILTING AN OBSTRUCTION DETECTED 4 - MIRROR UNIT IS NOT CONNECTED 8 - OUTPUT SHORT CIRCUIT OR H-BRIDGE OVERTEMPERATURE DETECTED
						1	MIRROR_R_L_ST	Mirror right light status	1	4	0	0 - LIGHT OFF 1 - LIGHT ON
						2	MIRROR_R_RESP_ERR	Mirror right response error	1	7	0	0 - NONE ERROR IN THE LAST FRAME RESPONSE FIELD 1 - ERROR IN THE LAST FRAME RESPONSE FIELD
MIRROR_R	0x32	0x32	MIRROR_R_TEMP	Mirror right temperature	1	0	MIRROR_R_TMP	Mirror right temperature	8	0	0	0x00:0xFF
MIRROR_MASTER	0x33	0x73	MIRROR_L_CMD	Mirror left command	1	0	MIRROR_L_T_CMD	Mirror left tilt command	4	0	0	0 - STOP MIRROR TILTING 1 - AHEAD MIRROR TILTING 2 - BACK MIRROR TILTING 4 - UP MIRROR TILTING 8 - DOWN MIRROR TILTING
						1	MIRROR_L_L_CMD	Mirror left light command	2	4	0	0 - LIGHT OFF 1 - LIGHT ON 2 - LIGHT FLASH
MIRROR_L	0x34	0xB4	MIRROR_L_STATUS	Mirror left status	1	0	MIRROR_L_HW_ST	Mirror left hardware status	4	0	0	0 - CURRENTLY NONE REPORT 1 - MOVING THE MIRROR 2 - DURING MIRROR TILTING AN OBSTRUCTION DETECTED 4 - MIRROR UNIT IS NOT CONNECTED 8 - OUTPUT SHORT CIRCUIT OR H-BRIDGE OVERTEMPERATURE DETECTED
						1	MIRROR_L_L_ST	Mirror left light status	1	4	0	0 - LIGHT OFF 1 - LIGHT ON
						2	MIRROR_L_RESP_ERR	Mirror left response error	1	7	0	0 - NONE ERROR IN THE LAST FRAME RESPONSE FIELD 1 - ERROR IN THE LAST FRAME RESPONSE FIELD
MIRROR_L	0x35	0xF5	MIRROR_L_TEMP	Mirror left temperature	1	0	MIRROR_L_TMP	Mirror left temperature	8	0	0	0x00:0xFF
MIRROR_MASTER	0x3C	0x3C	MasterReq	LIN Master Request Command	1	0	SYS_SLEEP	System Sleep Mode	8	0	0	0x00:0x7F
-	0x3D	0x7D	SlaveResp	LIN Slave Response Command	0	-	-	-	-	-	-	-

Apendix B : Software listing

Mirror_net.ldf

```
/**************************************************************************
*
* Freescale Semiconductor Inc. 2004 All rights reserved
*
**************************************************************************
*
* THIS SOFTWARE IS PROVIDED BY FREESCALE SEMICONDUCTOR "AS IS" AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED.  IN NO EVENT SHALL FREESCALE SEMICONDUCTOR OR ITS CONTRIBUTORS
* BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*
**************************************************************************
*
* Author : rc574c
*
**************************************************************************/
LIN_description_file;
LIN_protocol_version = "2.0";
LIN_language_version = "2.0";
LIN_speed = 10.4 kbps;

Nodes
{
    Master: master, 5ms, 1ms;
    Slaves: slave_right, slave_left;
}

Signals
{
    mirror_r_t_cmd : 4, 0, master, slave_right;
    mirror_r_l_cmd : 2, 0, master, slave_right;
    mirror_r_hw_st : 4, 0, slave_right, master;
    mirror_r_l_st : 1, 0, slave_right, master;
    mirror_r_resp_err : 1, 0, slave_right, master;
    mirror_r_tmp : 8, 0, slave_right, master;
    mirror_l_t_cmd : 4, 0, master, slave_left;
    mirror_l_l_cmd : 2, 0, master, slave_left;
    mirror_l_hw_st : 4, 0, slave_left, master;
    mirror_l_l_st : 1, 0, slave_left, master;
    mirror_l_resp_err : 1, 0, slave_left, master;
    mirror_l_tmp : 8, 0, slave_left, master;
}

Diagnostic_signals
{
    MasterReqB0: 8, 0;
    MasterReqB1: 8, 0;
    MasterReqB2: 8, 0;
    MasterReqB3: 8, 0;
    MasterReqB4: 8, 0;
    MasterReqB5: 8, 0;
    MasterReqB6: 8, 0;
    MasterReqB7: 8, 0;
    SlaveRespB0: 8, 0;
    SlaveRespB1: 8, 0;
    SlaveRespB2: 8, 0;
    SlaveRespB3: 8, 0;
    SlaveRespB4: 8, 0;
    SlaveRespB5: 8, 0;
    SlaveRespB6: 8, 0;
    SlaveRespB7: 8, 0;
}
```

```
Frames
{
    mirror_r_cmd : 0x30, master, 1
    {
        mirror_r_t_cmd, 0;
        mirror_r_l_cmd, 4;
    }
    mirror_r_status : 0x31, slave_right, 1
    {
        mirror_r_hw_st, 0;
        mirror_r_l_st , 4;
        mirror_r_resp_err, 7;
    }
    mirror_r_temp : 0x32, slave_right, 1
    {
        mirror_r_tmp, 0;
    }
    mirror_l_cmd : 0x33, master, 1
    {
        mirror_l_t_cmd, 0;
        mirror_l_l_cmd, 4;
    }
    mirror_l_status : 0x34, slave_left, 1
    {
        mirror_l_hw_st, 0;
        mirror_l_l_st , 4;
        mirror_l_resp_err, 7;
    }
    mirror_l_temp : 0x35, slave_left, 1
    {
        mirror_l_tmp, 0;
    }
}

Diagnostic_frames
{
    MasterReq: 60
    {
        MasterReqB0, 0;
        MasterReqB1, 8;
        MasterReqB2, 16;
        MasterReqB3, 24;
        MasterReqB4, 32;
        MasterReqB5, 40;
        MasterReqB6, 48;
        MasterReqB7, 56;
    }
    SlaveResp: 61
    {
        SlaveRespB0, 0;
        SlaveRespB1, 8;
        SlaveRespB2, 16;
        SlaveRespB3, 24;
        SlaveRespB4, 32;
        SlaveRespB5, 40;
        SlaveRespB6, 48;
        SlaveRespB7, 56;
    }
}
```

```
Node_attributes
{
    slave_right
    {
        LIN_protocol = "2.0";
        configured_NAD = 0x44;
        product_id = 0x0004, 0xf000, 1;
        response_error = mirror_r_resp_err;
        P2_min = 20 ms;
        ST_min = 20 ms;
        configurable_frames
           {
           mirror_r_cmd  = 0xf001;
           mirror_r_status = 0xf002;
           mirror_r_temp = 0xf003;
           }
    }
    slave_left
    {
        LIN_protocol = "2.0";
        configured_NAD = 0x45;
        product_id = 0x0004, 0xff00, 1;
        response_error = mirror_l_resp_err;
        P2_min = 20 ms;
        ST_min = 20 ms;
        configurable_frames
           {
           mirror_l_cmd  = 0xff01;
           mirror_l_status = 0xff02;
           mirror_l_temp = 0xff03;
           }
    }
}

Schedule_tables
{
    net_configuration
    {
        AssignFrameId {slave_right, mirror_r_cmd}      delay 20 ms;
        SlaveResp                                      delay 20 ms;
        AssignFrameId {slave_right, mirror_r_status}   delay 20 ms;
        SlaveResp                                      delay 20 ms;
        AssignFrameId {slave_right, mirror_r_temp}     delay 20 ms;
        SlaveResp                                      delay 20 ms;
        AssignFrameId {slave_left, mirror_l_cmd}       delay 20 ms;
        SlaveResp                                      delay 20 ms;
        AssignFrameId {slave_left, mirror_l_status}    delay 20 ms;
        SlaveResp                                      delay 20 ms;
        AssignFrameId {slave_left, mirror_l_temp}      delay 20 ms;
        SlaveResp                                      delay 20 ms;
    }
    run_mode
    {
        mirror_r_cmd      delay 20 ms;
        mirror_r_status   delay 20 ms;
        mirror_r_temp     delay 20 ms;
        mirror_l_cmd      delay 20 ms;
        mirror_l_status   delay 20 ms;
        mirror_l_temp     delay 20 ms;
    }
}
```

slave_left.prv

```
/*****************************************************************************
*
* Freescale Semiconductor Inc. 2004 All rights reserved
*
*****************************************************************************
*
* THIS SOFTWARE IS PROVIDED BY FREESCALE SEMICONDUCTOR "AS IS" AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED.  IN NO EVENT SHALL FREESCALE SEMICONDUCTOR OR ITS CONTRIBUTORS
* BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*
*****************************************************************************
*
* Author : rc574c
*
*****************************************************************************/

LIN_private_file;
LIN_protocol_version = "2.0";
LIN_language_version = "2.0";
concurrency_safety   = LTP;

network "Mirror_net"
{
    node slave_left;
    file "Mirror_net.Ldf";
    original_NAD = 0x45;
}

flag mirror_l_t_cmd_new latches signal mirror_l_t_cmd;
flag mirror_l_l_cmd_new latches signal mirror_l_l_cmd;

interface "i1"
{
    connects to Mirror_net;
    [include "uart.cfg"]
}
```

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

AN2885
Rev. 0, 11/2004

freescale™
semiconductor