Window Lift and relay based DC motor control Reference Design Using the S12VR

Document Number: DRM160
Rev. 1.1, 08/2016

Contents

Section number **Title** **Page**

Chapter 1
Overview

Chapter 2
Hardware Description

Chapter 3
Firmware Description

Chapter 1
Overview

1.1 Overview

This reference design describes the design of a relay driven DC motor controller using a Hall sensor for position feedback. The application for this design is an automotive window lift but the final application can be used anywhere a DC motor relay driven control is needed.

The design exhibits the suitability and advantages of the MagniV S12VR microcontroller for relay driven motor control. It serves as an example of an anti-pinch application using the MagniV S12VR series of microcontrollers.

The MC9S12VR is an optimized automotive 16 bit microcontroller focused on low cost, high performance, and low pin count. This device integrates an S12 microcontroller with a LIN physical interface, a 5 V regulator system, and analog blocks to control other elements of the system which operate at vehicle battery level (relay drivers, high side driver outputs, wakeup inputs). The MC9S12VR is targeted at generic automotive applications requiring single node LIN communications. Typical examples of these applications include window lift modules, seat modules, sunroof modules, etc.

The following features make the MC9S12VR ideal for reducing system cost, bill of materials cost, development cost, and saving circuit board space:

- 1 MHz internal RC oscillator with +/-1.3% accuracy over rated temperature range
- 1 on-chip LIN physical layer transceiver fully compliant with the LIN 2.2 standard & SAE J2602-2 LIN standard
- On-chip voltage regulator for all internal voltages
- 2 protected low side outputs to drive inductive loads
- Up to 2 protected high side outputs
- 4 high voltage inputs with wakeup and readable internally on ADC
- Up to 210 mA high current outputs
- 20 mA high current output for Hall sensor supply
- Battery voltage sense with reverse battery protection

Other useful features of the MC9S12VR are:

- HCS12 CPU core
- 64 or 48 Kbyte on-chip flash with ECC
- 512 byte EEPROM with ECC
- 2 Kbyte on-chip SRAM
- Phase locked loop (IPLL) frequency multiplier with internal filter
- 4-20 MHz amplitude controlled pierce oscillator
- Internal COP (watchdog) module (with separate clock source)
- Timer module (TIM) supporting input/output channels that provide a range of 16-bit input capture, output compare, and counter (up to 4 channels)
- 10 bit resolution ADC with up to 6 channels available on external pins
- One SPI module
- One SCI supporting LIN
- One additional SCI
- Autonomous periodic interrupt
- Chip temperature sensor

These features substantially reduce the cost of developing and manufacturing applications such as:

- Automotive power window lift / sunroof with anti-pinch
- Any relay driven DC motors
- LIN slaves with space constraints

The following image depicts a block diagram of the MC9S12VR, the main component of the window lift reference design:

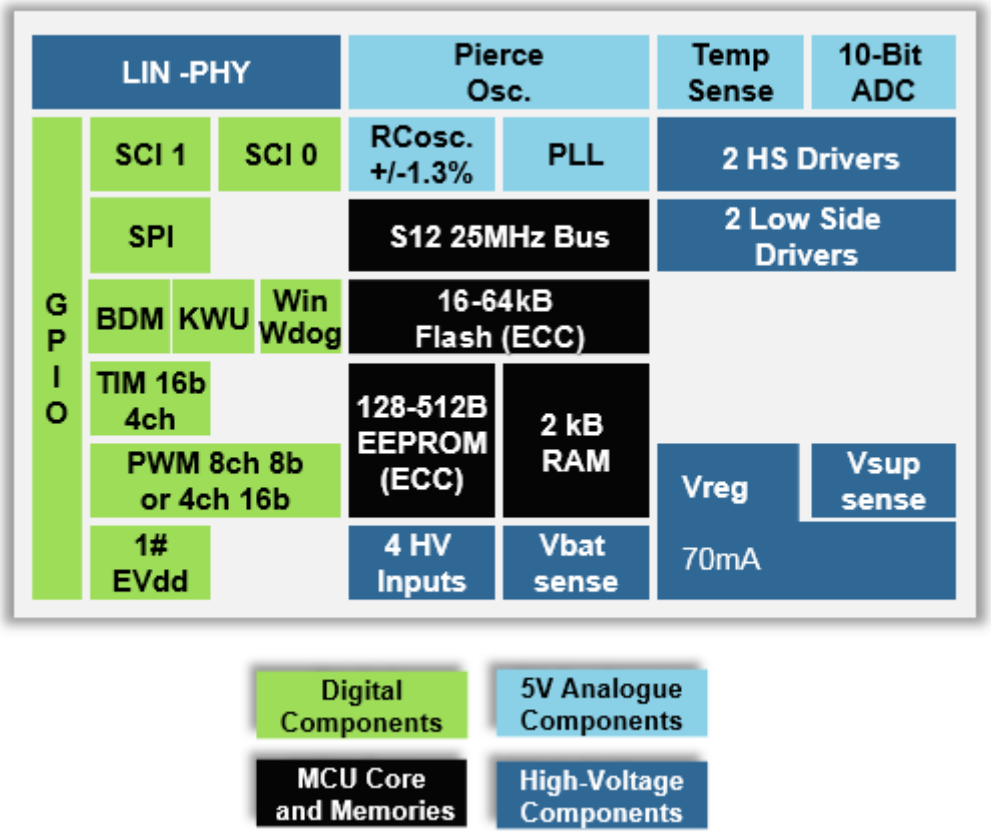

Figure 1-1. Block diagram

The following features are implemented in this reference design:

- Motor type: DC, 7 A typ. / 16 A max.
- Feedback: Hall encoder, voltage at motor terminals
- Communication protocol: LIN
- Hardware ports:
  - VBAD, GND, LIN
  - Motor up, motor down
  - Hall direction, Hall speed, Hall power, Hall ground
- PCB:
  - Two layer board, assembly on top side only
  - 2.4 x 1.9 inches (including demo switch and push buttons)
  - BOM parts = 40

Chapter 2
Hardware Description

2.1 Hardware description

This chapter explains how the hardware of the reference design was implemented. A detailed description of each module connection is described. The following block diagram illustrates all of the functions used at a high level:

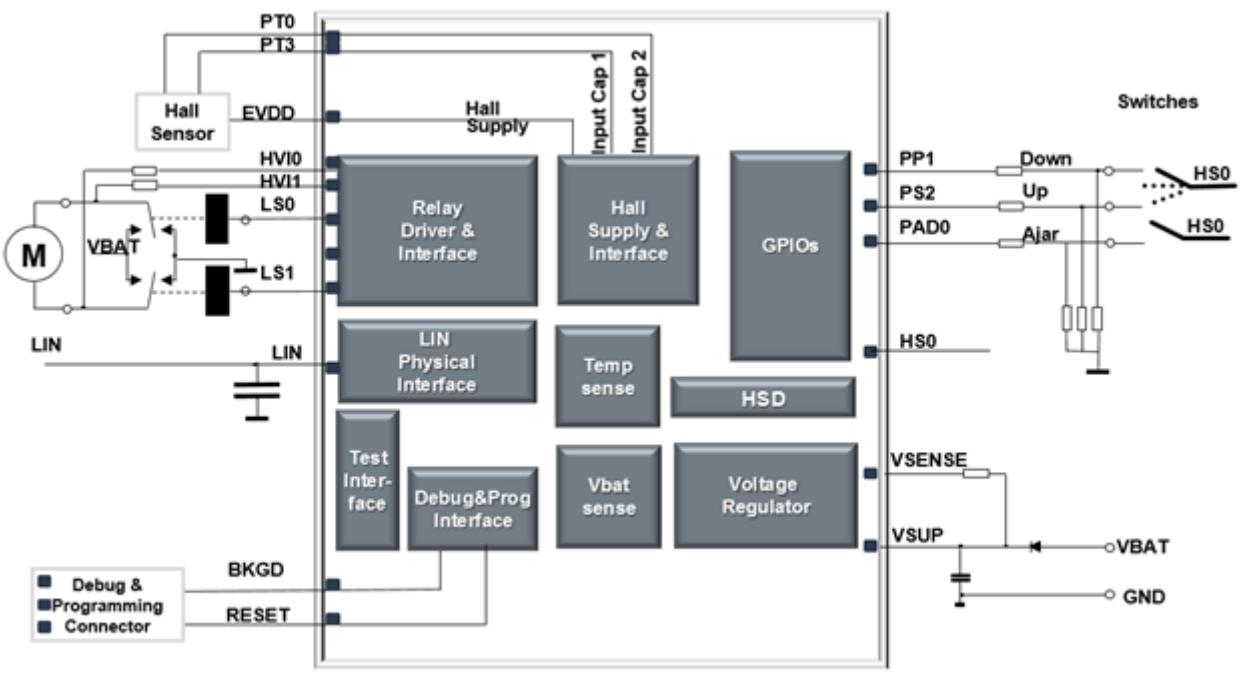

Figure 2-1. Window lift block diagram

The following paragraphs explain how the different pins and peripherals of the MC9S12VR are connected and used to provide different functions.

Hall sensor: The hall sensor acts as an encoder and outputs square signals that encode speed and direction information. These signals are measured with pins PT0 and PT3 used as input capture channels. The hall sensor is powered by the EVDD pin which is capable of supplying up to 20 mA @ 5 V.

DC motor: The DC motor is driven by a twin single pole double throw relay. This configuration allows the relay to also act as an H bridge allowing the MCU to polarize the motor in either direction. The two coils of the relay are driven by the low side driver 0 and 1 which can be connected directly to the relay coils as they integrate high current capability. They also integrate an internal active clamp that protects the device from coil discharge, without any need for external diodes. The voltage at the motor terminals is monitored by connecting the motor terminals to the high voltage inputs 0 and 1. These HVI inputs allow the ADC to sample this high voltage directly.

Debug: The BDM connector allows the user to reprogram and debug the board. It only requires two pins; RESET and BKGD.

VSUP: A low forward voltage diode protects the rest of the circuit from reverse battery conditions. Two 10 uF/50 V/1210 capacitors are used to decouple transients in the 100 KHz to 1 MHz range.

VSENSE: A 10 K resistor and a 6.8 nF/50 V/0402 capacitor make a low pass filter for stable battery measurements.

Motor actuator: The relay takes about 1.3 ms to open or close the terminals (max 5 ms). The relay is activated by 2 low side drivers. No diode is needed to protect from the relay coil discharge because the low side drivers already include an internal active clamp.

Switches: a three way switch is connected. This switch is for demo purposes as a real device may use a separate panel switch.

The following paragraphs describe how some of the internal modules of the S12VR map to this application and how they are used in the block diagram:

Relay driver & interface: This block uses Low Side Drivers (LSD) 0 and 1 to activate the relay coils. High voltage inputs (HVI) 0 and 1 are used to monitor the motor voltage.

LIN Physical Interface: The S12VR internal LINPHY and the SCI modules are used to implement LIN bus communication without any external components.

Debug & Prog Interface: The BDM module is in charge of implementing debugger tool communication and device reprogramming.

Hall supply and interface: Two timer channels in input capture mode implement the Hall encoder interface for measuring motor speed and position. The hall sensor is powered by the high current capability of the EVDD pin.

Voltage regulator: The S12VR has an integrated automotive voltage regulator that requires only reverse battery protection to connect directly to a 12 V automotive battery.

GPIOs: General purpose pins as inputs to read external switch interface.

The following pages detail the electrical schematics of the window lift reference design.

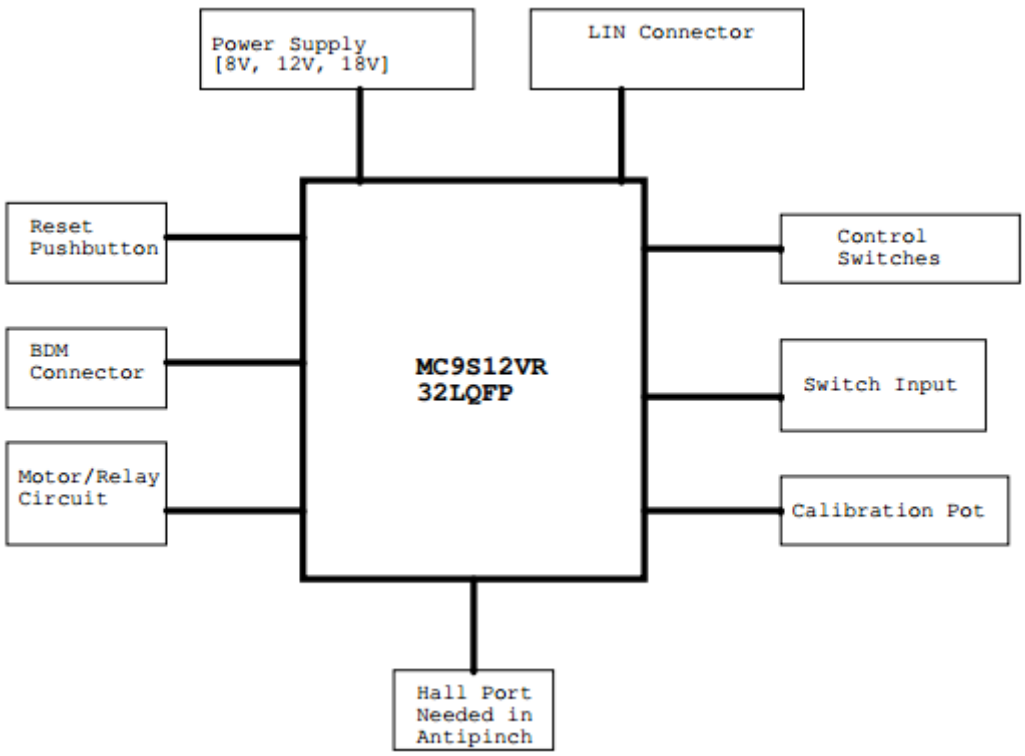

Figure 2-2. Block diagram

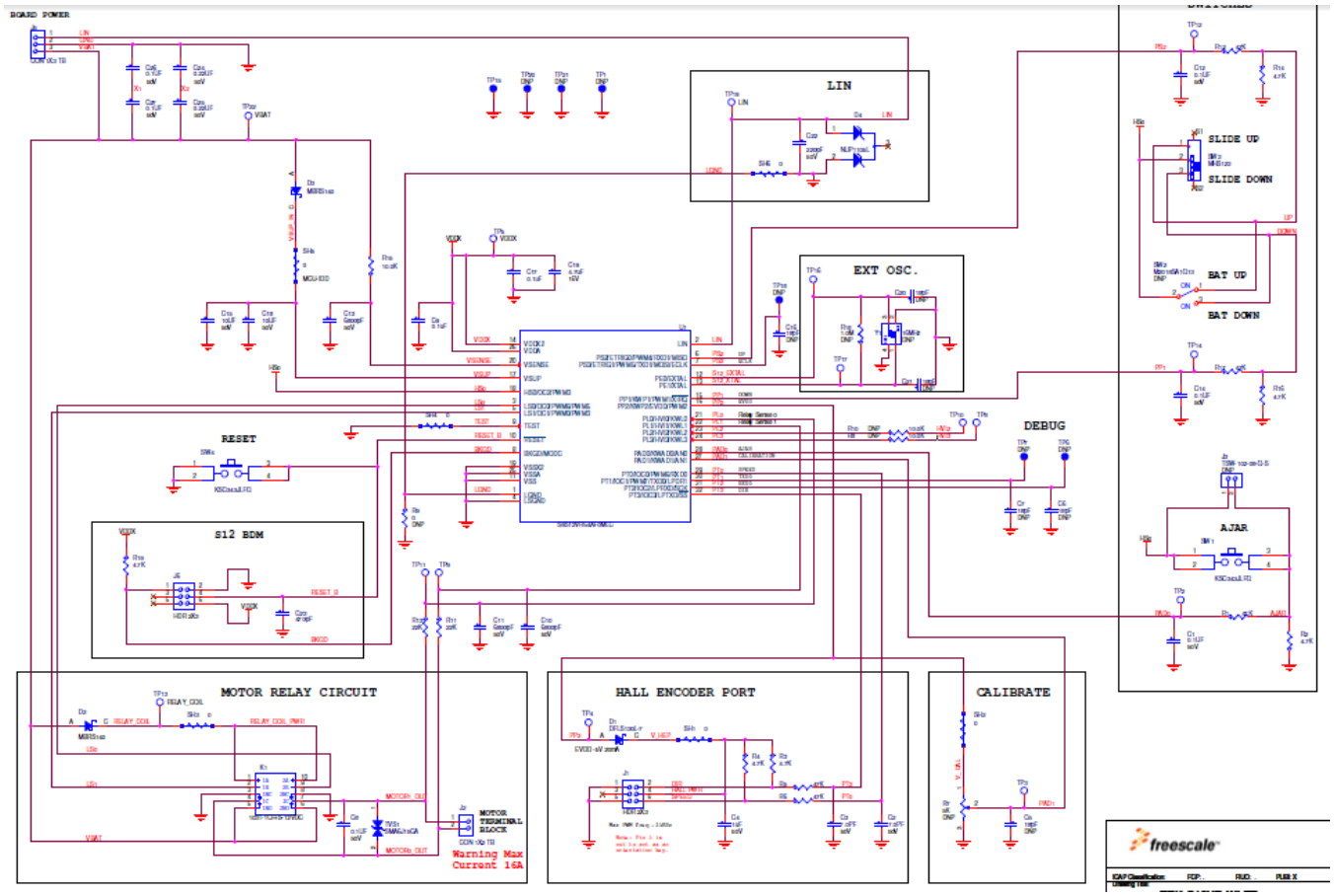

Figure 2-3. Schematics

Chapter 3
Firmware Description

3.1 Firmware description

There are two Codewarrior software projects provided with this reference design.

- **Up-down-no-anti-pinch**: This is a very basic software project that allows immediate use of the hardware by implementing a simple up down control with the relay and the input switch. Any DC motor can be connected to the relay terminals to test the board.
- **WindowLift.mcp**: This is a fully functioning window lift controller complete with anti-pinch algorithm.

The features implemented in the WindowLift.mcp Codewarrior project include:

- **Speed and direction:** tracks the window position counting the number of pulses provided by the encoder and the speed of the window measuring the pulse width of the encoder signal.
- **Voltage at the motor:** Reads the voltage at the motor up/down terminals.
- **Expected speed at 12 V:** Uses the Speed and Direction subsystem to provide an expected speed at 12 V. It depends on a calibration algorithm that stores the window expected speed at each position.
- **Pinch detection:** Uses the information from the subsystems above to determine the deviation from the computed speed and triggers the self-reversal subsystem when a threshold is passed.
- **Self-Reversal:** When triggered, it controls the self-reversal of the window. It either stops after a certain amount of distance is reversed or after the initial window position was reached.

The firmware that implements the window lift application with the anti-pinch feature is divided into several modules. These modules are illustrated in the following architecture diagram

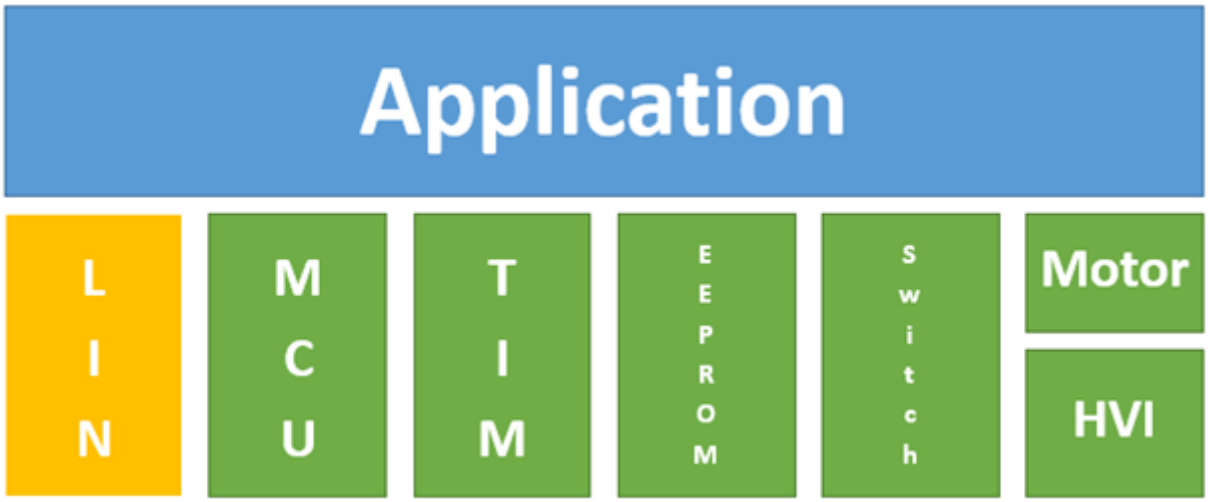

Figure 3-1. Architecture diagram

There are 5 different modules involved on application: MCU, TIM, EEPROM, Switch and Motor. And one driver: LIN

Each module interacts with certain hardware peripherals to configure them and provide services to the application. The modules will be described below:

- **MCU**: Configures the bus clock to 24 MHz and enables the LIN physical layer of the device.
- **TIM**: Configures the timer of the device. This timer uses the channel 0 as an input capture which is routed to the motor encoder. The timer measures the width of the encoder pulses in order to detect if the motor is being obstructed while in movement, it also counts the number of pulses to estimate the window position.
- **EEPROM**: Configures the Flash module of the device. Includes functions to read, erase and writes into the EEPROM memory region.
- **Switches**: Configures the ports routed to the switch as digital inputs. Detects when switch is being pressed up, down or not being pressed.
- **Motor/HVI**: Configures the low side drivers and the high voltage inputs. Provide functions to stop and move up or down the motor. High voltage inputs provide feedback if the motor is moving correctly. This module also enables EVDD to provide voltage to the motor's encoder.
- **LIN**: This application includes the NXP's LIN driver in order to configure the device as a slave and receive the up and down commands from a master device.

The application contains software routines that use the lower level modules in order to complete the main tasks. The application includes the following routines:

- **Self-calibration**: The first time the software is executed it drives the window up until it reaches stall condition, then the motor is driven down until it reaches stall again.

By storing the window length in encoder pulses the application can distinguish between a motor stalled by reaching the end of the path and a motor stalled by an object in the way. The motor is driven up again in order to get the maximum pulse width of the window when moving freely. This value is used as a threshold value to detect when an obstacle is detected when the window is moving. Calibration values are stored in the EEPROM.

- **Self-reversal**: When triggered, it controls the self-reversal of the window. It either stops after a certain amount of distance is reversed or after the initial window position was reached
- **Speed and direction**: tracks the window position and the measured period of the speed signal.
- **Power down**: Saves the window position before going into a low power mode.
- **LIN message detection**: Device configured as slave and receives signal from master when a button us pressed to move up or down the window.

Finally, the application uses all of these routines in conjunction to provide the direction control feature and the anti-pinch feature for a complete window lift application.

The application behavior is described by the following flow diagram:

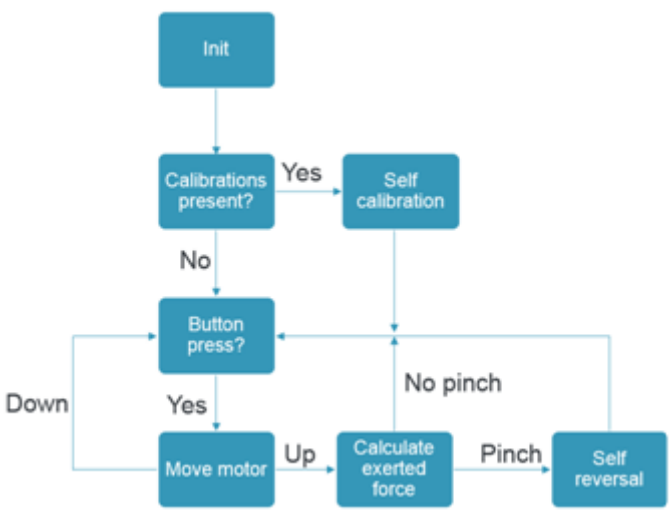

Figure 3-2. Flow diagram

As it can be seen from the flow diagram, the firmware behavior at a high level is pretty simple. The process can be described as follows:

1. MCU powers on. It initializes all the clocks, peripherals, and GPIO needed.
2. Software checks if calibrations are stored in the EEPROM. If they are not present then self-calibration will be performed.

NOTE

Self-calibration will move the motor down until the window encounters a stall condition (window completely open) and then move the window up until the window encounters a stall again (window completely closed). This process will be performed two times. User must be extremely careful not to obstruct the window during self-calibrate because the anti-pinch safety algorithm is not active during this time.

3. MCU waits until a button if pressed. If the down button is pressed no anti pinch is needed and the window will be moved down until the user stops pressing the down button or if the window reaches the end of the rail. If the up button is pressed the motor moves the window up while the MCU is continuously calculating the exerted force. Window can be also moved by a LIN message send from the master. If the window reaches the end of the rail then it is stopped. If it has not reached the end of the line and an unusually high force is being exerted, then the self-reversal activates, which drives the window down for a couple of seconds in response to a pinch event.

Calibration Process:

It is essential for this application to calibrate the first time is used. Depending on the window mechanism, its height and the motor used the number of encoder steps required to fully close or open the window may vary. Therefore, it is important to include a calibration routine on the algorithm. In this application the window goes up and down two times. The first time the size of the window in encoder steps is calculated by driving the motor down until it reaches stall condition and then going up counting the encoder steps until it reaches stall condition again. This value allows the application to detect when the motor is being obstructed by one the window's limits or by and object. In the application the stall condition when the motor is going up or down could be modified depending on the motor is being used:

#define MAX_STEP_TIME_UP 20000

#define MAX_STEP_TIME_DOWN 12000

The 20000 and 12000 represent the value of the timer counter. To convert it into time:

$$Timer\ Frequency = \frac{Fbus}{Prescaler} = \frac{24000000}{8} = 3000000\ Hz$$

$$Timer\ Period = \frac{1}{Timer\ Frequency} = \frac{1}{3\ MHz} = 0.333\ \mu s$$

$$MAX\_STEP\_TIME\_UP = (20000)(Timer\ Period) = 6.6\ ms$$

$$MAX\_STEP\_TIME\_DOWN = (12000)(Timer\ Period) = 3.9\ ms$$

This values are used to detect the window's limits and may vary depending on the motor. If the pulse width read by the timer is greater than this established values the algorithm will stop the motor as it is consider a stall condition. It is important to consider that when the window is being lifted the weight of the glass oppose to the motor movement. Therefore, the pulse width will be greater.

To complete the calibration the motor moves up and down a second time. This time the pulse width is analyzed while the motor is moving up as it changes during the trajectory as the next figure illustrates:

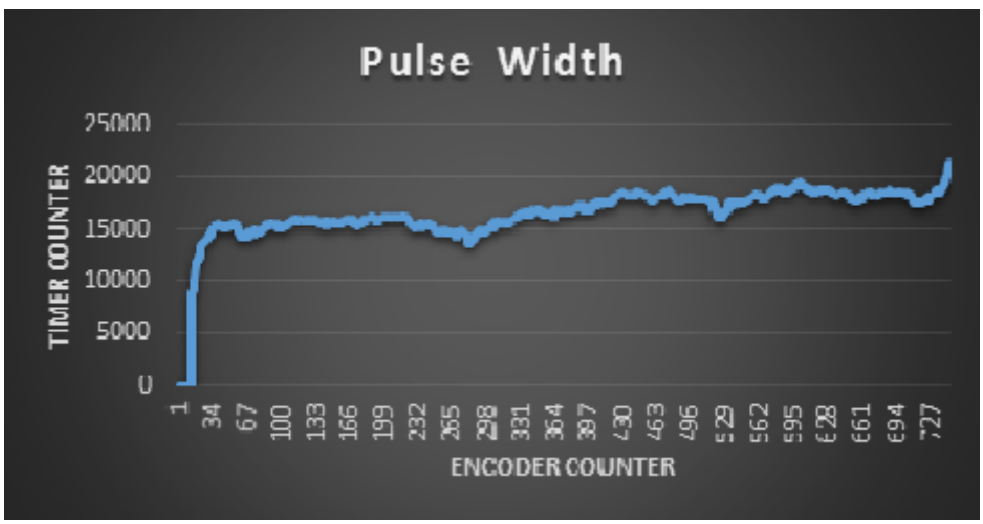

Figure 3-3. Pulse width along the window without obstacle

Due to the effects of the glass weight the pulse width increments as the window gets close to the top limit. To get a more precise threshold to detect if an obstacle is presented or not. The calibration algorithm store the maximum pulse width use it to detect if an object is blocking the window while it is going up. This values are stored in the EEPROM. If the device is turned off, the application reads the EEPROM, if calibration values are already stored no calibration routine is performed.

Pitch Detection:

As the window is moving up the software is continuously calculating the encoder pulse width and comparing it to the threshold value obtained in the calibration. If the pulse width surpasses the threshold value, the algorithm will perform an anti-pitch action and the window will move down a short distance. In the next figure it will be illustrated how the pulse width is affected when an object obstruct the windows movement.

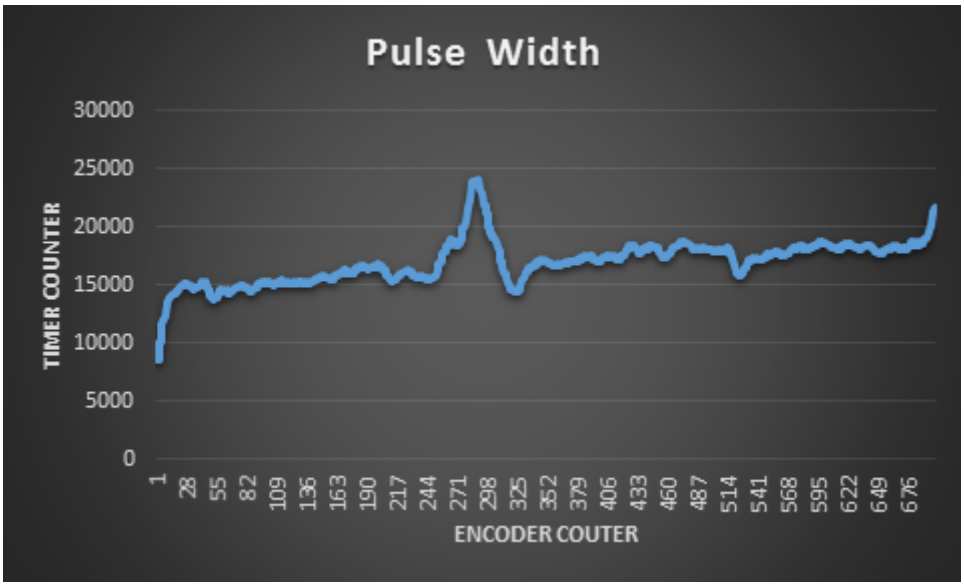

Figure 3-4. Pulse width along the window with obstacle

A spike could be notice when the movement of the window is blocked. This value surpasses the established threshold and anti-pith sequence will be started. The algorithm count the number of pulses and compares it to the size of the window. If the window movement is blocked by one of the windows limits the motor stops with no anti-pitch sequence.

LIN Configuration

LIN speed: 9600 bps

LIN signals:

- Button Down = 1
- Button Up = 2
- No button pressed = 2

In this application an S12ZVML128 was configured as master and depending on the button pressed send the up or down signal.

NOTE

It is not purpose of this document to explain how to configure the NXP's LIN driver. For more information on the matter please visit the application note **AN5122** and www.nxp.com

Software CPU and memory utilization

This reference design provides a good notion of how much resources will be used when implementing a relay driven DC motor application with a software anti-pinch algorithm, such as a window lift, sunroof, or other DC motor application.

- **FLASH**: less than 4KB. This leaves up to 60 KB free for other uses such as calibrations, complex stacks, drivers, etc.
- **EEPROM**: less than 10 bytes. This leaves up to 402 bytes free to be used by the application for other purposes.
- **RAM**: less than 550 bytes. This leaves 1498 bytes of RAM free for other purposes.

First time operation

It is recommended that you first try the “up-down-no-anti-pinch” software. This is a very basic software project that allows immediate use of the hardware by implementing a simple up down control with the relay and the input switch. Any DC motor can be connected to the relay terminals to test the board.

NOTE

This software does not have any safety check for stall conditions. If the motor is left running for prolonged times in a stalled condition permanent damage may occur to the motor.

After the user tests his motor with the “up-down-no-anti-pinch” software, the anti-pinch software can be tested. This requires a motor with a hall sensor and the hall encoder interface to be connected to the board. The software “WindowLift.mcp” has the anti-pinch algorithm implemented.

The first time this software is programmed to the S12VR it will check if calibrations are stored in the EEPROM. Since the first time the software is programmed calibrations will not be present the self-calibration will be performed.

NOTE

Self-calibration will move the motor down until the window encounters a stall condition (window completely open) and then move the window up until the window encounters a stall again (window completely closed). This procedure would be repeated two times. User must be extremely careful not to obstruct the window during self-calibrate because the anti-pinch safety algorithm is not active during this time.

After self-calibration is performed the data is stored permanently on EEPROM and the window lift can be used and safely power cycled. The anti-pinch algorithm will detect if any object obstructs the path of the window and reverse the window for a few seconds if a pinch condition was detected.

Appendix A

A.1 Appendix

References:

- AN5122. Using NXP's LIN Driver with the MagniV Family.
- S12Z MagniV Window Lift and Relay-based DC Motor Control.

A.2 Revision History

Table A-1. Revision History

Rev. No.	Date	Substantial Changes
1.1	08/2016	• LIN communication added. • Added LIN network description. • Editorial updates. • Added description about calibration and anti-pinch detection processes.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number DRM160
Revision 1.1, 08/2016