AN2334
Application Note

Complete car door module

Introduction

A car door module typically consists of a rubber-sealed carrier, onto which a variety of door components such as the window lift mechanism, the wing mirror electric motor, the wiring harness, the loud speaker, the door latch inner release cable, locks and various switches are fitted, forming a “cassette”.

The trend in both Europe and the U.S. is to increase the complexity of the door module by adding more electronic features needed to drive all door loads and functions, with the possibility to connect the module to other car subsystems via standard automotive communication protocols (LIN, CAN).

Among many of these automotive subsystems, the connectivity via a single wire, decentralizing electronic modules, reduces the number of wires required and in turn reduces wiring harness weight, contributes significantly to overall vehicle weight reduction. This is of concern to auto manufacturers, who are constantly striving to reduce vehicle weight and to improve fuel efficiency.

In this document, an electronic module is presented that controls all the car door functions, including the window lift, all latching/locking operations, wing mirror movement, mirror turn indicator light, defroster and some lamps. To reduce the risk of bodily injury, especially to children, this module also includes an advanced trapping detection feature for the window lift motor, which stops the window if a body member such as a finger, a hand or an arm is introduced into the window climbing area during the window climbing process. A low-cost, high performance, antipinch algorithm based on monitoring the window motor driver current, has been developed.

Contents

List of tables

List of figures

1 Car door module description

This note describes a complete electronic module used to drive all loads in a car door, connected via a LIN bus to the dashboard and to other doors, and via a parallel port to a PC (for demonstration purposes). The block diagram in *Figure 1* shows the system configuration.

Figure 1. Door module block diagram

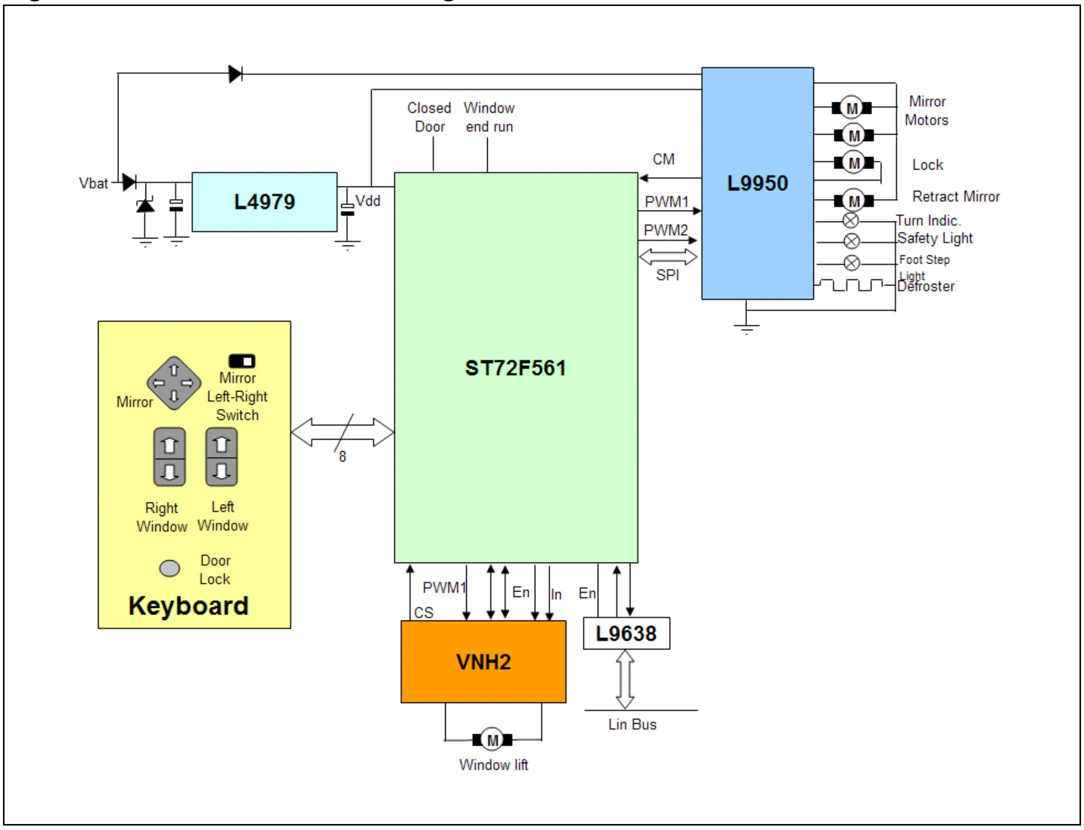


Table 1. Door module actuators

Actuators		P_{nom}	Working voltage	Load speed (Typ)	Load current (Typ)	Stall current
Window Lift	1 DC motor			>78 rpm	<2.5A	<20A
Door lock	1 DC motor					
Mirror axis control	2 DC motors				<2A	<10A
Mirror fold	1 DC motor		12 V			
Mirror defroster	1 grounded resistive load	100W				
Light bulbs	4 grounded resistive loads	5W				

The microcontroller is the ST72F561, a member of the ST7 microcontroller family, designed for mid-range applications with CAN (Controller Area Network) and LIN (Local Interconnect Network) interfaces. It is based on an industry standard 8-bit core, featuring an enhanced instruction set. The enhanced instruction set and addressing modes of the ST7 offer both power and flexibility to software developers, enabling the design of highly efficient and compact application codes. In addition to standard 8-bit data management, all ST7 microcontrollers feature true bit manipulation, 8x8 unsigned multiplication and indirect addressing modes.

The voltage regulator is the L4979 which offers high precision output voltage and a programmable watchdog timer with an external capacitor. The programmable watchdog timer allows microcontroller auto-recovery from software runaway failures.

The L9638 performs LIN-bus interface functions between the protocol handler in the microcontroller and the physical bus in automotive applications. It has a Sleep mode that allows the lowest current consumption of the transceiver. It is possible to wake up the transceiver through LIN-bus, Enable input or Wake-up input.

The new VNH2SP30 window power bridge driver provides a smooth and fully-protected motor drive via 20 kHz PWM. A current sense (CS) output is used to monitor motor torque that provides an antitrap function via the ADC inputs of the microcontroller.

Finally, the L9950 actuator driver controls mirror adjustment and fold-in/-out, as well as an advanced locking system, driving the door latch and the dead bolt motor. Five intrinsic high-side drivers are available to control various lamps or LEDs, including the mirror defroster, and sophisticated diagnostic algorithms allow digital and analog load status to be monitored by reading fold, lock motors and defroster currents.

2 Algorithms

After turn on or resetting, the microcontroller initializes all used peripherals (I/Os, Timers, ADC and LIN-SCI) and variables, drives the L9950 to open the left wing-mirror and sends a LIN message to do the same for the right wing-mirror. Afterward, it starts an infinite loop that can be stopped only by resetting or turning off the board.

The microcontroller starts polling on both key pins (for driving window lift) and the keypad.

As soon as a load is turned on, the L9950 Enable Bit is set to switch the device into active mode, turning on the Charge Pump Output. This output drives the gate of an external n-channel power MOS used for reverse polarity protection. This action, guaranteeing the reverse battery protection, needs about 300µs, which is the activation delay for every load.

When no load is driven, the Enable Bit is cleared and the device goes into standby mode for power saving.

It is possible to drive the window lift by using both the PC keypad and board keys (#4 / #5 - *Figure 9*). The "Up key" and "Down key" pins are configured in the input pull-up mode, so they are normally at a high value (5V); if the UP or DOWN buttons are pressed, two different behaviors are shown, depending on the duration of the pressing time. If the button is pressed for less than 100 ms, the glass moves up or down (depending on the key pressed) until the top or bottom part of the window is reached; if the pressed time exceeds 100 ms, the window moves up or down following the touch temporization. The same behavior occurs when a PC software keypad is used.

The "Window Up switch" pin (#9 - *Figure 9*), also configured in input pull-up mode, must be connected using a mechanical switch that senses the window end run, indicating that the door upper limit has been reached.

The PWM 8-bit Autoreload Timer is used to perform a task temporized at 1 ms.

This task is in charge of all temporized events:

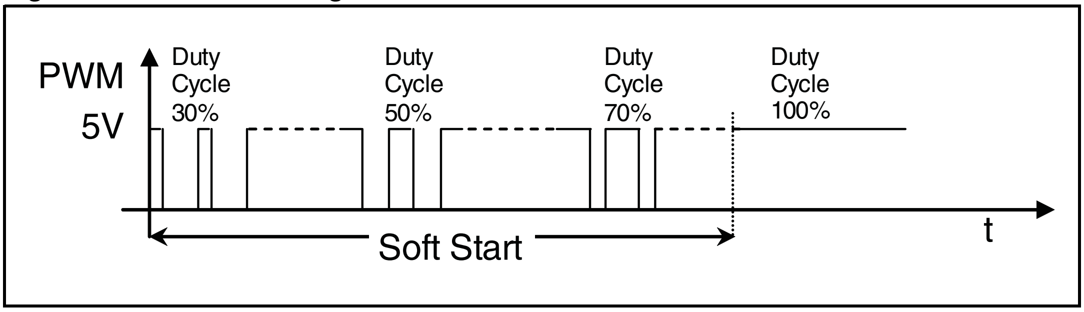
- It counts 250 ms before sending a LIN request message to the dashboard
- In case of window lift activation, it manages current sampling
- It also controls the mirror folding, locking and turn indicator light on and off switching timings

When either the Down or Up key (both in the board or on the PC keypad) are pressed, the Window_Lift routine is called:

- VNH2 InA and InB pins are, then, set or reset, depending on the key pressed, to lift the window up or down and,
- the 16-bit Timer is used to provide to the VNH2 a PWM signal with 20 kHz frequency and 30% duty cycle
- during a 1 ms task, the current sense voltage is acquired via ST7 ADC; the acquisition is averaged over a 10 ms period to eliminate noise. Motor power is calculated by multiplying the current sense by the estimated angular velocity. This value is averaged over 100 ms, providing a delayed signal compared to the original. The difference between power and averaged power must be compared with a threshold to determine whether a pinch event occurs. This threshold depends on the motor status (soft start up or steady state conditions).

Unless a Down/Up key is pressed or a pinch occurs, the duty cycle increases linearly until 100% is reached and the PWM becomes a constant (steady state phase, see *Figure 2*.)

Figure 2. VNH2 PWM signal



At this point the system waits for any event: A pressed key or a pinch.

If a key is pressed, the motor is stops, resetting VNH2 PWM and setting InA and InB for braking to Vbatt the motor and stopping the glass.

In case of a pinch event, the Window Up switch is first checked. If this switch is closed (upper limit reached), the glass is locked driving the motor up for 800 ms.

Otherwise, if the glass moves up, it is driven down for 800 ms to release the pinched object; if it goes down the motor is stopped immediately.

When an abnormal condition is detected (Open Load, Short circuit or thermal shutdown), a LIN message is sent to the PC VNH2 diagnostic node. Open Load is detected by current sense voltage monitoring (value) while a short circuit and thermal shutdown are detected using the VNH2 DIAGx pin. When DIAGx pin is reset, while INx is set, the status pin indicates a thermal shutdown; otherwise, if INx is reset the status pin detects a short circuit.

For further information about antipinch algorithms, please refer to “AN2095 - VNH2 for window lift with antipinch routine”.

When a command for L9950 loads is detected through keypad input pins, the microcontroller drives such loads through the L9950 SPI bus. Serial data for controlling outputs and for receiving status registers is sent via this bus.

For example, if one of the over-current bits is set, the corresponding driver is disabled. If the over-current recovery bit of the output is not set, the microcontroller must clear the over-current bit to enable the driver again.

If the thermal shutdown bit is set, all drivers go to high impedance state. Again, the microcontroller must clear the bit to enable the drivers.

When the Fold Mirror motor is activated, the relevant motor is driven for 4 seconds with a fully charged battery and for 6 seconds in other cases. Since the OUT1 is common to all mirror motors, it is impossible to drive two or more mirror motors at the same time.

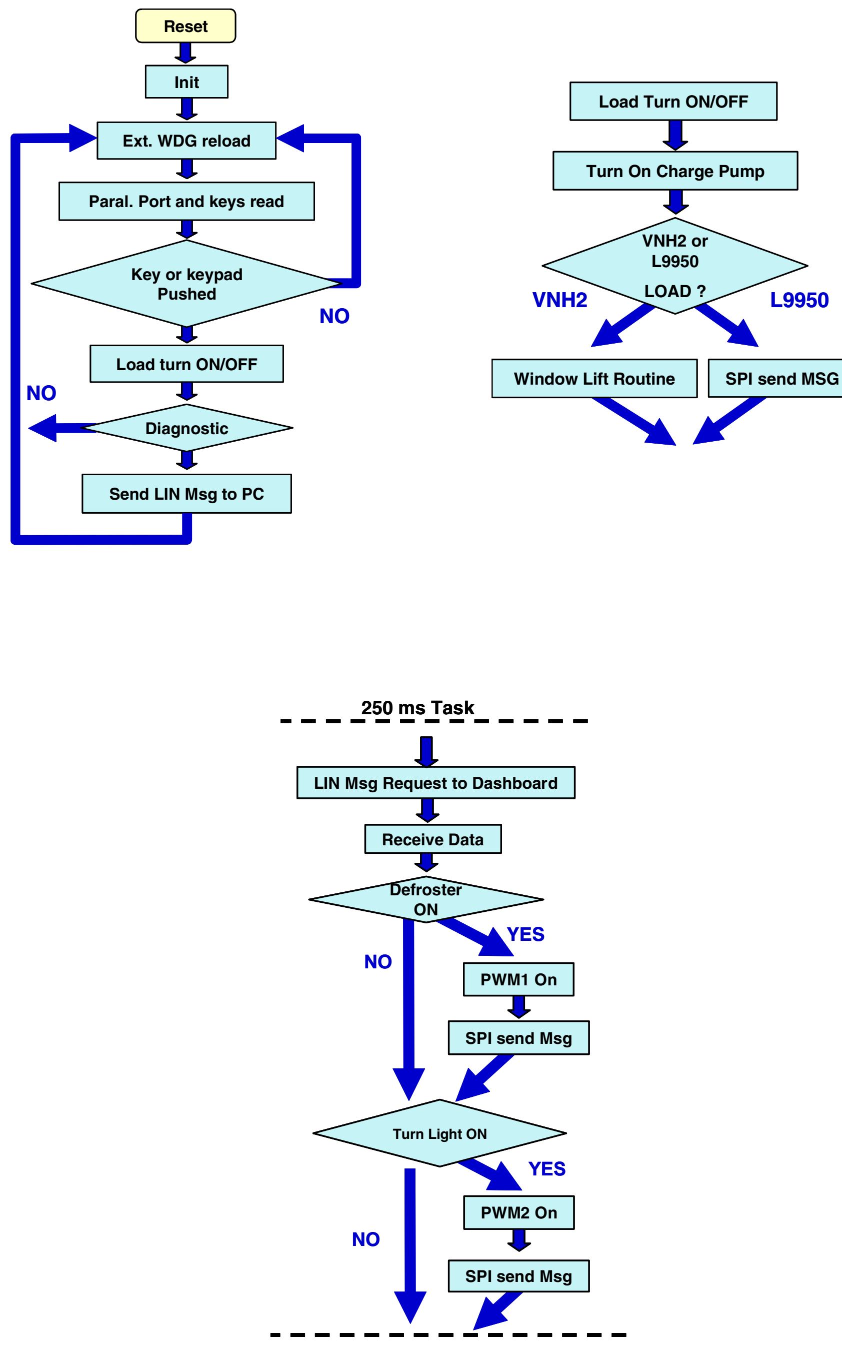
The turn indicator and the defroster are driven using PWM2 and PWM1 respectively.

For all lights, the over-current Recovery Enable bit is set by the microcontroller; this automatically reactivates the output after a delay time, resulting in a PWM modulated current with a programmable duty cycle.

The over-current recovery feature is intended for loads which have an initial current higher than the over-current limit of the output (for example, cold light bulbs).

The described algorithm, including LIN message managing, is stored in less than 10 kB on the microcontroller Flash memory.

Figure 3. Simplified algorithm flowcharts



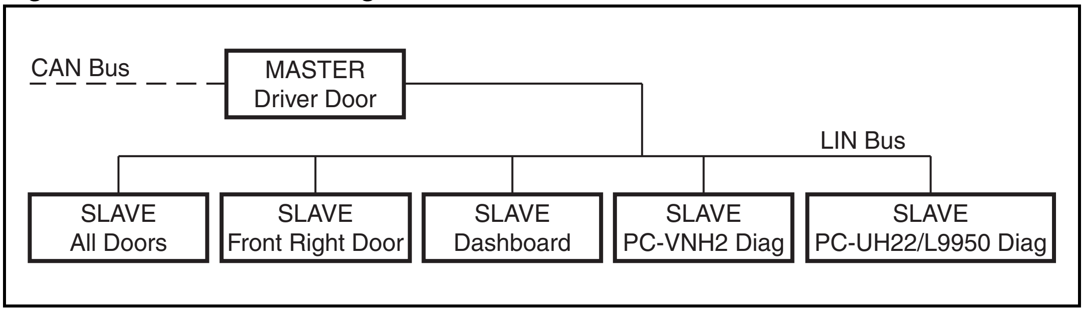
3 Local Interconnect Network (LIN) messages

Created by the LIN Consortium (a collection of automotive, software and semiconductor manufacturers), LIN is a low speed bus with a maximum speed of 20kbaud. The most significant advantage offered by LIN bus is the low cost of implementation. Implementing a LIN node costs approximately half of an equivalent CAN bus node.

The LIN bus protocol is based on the common UART byte interface. Any microcontroller with a UART interface can be used as a node on the LIN bus since the basic transmission uses the UART format. The LIN bus protocol specification 1.2 defines three standard baud rates: 2400 baud, 9600 baud and 19200 baud. Communication is based on a master/slave mechanism.

The bus is composed of one master node (Driver door) and five slave nodes (All Doors, Front Right Door, Dashboard, PC VNH2 Diagnostic and PC UH22/L9950 Diagnostic). All arbitration and collision management takes place in the master node to further simplify and lower the cost of the slave nodes.

Figure 4. Master/slave diagram

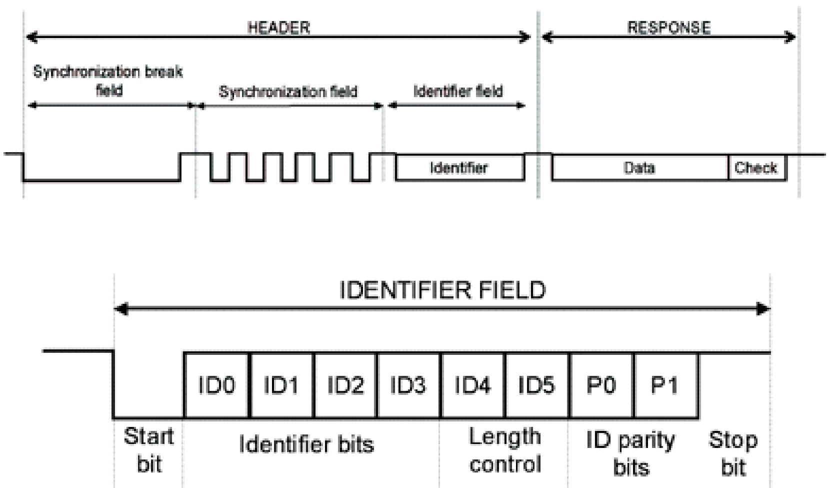


- **All Doors** is used for locking/unlocking all doors
- **Front Right Door** is the slave node in the front right door that receives all messages for mirror positioning (X-Y and fold) and for front right window lift
- **Dashboard** is questioned from the master node each 250 ms in order to give information for turning on/off turn indicator and defroster
- **PC-VNH2** is used for demonstration purposes to send VNH2 diagnostic information to PC
- **PC-UH22/L9950** is also used for demonstration purposes to send L9950 diagnostic information to PC

All communications on the bus take the form of messages, which have a defined format known as the message frame. A diagram showing the format of the message is shown in *Figure 5.* The message frame is composed of a header and a response. The header is further broken down into three fields:

- The synchronization break field, composed of 13 dominant bits (0) and at least one recessive bit (1), which indicates the beginning of a frame.
- The synchronization field, which allows a slave to be synchronized on the current master baud rate.
- The identifier field, which identifies the requested message and the length of the response field.

Figure 5. LIN bus message frame format diagram



Only the master node can initiate a message by sending a header field that is received by all nodes. Each slave node analyses the header and must be ready to send or receive data during the response field of the frame. The identifier field within the header informs all slave nodes in the network of the appropriate action to take. Such actions include:

– Receiving bytes transmitted in the response field
– Transmitting bytes in the response field
– Doing nothing

ST72F561 has a flexible microcontroller architecture that makes the implementation of LIN bus communication much easier than on other devices.

The LIN bus transmission of a master node, as this demoboard, has three distinct phases:

– Synchronization break field transmission: The length of this field is 13 dominant bits and 1 recessive bit
– Data byte transmission: The synchronization field, the identifier field, the data fields and the checksum field are each one byte fields.
– Data byte reception: The data fields and the checksum field are also transmitted in a byte wise manner.

The LIN bus protocol uses a standard baud rate: 19200 baud. In case of microcontroller debugging, when CPU frequency is divided by two, a 9600 baud protocol is also possible. The software PC allows users to change the baud rate.

It is necessary to define a specific ID (6 bits long) for each message, which is used for reception or transmission. The message ID is written in a hexadecimal format with the parity bits included. According to the LIN specification, the data field can be 1 to 8 bytes long (for LIN 1.2 and newer).

The parity bits P0 and P1 are calculated as follows:

$$P0=ID0 \oplus ID1 \oplus ID2 \oplus ID4$$

$$P0=ID1 \oplus ID3 \oplus ID4 \oplus ID5$$

Because the application is for demonstration purposes, it uses two LIN messages for transmission with PC (2 bytes long for VNH2 diagnostic and 8 bytes long for L9950 diagnostic).

Table 2. Application Messages

Message ID	Slave response source	Slave response destination	N.transmit [bytes]	Message Data	Description
LinMsg1 (0xc1)	Driver Door (M)	FR RL RR Doors (S)	2	See *Table 3*	Master transmit doors lock / unlock command
LinMsg2 (0x42)	Driver Door (M)	Front Right Door (S)	2	See *Table 4*	Master transmit FR Door commands
LinMsg4 (0x73)	Driver Door (M)	PC-L9950 Diag (S)	8	See *Table 7*	Master transmit diagnostic signals
LinMsg3 (0xC4)	Driver Door (M)	PC-VNH2 Diag (S)	2	See *Table 5*	Master transmit RL Door commands
LinMsg5 (0x11)	Dashboard (S)	Driver Door (M)	2	See *Table 6*	Master data request

Table 3. FR RL RR data

Description	Data Value
Unlock Doors	0x01
Lock Doors	0x02

Table 4. Front Right Door

Description	Data Value
Front Right Mirror Dx	0x01
Front Right Mirror Sx	0x02
Front Right Mirror Up	0x03
Front Right Mirror Down	0x04
Front Right Mirror Close	0x05
Front Right Mirror open	0x06
Front Right Window Up	0x07
Front Right Window Down	0x08

Table 5. PC-VNH2 Diag. data

Description	Data Value
No diag err	0x01
Therm. Shutdown Leg A	0x27
Therm. Shutdown Leg B	0x28
Short Circ. Leg A	0x29
Short Circ. Leg B	0x2A
Open Load	0x2B

Table 6. Dashboard data

Description	Data Value
No action	0x01
Defroster turn on	0x02
Turn light turn on	0x03
Defroster and Turn light turn on	0x04

Table 7. PC - L9950 Diag. data

Data0		Data1		Data2		Data3		Data4	
Bit	Description	Bit	Description	Bit	Description	Bit	Description	Bit	Description
0	VS Over Voltage	0	Out 2 HS Over Current	0	Out 6 HS Over Current	0	Out 2 LS Open Load	0	Out 6 LS Open Load
1	VS Under Voltage	1	Out 3 LS Over Current	1	Out 7 HS Over Current	1	Out 2 HS Open Load	1	Out 6 HS Open Load
2	Thermal Shutdown	2	Out 3 HS Over Current	2	Out 8 HS Over Current	2	Out 3 LS Open Load	2	Out 7 HS Open Load
3	Temperature Warning	3	Out 4 LS Over Current	3	Out 9 HS Over Current	3	Out 3 HS Open Load	3	Out 8 HS Open Load
4	N.U.	4	Out 4 HS Over Current	4	Out 10 HS Over Current	4	Out 4 LS Open Load	4	Out 9 HS Open Load
5	Out 1 LS Over Current	5	Out 5 LS Over Current	5	Out 11 HS Over Current	5	Out 4 HS Open Load	5	Out 10 HS Open Load
6	Out 1 HS Over Current	6	Out 5 HS Over Current	6	Out 1 LS Open Load	6	Out 5 LS Open Load	6	Out 11 HS Open Load
7	Out 2 LS Over Current	7	Out 6 LS Over Current	7	Out 1 HS Open Load	7	Out 5 HS Open Load	7	N.U.

4 PC Keypad software

Before switching on the board, the PC “Keypad” software must be run, defining the parallel port base address (see Windows Device Manager) and the COM serial port to be used.

PC software has four different functions: Keyboard, dashboard, LIN analyzer, VNH2 and L9950 diagnostic view.

– **Keyboard** is connected to the microcontroller via a parallel port and provides to the microcontroller an 8-bit word that encodes all possible user actions as described in the following *Table 8* and *Table 9*:

Table 8. Window and door lock coding

Window				Door	Parallel Port
Up Left	Down Left	Up Right	Down Right	Lock / Unlock	
0	0	0	0	0	**192**
0	0	0	1	0	**194**
0	1	0	1	0	**202**
0	1	0	0	0	**200**
0	0	1	0	0	**196**
1	0	1	0	0	**212**
1	0	0	0	0	**208**
0	0	0	0	1	**193**
0	0	0	1	1	**195**
0	1	0	1	1	**203**
0	1	0	0	1	**201**
0	0	1	0	1	**197**
1	0	1	0	1	**213**
1	0	0	0	1	**209**

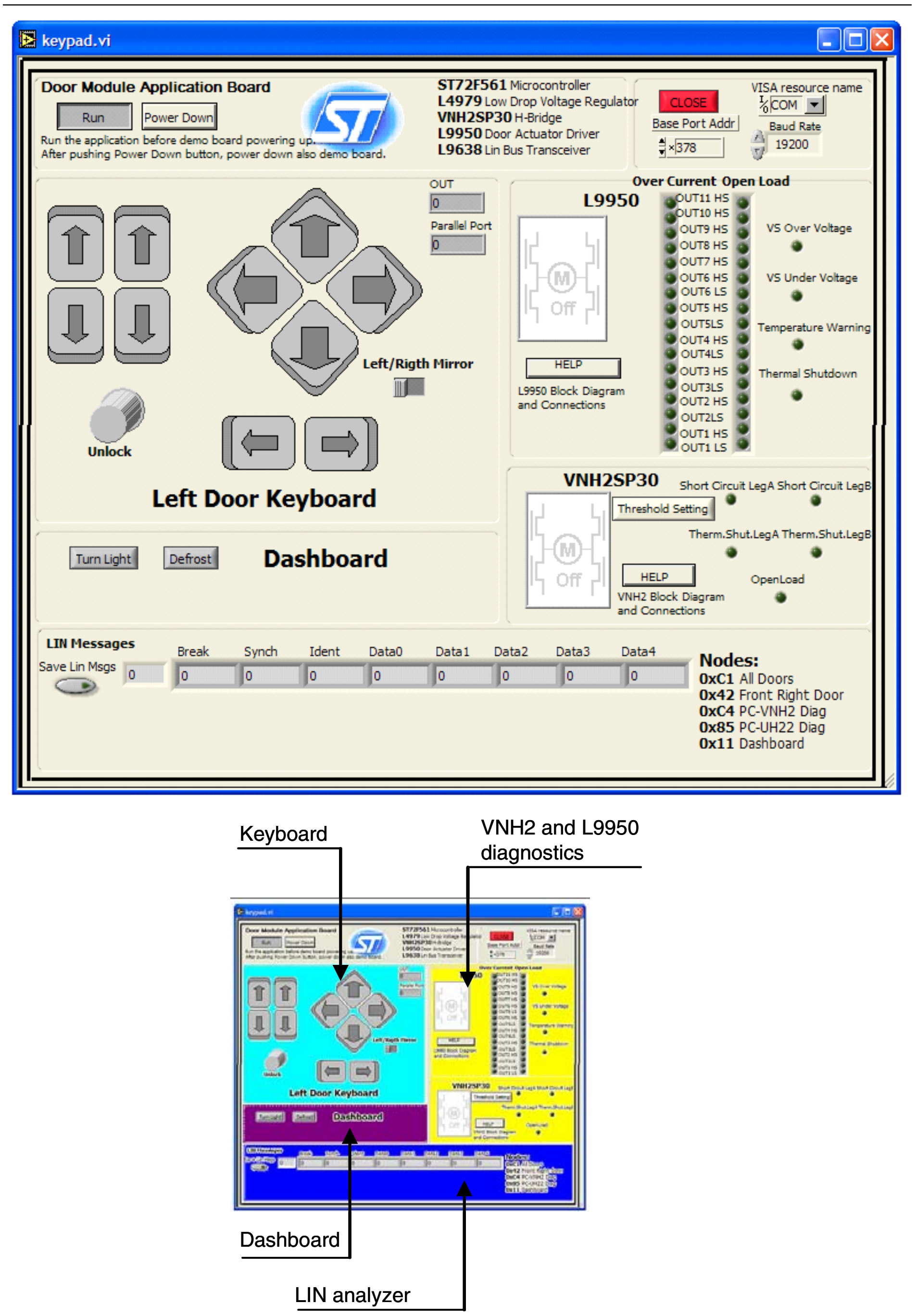
Table 9. Mirror and door lock coding

Mirror							Door	Parallel Port
Open	Close	Left / Right	Up	Down	Left	Right	Lock / Unlock	
0	0	0	0	0	0	0	0	**0**
1	0	0	0	0	0	0	0	**128**
0	1	0	0	0	0	0	0	**64**
1	0	1	0	0	0	0	0	**160**
0	1	1	0	0	0	0	0	**96**
0	0	0	1	0	0	0	0	**16**
0	0	0	0	1	0	0	0	**8**
0	0	0	0	0	1	0	0	**4**
0	0	0	0	0	0	1	0	**2**
0	0	1	1	0	0	0	0	**48**
0	0	1	0	1	0	0	0	**40**
0	0	1	0	0	1	0	0	**36**
0	0	1	0	0	0	1	0	**34**
0	0	0	0	0	0	0	1	**1**
1	0	0	0	0	0	0	1	**129**
0	1	0	0	0	0	0	1	**65**
1	0	1	0	0	0	0	1	**161**
0	1	1	0	0	0	0	1	**97**
0	0	0	1	0	0	0	1	**17**
0	0	0	0	1	0	0	1	**9**
0	0	0	0	0	1	0	1	**5**
0	0	0	0	0	0	1	1	**3**
0	0	1	1	0	0	0	1	**49**
0	0	1	0	1	0	0	1	**41**
0	0	1	0	0	1	0	1	**37**
0	0	1	0	0	0	1	1	**35**

- **Dashboard** is a LIN slave node and is used to switch on or off the turn indicator light and the defroster.
- **LIN Analyzer** “sniffs” all LIN messages that flow through bus.
- **VNH2 and L9950 diagnostic** graphically shows the LIN messages addressing VNH2 and L9950 diagnostic nodes.

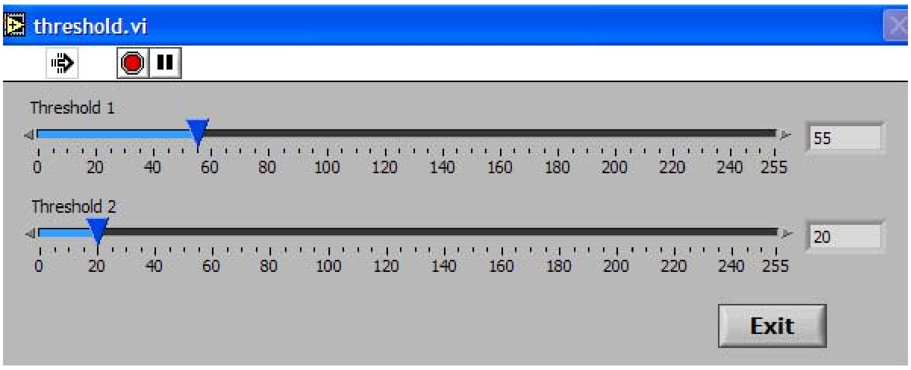
Figure 6 displays the program main window snapshot.

Figure 6. PC keypad screenshot



PC software can also change the thresholds for the window lift routine. Any threshold change modifies the motor torque in a pinch state. For further details, please refer to "AN2095 - VNH2 for window lift with antipinch routine".

Figure 7. Adjust thresholds screenshot



5 Hardware implementation

The voltage regulator, which provides the required 5V during normal mode, is enabled by the LIN transceiver through Inhibit Output pin INH. To reset the microcontroller with the watchdog timer in case of a missing pulse, a jumper must be installed in the Voltage Regulator Reset connector (#1 - *Figure 8*).

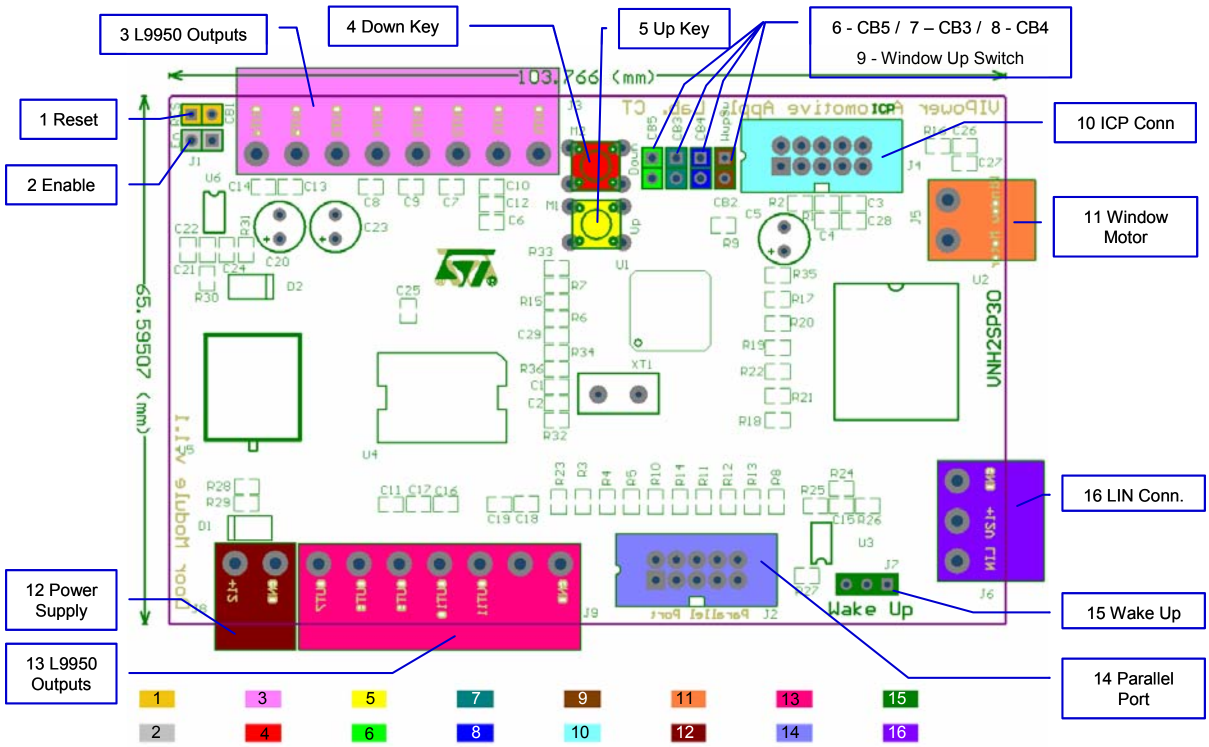
The jumper must be removed in case of microcontroller reprogramming; in fact, in this phase, the microcontroller cannot provide pulses to the voltage regulator that should provide a low level reset. Moreover, during the programming phase, to enable the Voltage regulator a jumper must be installed in the Voltage Regulator Enable connector (#2 - *Figure 8*) because in this case the LIN bus interface L9638 cannot provide the enabling voltage.

In summary, the microcontroller can be reprogrammed using the ICP connector (#10 - *Figure 8*) by removing the Reset jumper (#1 - *Figure 8*) and settling the Enable jumper (#2 - *Figure 8*). See the ST72F561 datasheet for more details.

A jumper on CB4 connector (#8 - *Figure 8*) allows using the board without using the LIN Interface as detailed in *Section 3: Local Interconnect Network (LIN) messages*.

The board connector locations are given in *Figure 8.*

Figure 8. Board connectors



The L9950 loads are shown in the following table:

Table 10. L9950 Loads

L9950 Outputs	Load
OUT1	Common Mirror Motors
OUT2	X Mirror Motor
OUT3	Y Mirror Motor
OUT4	Lock Motor
OUT5	Lock Motor
OUT6	Mirror Fold Motor
OUT7	Exterior Light
OUT8	Footstep Light
OUT9	Safety Light
OUT10	Turn Indicator
OUT11	Defroster

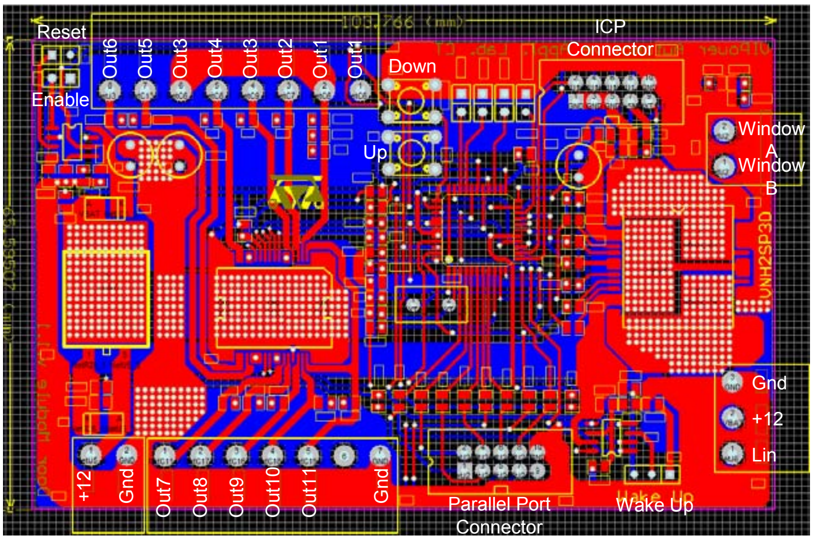
The input signals INA and INB are directly interfaced with the microcontroller to select the motor direction and the brake condition. The DIAGA/ENA or DIAGB/ENB, connected to the I/O microcontroller and configured as input pull-up, enable the legs of the bridge. They also provide a digital diagnostic signal. The CS pin allows monitoring the motor current by delivering a current proportional to its value. The PWM, up to 20 KHz, allows control of the motor speed in all possible conditions.

The reverse polarity protection MOSFET needs a zener diode and a resistor between gate and source to protect against ISO pulses and for proper turn off in static reverse polarity.

In master node application, a LOW ohmic resistor must be connected externally between LIN and battery to allow the maximum transmission rate.

Finally, all outputs need a 10nF capacitor to protect the module against 8 kV ESD events.

Figure 9. Board Layout



Appendix A Schematic

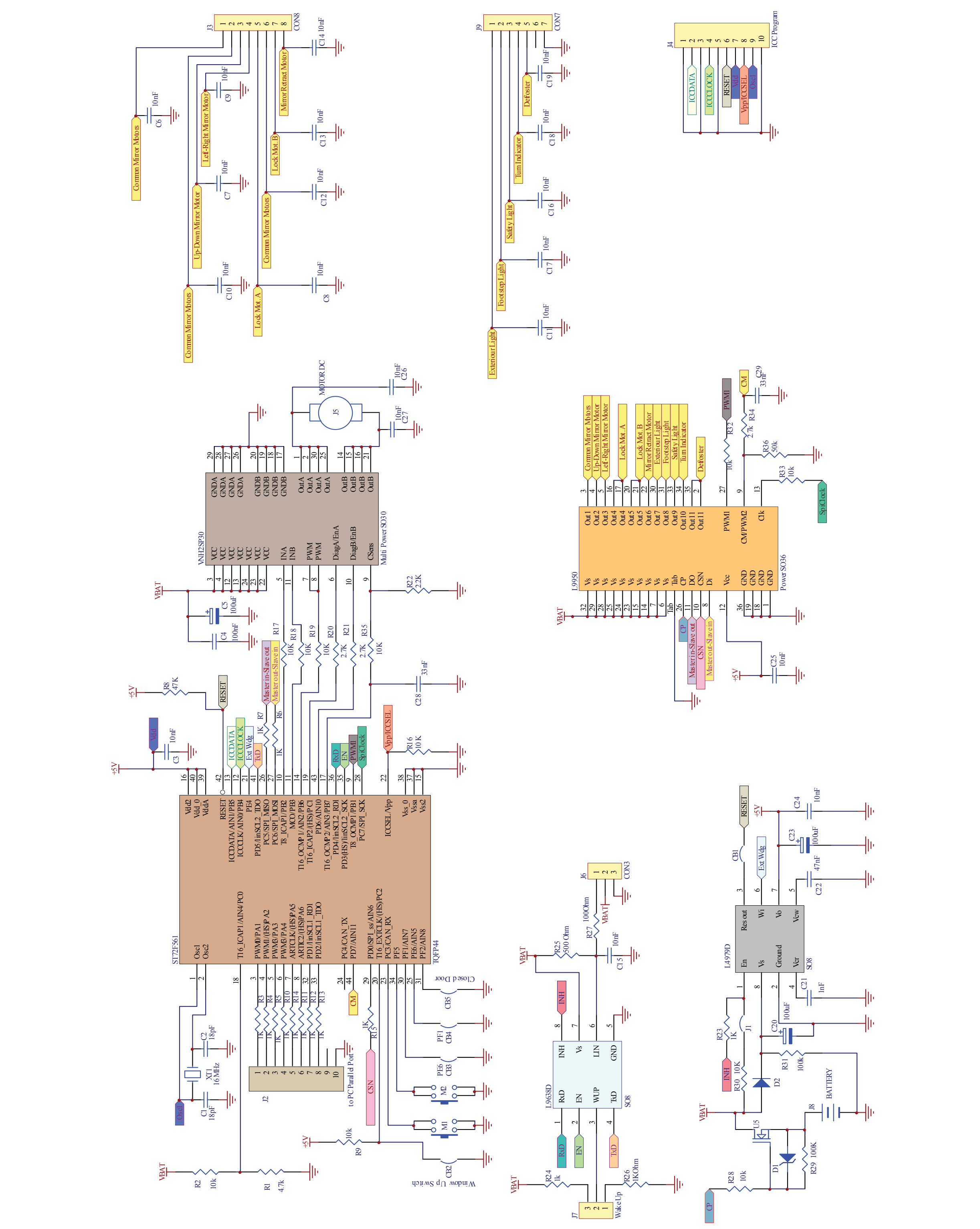


Figure 10. Board Schematic

Revision history

Table 11. Document revision history

Date	Revision	Changes
07-Apr-2006	1	Initial release.
19-Sep-2013	2	Updated disclaimer.