

System Outline

BLDC motors are gaining popularity in automotive applications, primarily for HVAC and engine cooling fans. Compared to the DC brush motor, the BLDC one uses electronic commutation instead of a mechanical commutator, leading to higher reliability and higher efficiency of whole system. Also, as the BLDC motor rotor generates the rotor's magnetic flux, the further efficiency of the electromechanical conversion is achieved.

System Outline

This document assumes a basic knowledge of implementing LIN 2.0 connectivity on Freescale MCUs as described in AN2767; only the key connectivity features for the LIN-enabled BLDC motor are described here. It is recommended to first study AN2767 and other valuable resources provided in the [“References”](#) section.

Cluster Topology

The LIN-enabled BLDC engine fan cluster consists of two nodes:

- **Master node** — provides the cluster with the required fan speed information together with run/stop commands and error tracking.
- **Slave node** — handles the BLDC motor control and provides the cluster with actual fan speed information together with the fan run/error status.

Message Strategy

The message strategy as designed (the master node sends the duty cycle and the slave node provides the actual motor speed) enables users to reduce the influence of the BLDC motor used and the dependency of the duty cycle on the speed.

The message strategy of the BLDC engine fan LIN cluster can be seen in [Table 1](#).



Table 1. LIN-Enabled BLDC Engine Fan Messaging Strategy

Node Name for Signal Provider	ID [0..5]	LIN Frame ID	Frame Name	Frame Description	Frame Size (Bytes)	Signal Name	Signal Description	Raw Value Range	Which nodes are reading this response data?		Comments
									master	slave	
master	0x01	0x0010	master_info	Contains information from the master node	2	master_speed	Required speed of the BLDC fan	0-0x63		X	Carry information about the required duty cycle.
						master_fan_on	Run the BLDC fan	0 = Stop the BLDC fan 1 = Run the BLDC fan		X	
						master_reset_fan	Clear all BLDC fan warning/error flags	0 = Leave the flags unchanged 1 = Clear all warning/error flags		X	
master	0x02	0x0020	resolving	The resolving process has been finished	1	master_resolving_done	The resolving process has been finished	0 = Resolving has not been finished 1 = Resolving has been finished		X	
slave	0x05	0x0050	slave_info	Contains information about the actual fan speed and status	2	slave_speed	Actual BLDC fan speed	0-0xFF	X		In rounds per second.
						slave_dc_bus_overcurrent	An over current occurs on the BLDC fan	0 = The DC bus current is within limits 1 = An over current occurs	X		The over current flag is set if the DC bus current exceeds the limits consecutively 5 times.
						slave_dc_bus_overvoltage	An over voltage occurs on the BLDC fan	0 = The DC bus voltage is within limits 1 = An over voltage occurs	X		The over voltage flag is set if the DC bus voltage exceed the limits consecutively 10 times.
						slave_resp_error	Used for LIN diagnostics	N/A	X		

Hardware

Master Node

In addition to the diagnostic frames, the master node sends information about the required BLDC motor duty cycle (*master_speed*), the fan on/off command (*master_fan_on*), and the reset signal (*master_reset_fan*) used to clear the over-current and over-voltage flags of the slave node. The master node also sends the *master_resolving_done* signal, used for on-run connection detection feature of the cluster (see “[On-Run Connection Detection and Connection Loss-Catching](#)” section for more details).

Slave Node

The slave node provides the master node with the actual BLDC motor speed in rounds per second (*slave_speed*), and the over current (*slave_dc_bus_overcurrent*) and over voltage (*slave_dc_bus_overnoltage*) flags, used for reporting that the DC bus current/voltage exceeds the safe limits. Also, the slave node sends the *slave_resp_error* signal used for LIN diagnostic purposes.

Schedule Tables

There are two schedule tables defined in the application:

- *sch_conflict_resolving* — used for assigning the frame ID to the nodes and also utilized for on-run connection detection feature.
- *normal_mode* — runs in normal mode and transfers the information from the master node as well as from the slave one.

Hardware

The slave node hardware concept can be seen in [Figure 2](#).

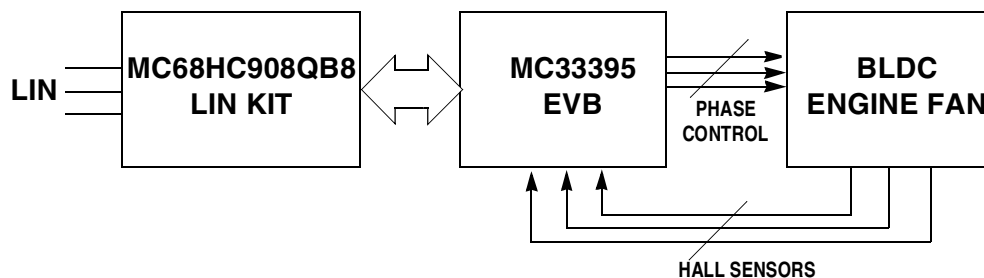


Figure 2. Slave Node Hardware Concept

The slave node consists of three main parts:

- MC68HC908QB8 LIN kit board.
- MC33395 evaluation board, as an example of the Freescale three phase gate driver portfolio (see “[References](#)”).
- The W3G300-EQ22-90 BLDC engine fan, as an example of the EBM-Papst fan production line (see “[References](#)” for more details).

MC33395 Evaluation Board (EVB)

The MC33395 EVB (see “References” for more details) is used as the power side of the design. Using the EVB allows the developer to concentrate on the MC control routines as well as the communication part of the project without the need to focus on the hardware development.

The MC33395 (see “References”) is an example of the Freescale three-phase gate driver IC portfolio. The key features of MC33395 are as follows:

- Drives six N-channel low $R_{DS(ON)}$ power MOSFETs
- Built-in charge pump circuitry
- Built-in current sense comparator and output drive current limiting
- Built-in PWM mode control logic
- Built-in circuit protection
- Designed for fractional to integral HP BLDC motors

The MC33395 simplified internal block diagram is shown in Figure 3:

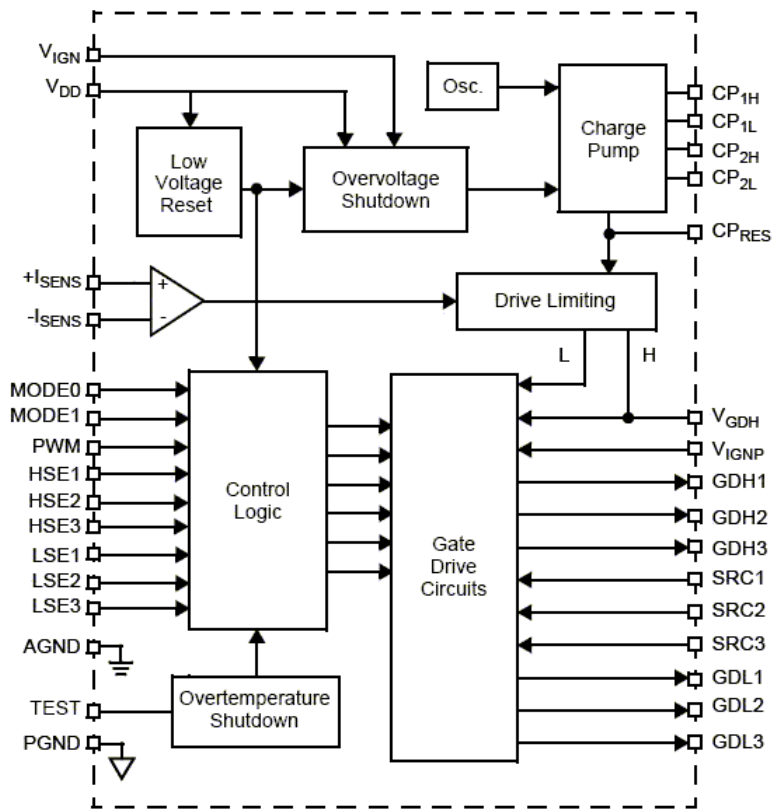


Figure 3. MC33395 Simplified Internal Block Diagram

Hardware

In addition to MC33395 embedded features, the MC33395 EVB includes the following blocks:

- Phase current measurement circuits
- DC bus voltage measurement circuit
- Back EMF and zero-crossing circuits to enable sensor less BLDC motor management
- Brake circuit useful to dissipate regenerative motor energy during active deceleration periods

The MC33395 EVB is connected to the slave MCU via a 40-pin ribbon cable. The signals used can be found in [Table 2](#):

Table 2: MC33395 EVB Signals

Pin No.	MC33395 EVB Signal Name	QB8 LIN Kit Board Pin	Description
1	PWM_AT	PTB0	High side switch of phase A
3	PWM_AB	PTB3	Low side switch of phase A
5	PWM_BT	PTB1	High side switch of phase B
7	PWM_BB	PTB6	Low side switch of phase B
9	PWM_CT	PTB2	High side switch of phase C
11	PWM_CB	PTB7	Low side switch of phase C
2,4,6,8,10	Shielding_D	GND	Shielding
12	GND	GND	Digital power supply ground
17,18	GND_A	GND	Analog power supply ground
21	V_sense_DCB	PTA5	DC bus voltage
22	I_reconst_DCB	PTA4	Sum of phase current
31	PWM_ctrl	PTA0	PWM control

Description of the MC33395 EVB jumpers and switches positions can be seen in [Table 3](#):

Table 3: MC33395 EVB Settings

Jumper/Switch	Position	Description
JP2	– (minus)	Low state (logic 0)
JP3	– (minus)	
JP5	uni	Controller board gives 1 PWM signal

MC68HC908QB8 LIN Kit Board

The MC68HC908QB8 LIN kit board is used as the controller board of the slave unit. Because the standard QB8 LIN kit board occupies some of the MCU pins needed for BLDC and MC33395 EVB connection, the board itself must be modified:

- The bonding between the PTB7 MCU port and MC33661 LIN interface enable pin (EN) must be disconnected. The R16 resistor must remain connected to the MC33661.

The simplified MC68HC908QB8 LIN kit board circuit after modification can be seen in [Figure 4](#):

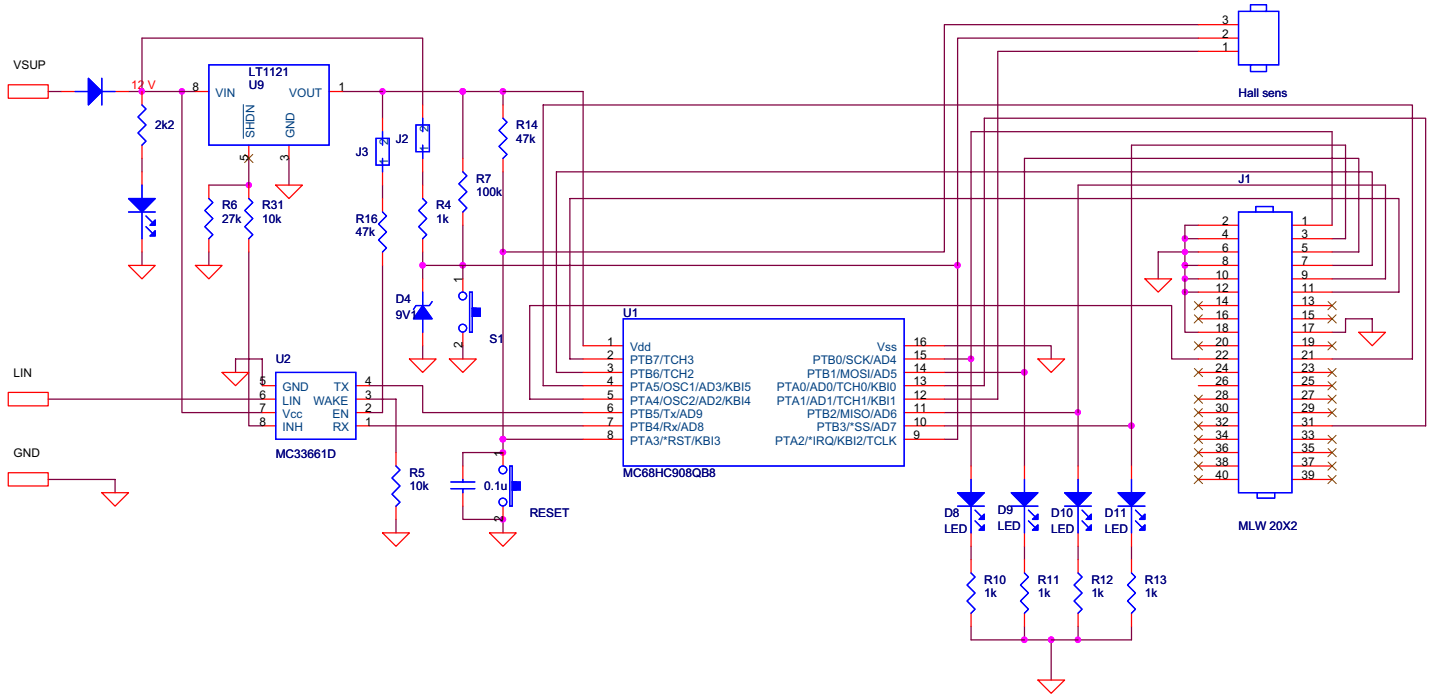


Figure 4: Simplified MC68HC908QB8 LIN Kit Board Circuit (Modified)

Except for the pins used for the MC33395 EVB connection (see [Table 2](#)), PTA1, PTA2, and PTA3 MCU port pins are used for connecting the BLDC motor Hall sensors. Because the PTB7 MCU port pin is required for the BLDC motor control, the enable pin (EN) of the MC33661 LIN interface cannot be used to control the switchable voltage regulator via the INH output pin of MC33661 LIN physical interface.

NOTE

This hardware concept is used for demonstration/development purposes. Freescale Semiconductor offers highly integrated IDC components, which combine the MCU, analog power component, and an incorporated LIN physical interface. This single IDC component can achieve the combined functionality of the EVB and LIN kit boards. Please visit www.freescale.com/analog for more details.

Software

The LIN-enabled BLDC engine fan software consists of the following files (please note, that only key files are listed here):

- *board.h* — contains definitions of the ports and pins used for connection with MC33395 EVB as well as with the BLDC motor Hall sensors; also contains the BLDC-motor-specific constants (maximum speed, commutation vectors, maximum speed step, etc.); and the bus voltage and current limits.
- *main.h* — includes the *main.c* functions prototypes and definition of the BLDC status structure.
- *main.c* — consists of the main infinite loop and all related functions, please refer to the [“Main Program Structure”](#) section for more details.
- *pwm.h* — includes the *pwm.c* functions prototypes and definition of the PWM maximum speed and initial duty cycle.
- *pwm.c* — contains the PWM initialization function.
- *t0ch_isr.c* — handles the timer channel 0 output compare ISR; this ISR is enabled only if deceleration of the BLDC is required.
- *t1ch_isr.c* — contains the timer channel 1 input capture ISR; this interrupt service routine is used for BLDC speed measurement.
- *tof_isr.c* — handles the timer overflow ISR; This is the main routine of the BLDC commutation. This routine is also used for acceleration of the BLDC fan.
- *esci_rx_isr.c* — the ESCI Rx ISR handler is covered by this file.

Software Considerations

The LIN-enabled BLDC engine fan software is written in C using the Metrowerks CodeWarrior for HC08 v 3.1. For the LIN communication, the Volcano LIN Target Package is used. At time of development, the Volcano LTP for MC68HC908QB8 MCU was not available. However, because the ESCI module of the MC68HC908EY16 is the same as on the MC68HC908QB8 device, the LTP20_3_1_12 for MC68HC908EY16 is used for LIN communication.

Main Program Structure

The main program flow chart of the LIN-enabled BLDC engine fan software can be seen in [Figure 5](#):

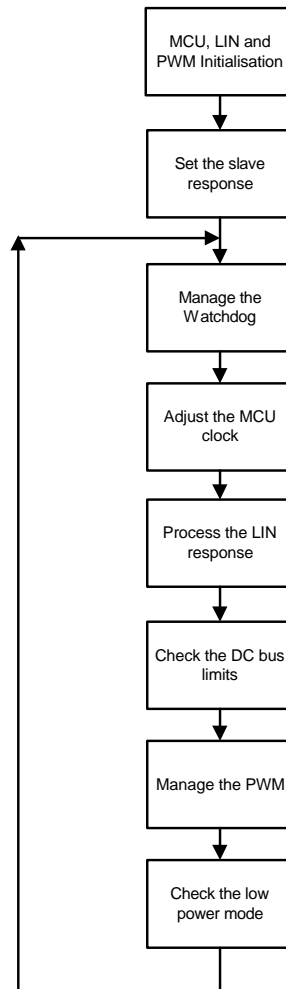


Figure 5: Main Program Flow Chart

As shown in [Figure 5](#), after startup initialization, the main infinite loop is processed. It consists of six sub tasks:

- **Manage the Watchdog** — takes care of the COP module, which enables the software to recover from runaway code.
- **Adjust MCU clock** — trims the MCU clock according to the synchronization field of received LIN messages.
- **Process the LIN response** — sends and receives messages to/from the master node and calculates the new BLDC motor speed. Also handles the limitation of BLDC motor speed change. The flow chart of this subtask can be found in [Figure 6](#):

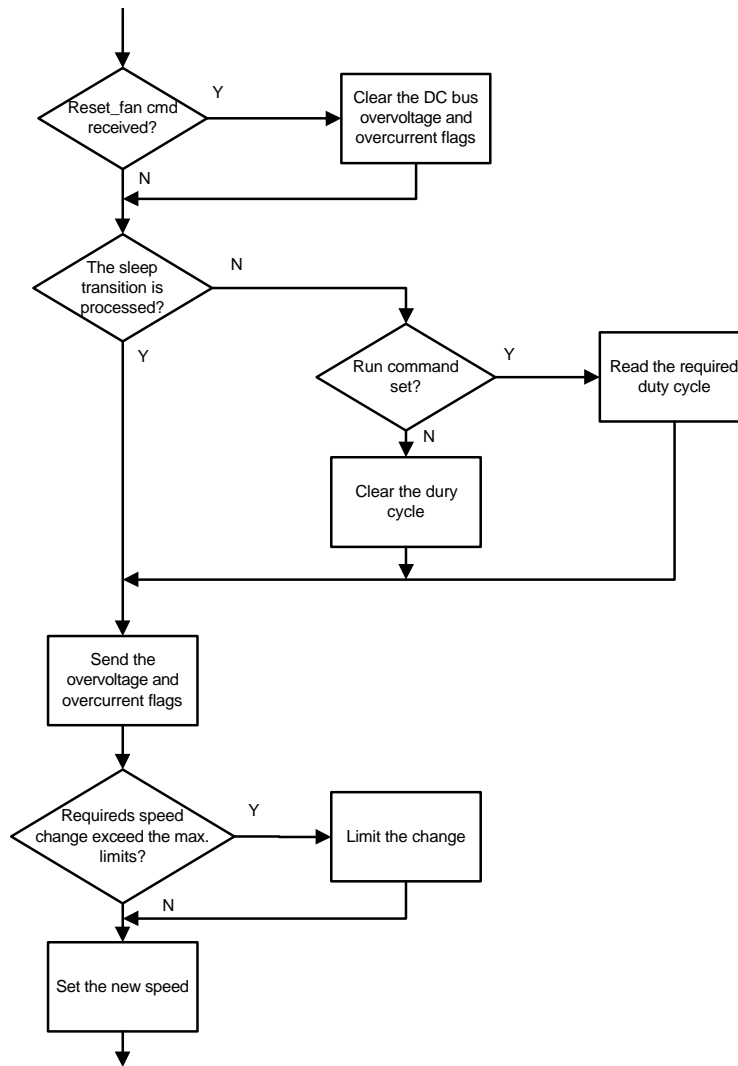


Figure 6: Process the LIN Response Subtask Flow Chart

- **Check the DC bus limits** — subtask checks the DC bus voltage and phases current to protect the MC33395 EVB against over current and over voltage harm. The protection is designed as two steps.
 - Whenever the current or voltage exceeds the given limits, the BLDC motor phases are temporarily disabled.
 - If the limits are broken 5 times consecutively (for current) or 10 times (for voltage), the BLDC motor is stopped and the fault is reported to the master node.

It is the responsibility of the master node to take appropriate steps to handle the reported situation, clear the over current and over voltage fault flags, and possibly run the motor again. Because the MC33395 EVB also contains over current protection, the software limits are tighter than the hardware limits.

The flow chart of these protections can be seen in [Figure 7](#).

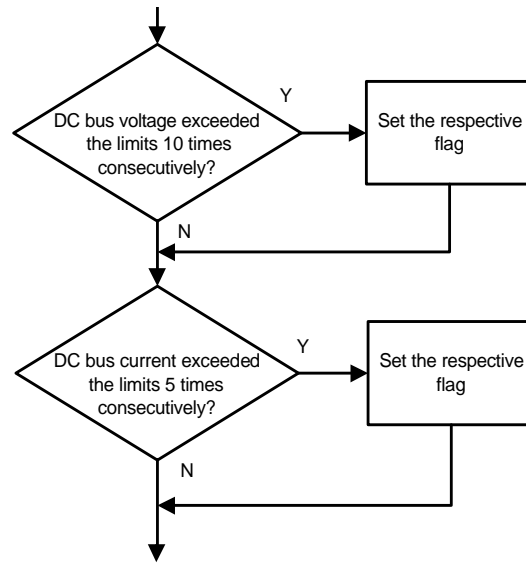


Figure 7: Check the DC Bus Limits Subtask Flow Chart

- **Manage the PWM** — used for managing the transitions from 100% and from 0% duty cycles, as well as for management of the BLDC speed decrease request. The flow chart of this subtask is shown in [Figure 8](#)

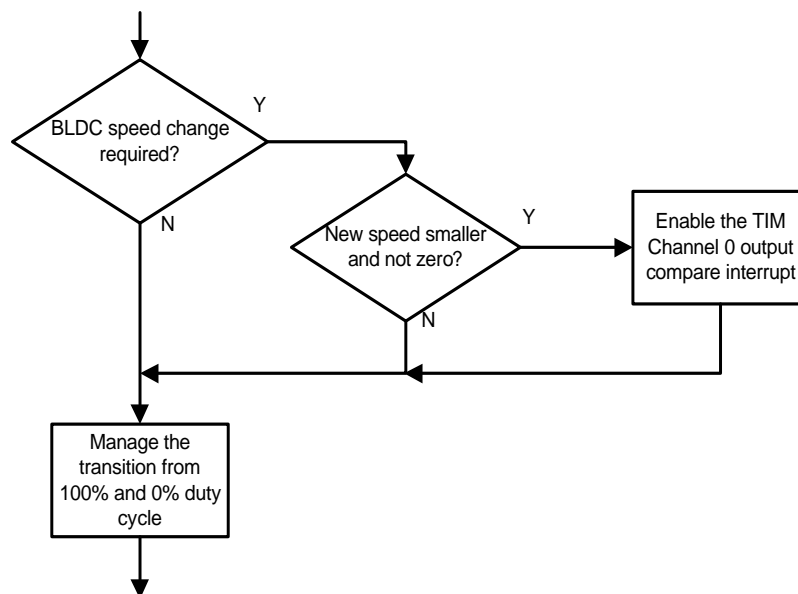


Figure 8: Manage the PWM Subtask Flow Chart

- Check the low power mode** — is the final part of the main loop in which the examination of master message correct reception is done. If there is no proper communication with the master node for 4 seconds¹, the go to sleep request flag (*sleep_fan*) is set. The same action is processed if the LIN *goto_sleep* command is received from the master node via the LIN bus. Setting the *sleep_fan* flag disables the BLDC motor phases and the motor starts free wheeling. As soon as the BLDC motor speed falls to zero and the *sleep_fan* is set, PWM generation is disabled and the fan is stopped. The flow chart of this subtask can be found in [Figure 9](#):

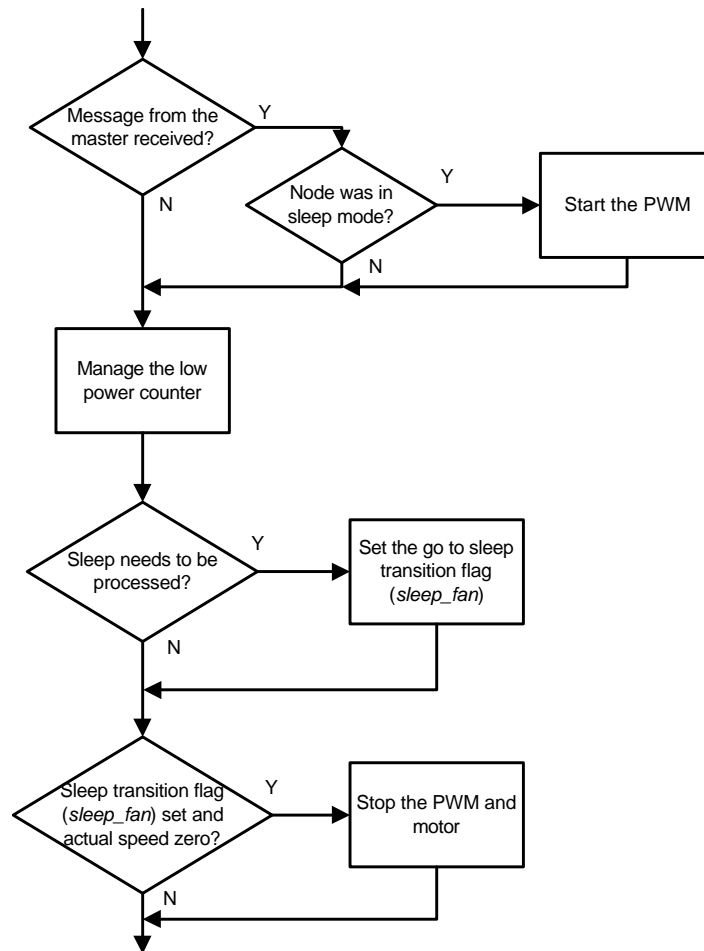


Figure 9. Check the Low Power Mode Section Flow Chart

BLDC Phase Commutation

The BLDC phase commutation is driven according to the Hall sensors pattern to keep the angle between the rotor flux and stator flux near to 90°. The Hall sensors pattern is read on the TIM overflow; therefore, it runs on the PWM frequency. The appropriate output commutation vector is put on the phases in cases where no obstacle has been detected (i.e., run command from the master node received, no over current and no over voltage detected, no temporary shut down, and no go to sleep transition using the *sleep_fan* processed). The direction of the BLDC motor rotation can be changed by using the inverse commutation table.

1. The 4 seconds time margin is mandatory in LIN 2.0 specification.

The commutation table for clockwise rotation for BLDC motor used can be seen in [Table 4](#):

Table 4: Commutation Table for Clockwise Rotation

Vector No.	Hall Sensor			Motor Phase			Motor Phases High/Low Side Switch						Output vector
	Hall1	Hall2	Hall3	L1	L2	L3	L1-HS	L2-HS	L3-HS	L1-LS	L2-LS	L3-LS	
1	1	1	0	0	+Ucc	-Ucc	0	1	0	0	0	1	0x82
2	1	0	1	+Ucc	-Ucc	0	1	0	0	0	1	0	0x41
3	1	0	0	+Ucc	0	-Ucc	1	0	0	0	0	1	0x81
4	0	1	1	-Ucc	0	+Ucc	0	0	1	1	0	0	0x0C
5	0	1	0	-Ucc	+Ucc	0	0	1	0	1	0	0	0x0A
6	0	0	1	0	-Ucc	+Ucc	0	0	1	0	1	0	0x44

On-Run Connection Detection and Connection Loss-Catching

The on-run connection detection with connection loss-catching is based on a technique used in AN2767 (see [“References”](#)).

Each data frame ID must be assigned to the respective LIN 2.0 slave node before it can be used. At the beginning of LIN communication, the schedule table with ID assignment commands (often called the *conflict_resolving* schedule table) must be processed. This table is usually called once after a cluster power up or master node reset. This feature is used for on-run connection detection. At the end of this table, there is a *resolving* frame transfer. This frame is transferred only once and carries the *resolving_done* signal. If the slave node receives this flag, it assumes that the resolving process has been successfully completed and sends to the master an acknowledgment, using the *SlaveResponse* frame¹ filled with 0x80 data. On the other hand, if the slave node, after reset, detects that the *resolving_done* flag has not been set (and thus no resolving frame has been received), it sends the *SlaveResponse* frame filled with 0x81 data. Whenever the master node receives a *SlaveResponse* frame with the 0x81 content, it completes the actually used schedule table and processes the *conflict_resolving* schedule table, to re-initialize the LIN cluster to reflect the new devices on the bus.

The connection loss-catching is based on the *l_ifc_read_status()* status function of the LIN driver (see [“References”](#)). If the slave node detects no reception of data from the master node for 4 seconds, it perceives the connection with the master node to have been lost and processes the steps to enter the low power mode.

1. The *SlaveResponse* and *MasterReq* frames should be included in the schedule table used for normal data transfer mode to enable the diagnostic layer.

References

Speed Measurement and Speed Control

The software counter is used for BLDC fan speed measurement. This counter is incremented on each TIM overflow; therefore it runs on the PWM speed. The TIM channel 1 input capture feature is used to capture the time (using the TIM-overflow-based software counter) between two rising edges of Hall sensor 1. This time (counter value) is then used for speed calculation. Please note, that three pulses of Hall sensor are generated per one mechanical revolution, and thus the calculated speed must be divided by 3.

The BLDC fan speed is controlled in a closed loop by the PWM duty cycle using a maximum step limitation technique. If the master requests a speed that requires a change less than or equal to $\pm 15\%$ of the maximum speed, the requested speed will be applied immediately to the slave motor. If the requested speed requires a change greater than $\pm 15\%$, the actual speed will be equal to the original speed with the maximum $\pm 15\%$ incremental change.

References

See the following documents and Freescale Semiconductor's LIN resources page:
www.freescale.com/LIN

1. MC68HC908QB8 Data sheet,
Freescale document number: MC68HC908QB8
2. LIN Specification Package, Revision 2.0, 23 September 2003,
www.lin-subbus.org
3. LIN 2.0 Connectivity on Freescale 8/16-bit MCUs Using Volcano LTP,
Freescale Document number: AN2767
4. LIN Kits LIN Evaluation Boards,
Freescale document number: AN2573
5. MC33395 Data sheet,
Freescale document number: MC33395
6. Low Power BLDC Drive for Fans Using the MC68HC908QY4 Reference Design
Freescale document number: DRM046
7. www.ebmpapst.com
EBM-Papst web pages
8. 33395 Evaluation Motor Board Design Reference Manual
Freescale document number: DRM33395/D

This page is intentionally blank.

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.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

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>

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2005.