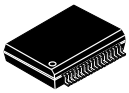
# NCV70627

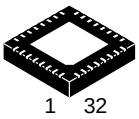
# Micro-stepping Motor Driver

ON Semiconductor®

www.onsemi.com



**SSOP−EP**
**36 LEAD**
**CASE 940AB**




**QFN32, 5x5**
**CASE 488AM**

**ORDERING INFORMATION**

See detailed ordering, marking and shipping information in the package dimensions section on page 2 of this data sheet.

### Introduction

The NCV70627 is a single−chip micro−stepping motor driver with position controller and control/diagnostic interface. It is ready to build dedicated mechatronics solutions connected remotely with a LIN master.

The chip receives positioning instructions through the bus and subsequently drives the motor coils to the desired position. The on−chip position controller is configurable (OTP or RAM) for different motor types, positioning ranges and parameters for speed, acceleration and deceleration. The NCV70627 acts as a slave on the LIN bus and the master can fetch specific status information like actual position, error flags, etc. from each individual slave node.

An integrated sensor−less step−loss detection prevents the positioner from loosing steps and stops the motor when running into stall. This enables silent, yet accurate position calibrations during a referencing run and allows semi−closed loop operation when approaching the mechanical end−stops.

The chip is implemented in I3T50 technology, enabling both high voltage analog circuitry and digital functionality on the same chip. The NCV70627 is fully compatible with the automotive voltage requirements. Due to the technology, the device is especially suited for use in applications with fluctuating battery supplies.

## PRODUCT FEATURES

### Motordriver

- Micro−stepping Technology
- Sensorless Step−loss Detection
- Peak Current up to 800 mA
- Low Temperature Boost Current up to 1100 mA
- Programmable Current Stabilization Phase
- Fixed Frequency PWM Current−control
- Automatic Selection of Fast and Slow Decay Mode
- No External Fly−back Diodes Required
- Compliant with 14 V Automotive Systems

### Controller with RAM and OTP Memory

- Position Controller
- Configurable Speeds and Acceleration
- Input to Connect Optional Motion Switch

### LIN Interface

- Physical Layer Compliant to LIN rev. 2.0. Data−link Layer Compatible with LIN rev. 1.3 (Note 1)
- Field−programmable Node Addresses
- Dynamically Allocated Identifiers
- Diagnostics and Status Information

### Protection

- Overcurrent Protection
- Open−circuit Detection
- High Temperature Warning and Management
- Low Temperature Flag
- LIN Bus Short−circuit Protection to Supply and Ground
- Lost LIN Safe Operation
- Enhanced Under Voltage Management

### Power Saving

- Powerdown Supply Current < 150 μA
- 3.3 V Regulator with Wake−up On LIN Activity

### EMI Compatibility

- LIN Bus Integrated Slope Control
- HV Outputs with Slope Control
- This is a Pb−Free Device

1. Minor exceptions to the conformance of the data−link layer to LIN rev. 1.3.

© Semiconductor Components Industries, LLC, 2015

## Applications

The NCV70627 is ideally suited for small positioning applications. Target markets include: automotive (headlamp alignment, HVAC, idle control, cruise control), industrial equipment (lighting, fluid control, labeling, process control, XYZ tables, robots...) and building automation (HVAC, surveillance, satellite dish, renewable energy systems). Suitable applications typically have multiple axes or require mechatronics solutions with the driver chip mounted directly on the motor.

**Table 1. ORDERING INFORMATION**

| Part No. | Peak Current | End Market/Version | Package* | Shipping† |
|---|---|---|---|---|
| NCV70627DQ001G | 800/1100 mA (Note 2) | Automotive High Temperature Version | SSOP−36EP (Pb−Free) | 47 Units/Rail |
| NCV70627DQ001R2G | 800/1100 mA (Note 2) | | | 1500/Tape & Reel |
| NCV70627MW002R2G | 800/1100 mA (Note 2) | Automotive | QFN32 (Pb−Free) | 5000 / Tape & Reel |

*For additional information on our Pb−Free strategy and soldering details, please download the ON Semiconductor Soldering and Mounting Techniques Reference Manual, SOLDERRM/D.

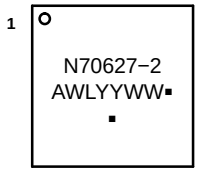
†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specifications Brochure, BRD8011/D.

2. The device boost current. This applies for operation under the thermal warning level only.

**MARKING DIAGRAMS**




SSOP




QFN32

A = Assembly Location
WL = Wafer Lot
YY = Year
WW = Work Week
G or ▪ = Pb−Free Package

(Note: Microdot may be in either location)

**Table 2. ABSOLUTE MAXIMUM RATINGS**

| Parameter | | Min | Max | Unit |
|---|---|---|---|---|
| $V_{BB}$, VHW2 | Supply voltage, hardwired address pin (Note 4) | −0.3 | +40 (Note 3) | V |
| Vlin | Bus input voltage (Note 4) | −40 | +40 | V |
| $T_J$ | Junction temperature range (Note 5) | −50 | +175 | °C |
| $T_{stg}$ | Storage temperature range (Note 6) | −55 | +160 | °C |
| Vesd (Note 7) | HBM Electrostatic discharge voltage on LIN pin | −4 | +4 | kV |
| | HBM Electrostatic discharge voltage on other pins | −2 | +2 | kV |
| | MM Electrostatic discharge voltage on other pins | −200 | +200 | V |

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

NOTE: A mission profile (Note 5) is a substantial part of the operation conditions; hence the Customer must contact ON Semiconductor in order to mutually agree in writing on the allowed missions profile(s) in the application.

3. For limited time: $V_{BB}$ <0.5 s, SWI and HW2 pins <1.0 s.
4. Maximum allowed voltage between two device pins is 60 V.
5. The circuit functionality is not guaranteed outside the Operating junction temperature range.
   A mission profile describes the application specific conditions such as, but not limited to, the cumulative operating conditions over life time, the system power dissipation, the system's environmental conditions, the thermal design of the customer's system, the modes, in which the device is operated by the customer, etc.
6. For limited time up to 100 hours. Otherwise the maximum storage temperature is 85°C.
7. HBM according to AEC−Q100: EIA−JESD22−A114−B (100 pF via 1.5 kΩ) and MM according to AEC−Q100: EIA−JESD22−A115−A.

**Table 3. OPERATING RANGES**

| Parameter | | Min | Max | Unit |
|---|---|---|---|---|
| $V_{BB}$ | Supply voltage | +5.5 | +29 | V |
| $T_{JP}$ | Parametric Operating junction temperature range (Note 8) | −40 | +145 | °C |
| $T_{JF}$ | Functional Operating junction temperature range (Note 9) | −40 | +160 | °C |

8. The parametric characteristics of the circuit are not guaranteed outside the parametric operating junction temperature range.
9. The maximum functional operating temperature range can be limited by thermal shutdown Ttsd.

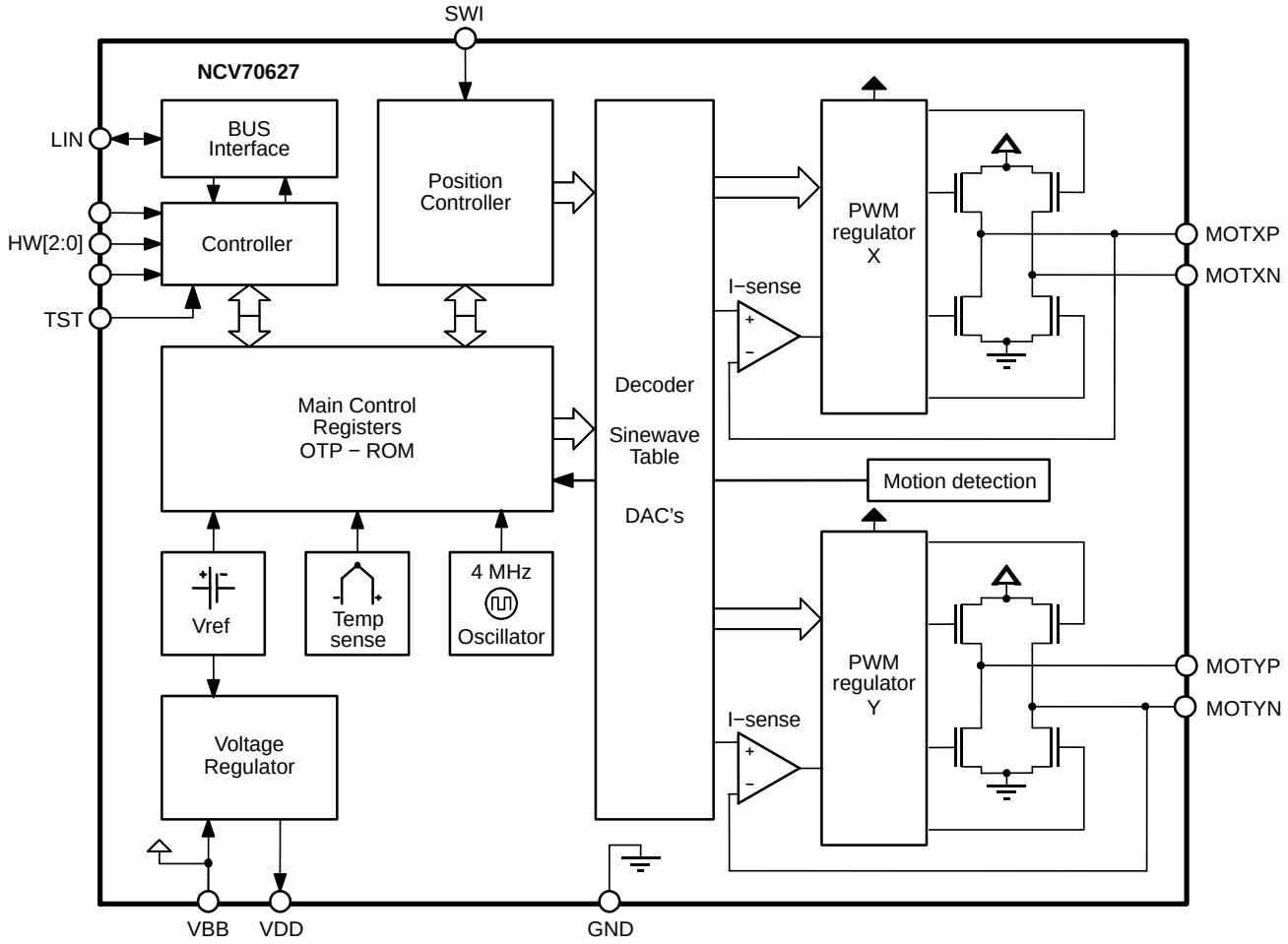



**Figure 1. Block Diagram**

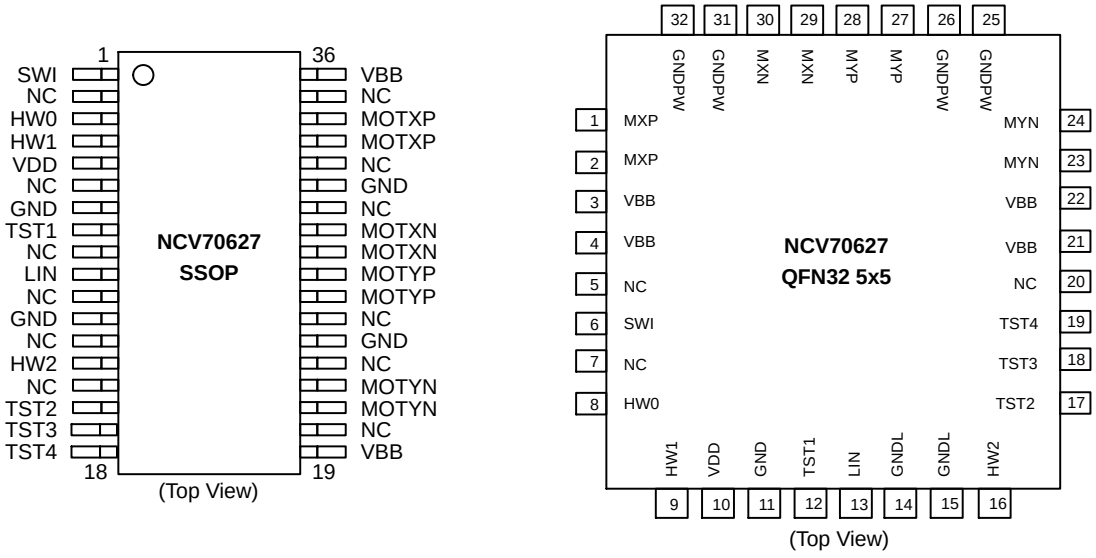



**Figure 2. Pinout Diagrams**

**Table 4. PIN DESCRIPTIONS − SSOP PACKAGE**

| Pin No. | Pin Name | Pin Description | |
|---|---|---|---|
| 1 | SWI | Switch input | |
| 3 | HW0 | Bit 0 of LIN−ADD | To be tied to GND or $V_{DD}$ |
| 4 | HW1 | Bit 1 of LIN−ADD | |
| 5 | VDD | Internal supply (needs external decoupling capacitor) | |
| 7, 12, 24, 31 | GND | Ground, heat sink | |
| 8 | TST1 | Test pin (to be tied to ground in normal operation) | |
| 10 | LIN | LIN−bus connection | |
| 14 | HW2 | Bit 2 LIN−ADD | |
| 16 | TST2 | Test pin (to be tied to ground in normal operation) | |
| 17 | TST3 | Test pin (to be tied to ground in normal operation) | |
| 18 | TST4 | Test pin (to be tied to ground in normal operation) | |
| 19, 36 | VBB | Battery voltage supply | |
| 21, 22 | MOTYN | Negative end of phase Y coil | |
| 26, 27 | MOTYP | Positive end of phase Y coil | |
| 28, 29 | MOTXN | Negative end of phase X coil | |
| 33, 34 | MOTXP | Positive end of phase X coil | |
| 2, 6, 9, 11, 13, 15, 20, 23, 25, 30, 32, 35 | NC | Not used | |

**Table 5. PIN DESCRIPTIONS − QFN PACKAGE**

| Pin No. | Pin Name | Pin Description | |
|---|---|---|---|
| 1, 2 | MXP | Positive end of phase X coil | |
| 3, 4, 21, 22 | VBB | Battery voltage supply | |
| 5, 7, 20 | NC | Not used | |
| 6 | SWI | Switch input | |
| 8 | HW0 | Bit 0 of LIN−ADD | To be tied to GND or $V_{DD}$ |
| 9 | HW1 | Bit 1 of LIN−ADD | |
| 10 | VDD | Internal supply (needs external decoupling capacitor) | |
| 11 | GND | Ground | |
| 12 | TST1 | Test pin (to be tied to ground in normal operation) | |
| 13 | LIN | LIN−bus connection | |
| 14, 15 | GNDL | Ground | |
| 16 | HW2 | Bit 2 LIN−ADD | |
| 17 | TST2 | Test pin (to be tied to ground in normal operation) | |
| 18 | TST3 | Test pin (to be tied to ground in normal operation) | |
| 19 | TST4 | Test pin (to be tied to ground in normal operation) | |
| 23, 24 | MYN | Negative end of phase Y coil | |
| 25, 26, 31, 32 | GNDPW | Ground | |
| 27, 28 | MYP | Positive end of phase Y coil | |
| 29, 30 | MXN | Negative end of phase X coil | |

## Package Thermal Resistance

The NCV70627 is available in thermally optimized SSOP−36 and QFN32 packages. For the optimizations, the package has an exposed thermal pad which has to be soldered to the PCB ground plane. The ground plane needs thermal vias to conduct the heat to the bottom layer. Figure 3 gives examples for good power distribution solutions.

The thermal resistances are presented in Table 6: Thermal resistance.

**Table 6. THERMAL RESISTANCE**

| Characteristics | Package | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Thermal Resistance, Junction−to−Exposed Pad (Note 10) | SSOP−36 | $R_{\theta JP}$ | − | 3.3 | − | K/W |
| Thermal Resistance, Junction−to−Exposed Pad (Note 10) | QFN32 | $R_{\theta JP}$ | − | 14 | − | K/W |

10. Also includes typical solder thickness under the Exposed Pad (EP).

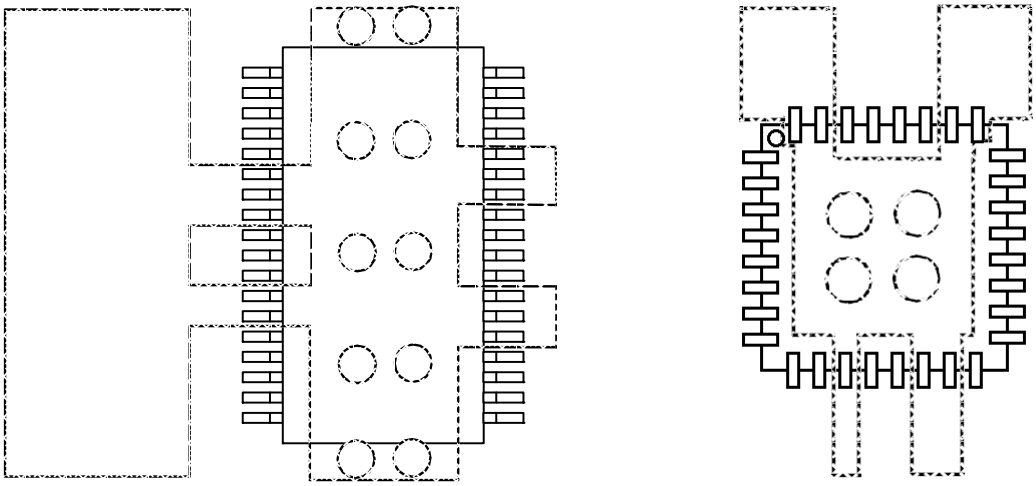


**Figure 3. Example of SSOP−36 and QFN32 PCB Ground Plane Layout. Preferred layout at top and bottom connected with through−hole filled vias**

## DC Parameters

The DC parameters are guaranteed over junction temperature from −40 to 145°C and $V_{BB}$ in the operating range from 5.5 to 29 V, unless otherwise specified. Convention: currents flowing into the circuit are defined as positive.

**Table 7. DC PARAMETERS**

| Symbol | Pin(s) | Parameter | Test Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| **MOTORDRIVER** | | | | | | | |
| $I_{MS-max,Peak}$ | MOTXP MOTXN MOTYP MOTYN | Max current through motor coil in normal operation | $V_{BB}$ = 14 V | | 800 | | mA |
| $I_{MS-max,RMS}$ | | Max rms current through coil in normal operation | $V_{BB}$ = 14 V | | 570 | | mA |
| $I_{MSabs}$ | | Absolute error on coil current (Note 11) | $V_{BB}$ = 14 V, $T_j$ =145°C | −10 | | 10 | % |
| $I_{MSrel}$ | | Matching of X & Y coil currents | $V_{BB}$ = 14 V | −7 | 0 | 7 | % |
| $I_{MS-boost_Peak}$ | | Max peak current during booster function | $V_{BB}$ = 14 V, T < $T_{tw}$ | | 1100 | | mA |
| $R_{DS(on)}$ | | On resistance of High side + Low side Driver at $I_{MSmax}$ | $T_j \le 25°C$ | | | 1.8 | Ω |
| | | | $T_j$ = 145°C | | | 2.4 | Ω |
| **LIN TRANSMITTER** (Note 19) | | | | | | | |
| $I_{bus_off}$ | LIN | Dominant state, driver off | $V_{bus}$ = 0 V, $V_{BB}$ = 7 V & 18 V | −1 | | | mA |
| $I_{bus_off}$ | | Recessive state, driver off | $V_{bus}$ = $V_{bat}$, $V_{BB}$ = 7 V & 18 V | | | 10 | μA |
| $I_{bus_off}$ | | Recessive state, driver off | $V_{BB}$ = 0 V (Note 11) | | | 10 | μA |
| $I_{bus_lim}$ | | Current limitation | $V_{BB}$ = 7 V & 18 V | 50 | 75 | 200 | mA |
| $R_{slave}$ | | Pullup resistance | $V_{BB}$ = 7 V & 18 V | 20 | 30 | 47 | kΩ |
| **LIN RECEIVER** (Note 19) | | | | | | | |
| $V_{bus_dom}$ | LIN | Receiver dominant state | $V_{BB}$ = 7 V & 18 V | 0 | | 0.4 * $V_{BB}$ | V |
| $V_{bus_rec}$ | | Receiver recessive state | $V_{BB}$ = 7 V & 18 V | 0.6 * $V_{BB}$ | | $V_{BB}$ | V |
| $V_{bus_hys}$ | | Receiver hysteresis | $V_{BB}$ = 7 V & 18 V | 0.05 * $V_{BB}$ | | 0.2 * $V_{BB}$ | V |
| **THERMAL WARNING & SHUTDOWN** | | | | | | | |
| $T_{tw}$ | | Thermal warning (Notes 12 and 13) | | 150 | 157 | 165 | °C |
| $T_{tsd}$ | | Thermal shutdown (Note 14) | | | $T_{tw}$ + 10 | | °C |
| $T_{low}$ | | Low temperature warning (Note 14) | | | $T_{tw}$ − 167 | | °C |
| **SUPPLY AND VOLTAGE REGULATOR** | | | | | | | |
| $V_{bbOTP}$ | $V_{BB}$ | Supply voltage for OTP zapping (Note 15) | | 13.0 | | 18.0 | V |
| $UV_2$ | | Stop voltage low threshold | | 5.48 | 5.90 | 6.32 | V |
| $UV_3$ | | Decelerated stop voltage low threshold | UV3Thr[2:0] = 000 | 5.48 | 5.90 | 6.32 | V |
| | | | UV3Thr[2:0] = 001 | 5.86 | 6.30 | 6.74 | V |
| | | | UV3Thr[2:0] = 010 | 6.23 | 6.70 | 7.17 | V |
| | | | UV3Thr[2:0] = 011 | 6.60 | 7.10 | 7.60 | V |

11. Tested in production for 800 mA, 400 mA, 200 mA and 100 mA current settings for both X and Y coil.
12. Parameter guaranteed by trimming relevant OTPs in production.
13. No more than 100 cumulated hours in life time above Tw.
14. Thermal shutdown and low temperature warning are derived from thermal warning. Guaranteed by design.
15. A buffer capacitor of minimum 100 μF is needed between $V_{BB}$ and GND. Short connections to the power supply are recommended.
16. Pin $V_{DD}$ must not be used for any external supply
17. The RAM content will not be altered above this voltage.
18. External resistance value seen from pin SWI or HW2, including 1 kΩ series resistor. For the switch OPEN, the maximum allowed leakage current is represented by a minimum resistance seen from the pin.
19. While LIN is only specified for operation above 7 V $V_{BB,}$ the device can operate LIN at lower voltages down to 5.5 V. Under these conditions the LIN specific parameters are not guaranteed.

**Table 7. DC PARAMETERS**

| Symbol | Pin(s) | Parameter | Test Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| **SUPPLY AND VOLTAGE REGULATOR** | | | | | | | |
| $UV_3$ | $V_{BB}$ | Decelerated stop voltage low threshold | UV3Thr[2:0] = 100 | 6.97 | 7.50 | 8.03 | V |
| | | | UV3Thr[2:0] = 101 | 7.34 | 7.90 | 8.46 | V |
| | | | UV3Thr[2:0] = 110 | 7.71 | 8.30 | 8.89 | V |
| | | | UV3Thr[2:0] = 111 | 8.09 | 8.70 | 9.31 | V |
| $UV_1$ | $V_{BB}$ | Stop voltage high threshold, Ratio metric coupled to UV3Thr[2:0]. | UV3Thr[2:0] = 000 | 6.18 | 6.62 | 7.06 | V |
| | | | UV3Thr[2:0] = 001 | 6.60 | 7.07 | 7.54 | V |
| | | | UV3Thr[2:0] = 010 | 7.02 | 7.52 | 8.01 | V |
| | | | UV3Thr[2:0] = 011 | 7.44 | 7.97 | 8.49 | V |
| | | | UV3Thr[2:0] = 100 | 7.86 | 8.41 | 8.97 | V |
| | | | UV3Thr[2:0] = 101 | 8.28 | 8.86 | 9.45 | V |
| | | | UV3Thr[2:0] = 110 | 8.70 | 9.31 | 9.93 | V |
| | | | UV3Thr[2:0] = 111 | 9.12 | 9.76 | 10.41 | V |
| $I_{bat}$ | | Total current consumption | Unloaded outputs, $V_{BB}$ = 29 V | | 3.50 | 10.0 | mA |
| $I_{bat_s}$ | | Sleep mode current consumption | $V_{BB}$ = 5.5 V & 18 V | | | 150 | μA |
| $V_{DD}$ | $V_{DD}$ | Regulated internal supply (Note 16) | 5.5 V < $V_{BB}$ < 29 V | 3.1 | 3.3 | 3.5 | V |
| $V_{ddReset}$ | | Digital supply reset level @ power down (Note 17) | | | | 3.0 | V |
| $I_{ddLim}$ | | Current limitation | Pin shorted to ground $V_{BB}$ = 14 V | | | 85 | mA |
| **SWITCH INPUT AND HARDWIRE ADDRESS INPUT** | | | | | | | |
| $Rt_{_OFF}$ | SWI HW2 | Switch OPEN resistance (Note 18) | | 10 | | | kΩ |
| $Rt_{_ON}$ | | Switch ON resistance (Note 18) | Switch to GND or $V_{BB}$ | | | 1.9 | kΩ |
| $V_{bb_sw}$ | | $V_{BB}$ range for guaranteed operation of SWI and HW2 | | 5.5 | | 29 | V |
| $I_{lim_sw}$ | | Current limitation | Short to GND or $V_{bat}$ $V_{BB}$ = 29 V | 20 | 30 | 45 | mA |
| **HARDWIRED ADDRESS INPUTS AND TEST PIN** | | | | | | | |
| $V_{ihigh}$ | HW0 HW1 TST | Input level high | $V_{BB}$ = 14 V | 1.9 | | | V |
| $V_{ilow}$ | | Input level low | $V_{BB}$ = 14 V | | | 1.4 | V |
| $HW_{hyst}$ | | Hysteresis | $V_{BB}$ = 14 V | | 1 | | V |

11. Tested in production for 800 mA, 400 mA, 200 mA and 100 mA current settings for both X and Y coil.
12. Parameter guaranteed by trimming relevant OTPs in production.
13. No more than 100 cumulated hours in life time above Tw.
14. Thermal shutdown and low temperature warning are derived from thermal warning. Guaranteed by design.
15. A buffer capacitor of minimum 100 μF is needed between $V_{BB}$ and GND. Short connections to the power supply are recommended.
16. Pin $V_{DD}$ must not be used for any external supply
17. The RAM content will not be altered above this voltage.
18. External resistance value seen from pin SWI or HW2, including 1 kΩ series resistor. For the switch OPEN, the maximum allowed leakage current is represented by a minimum resistance seen from the pin.
19. While LIN is only specified for operation above 7 V $V_{BB,}$ the device can operate LIN at lower voltages down to 5.5 V. Under these conditions the LIN specific parameters are not guaranteed.

## AC Parameters

The AC parameters are guaranteed over junction temperature from −40 to 145°C and $V_{BB}$ in the operating range from 5.5 to 29 V, unless otherwise specified. The LIN transmitter and receiver physical layer parameters are compliant to LIN rev. 2.0 & 2.1.

**Table 8. AC PARAMETERS**

| Symbol | Pin(s) | Parameter | Test Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| **POWERUP** | | | | | | | |
| $T_{pu}$ | | Power−up time | Guaranteed by design | | | 10 | ms |
| **INTERNAL OSCILLATOR** | | | | | | | |
| $f_{osc}$ | | Frequency of internal oscillator | $V_{BB}$ = 14 V | 3.6 | 4.0 | 4.4 | MHz |
| **LIN TRANSMITTER CHARACTERISTICS ACCORDING TO LIN V2.0 & V2.1** | | | | | | | |
| D1 | LIN | Duty cycle 1 = $t_{Bus_rec(min)}$ / (2 x $t_{Bit}$); See Figure 4 | THRec(max) = 0.744 x $V_{BB}$<br>THDom(max) = 0.581 x $V_{BB}$;<br>$V_{BB}$ = 7.0 V...18 V;<br>$t_{Bit}$ = 50 μs | 0.396 | | | |
| D2 | | Duty cycle 2 = $t_{Bus_rec(max)}$ / (2 x $t_{Bit}$); See Figure 4 | THRec(min) = 0.422 x $V_{BB}$<br>THDom(min) = 0.284 x $V_{BB}$;<br>$V_{BB}$ = 7.6 V...18 V;<br>$t_{Bit}$ = 50 μs | | | 0.581 | |
| **LIN RECEIVER CHARACTERISTICS ACCORDING TO LIN V2.0 & V2.1** | | | | | | | |
| trx_pdr | LIN | Propagation delay bus dominant to RxD = low | $V_{BB}$ = 7.0 V & 18 V;<br>See Figure 4 | | | 6 | μs |
| trx_pdf | | Propagation delay bus recessive to RxD = high | $V_{BB}$ = 7.0 V & 18 V;<br>See Figure 4 | | | 6 | μs |
| trx_sym | | Symmetry of receiver propagation delay | trx_pdr − trx_pdf | −2 | | +2 | μs |
| **SWITCH INPUT AND HARDWIRE ADDRESS INPUT** | | | | | | | |
| $T_{sw}$ | SWI HW2 | Scan pulse period (Note 20) | $V_{BB}$ = 14 V | | 1024 | | μs |
| $T_{sw_on}$ | | Scan pulse duration (Note 20) | $V_{BB}$ = 14 V | | 128 | | μs |
| **MOTORDRIVER** | | | | | | | |
| $F_{pwm}$ | MOTx | PWM frequency (Note 20) | PWMfreq = 0 (Note 21) | 20.6 | 22.8 | 25.0 | kHz |
| | | | PWMfreq = 1 (Note 21) | 41.2 | 45.6 | 50.0 | kHz |
| $F_{jit_depth}$ | | PWM jitter modulation depth | PWMJen = 1 (Note 21) | | 10 | | % |
| $T_{brise}$ | | Turn−on transient time | Between 10% and 90% | | 170 | | ns |
| $T_{bfall}$ | | Turn−off transient time | | | 140 | | ns |
| $T_{stab}$ | | Run current stabilization time (Note 20) | TStab[1:0] = 00 | 14.4 | 16 | 17.6 | ms |
| | | | TStab[1:0] = 01 | 19.8 | 22 | 24.2 | ms |
| | | | TStab[1:0] = 10 | 25.2 | 28 | 30.8 | ms |
| | | | TStab[1:0] = 11 | 28.8 | 32 | 35.2 | ms |
| **SUPPLY** | | | | | | | |
| $T_{UV1_deb}$ | VBB | UV1 level debounce time (Note 20) | UV3debT = 0 | | 96 | | μs |
| | | | UV3debT = 1 | | 256 | | μs |

20. Derived from the internal oscillator
21. See `SetMotorParam` and PWM Regulator

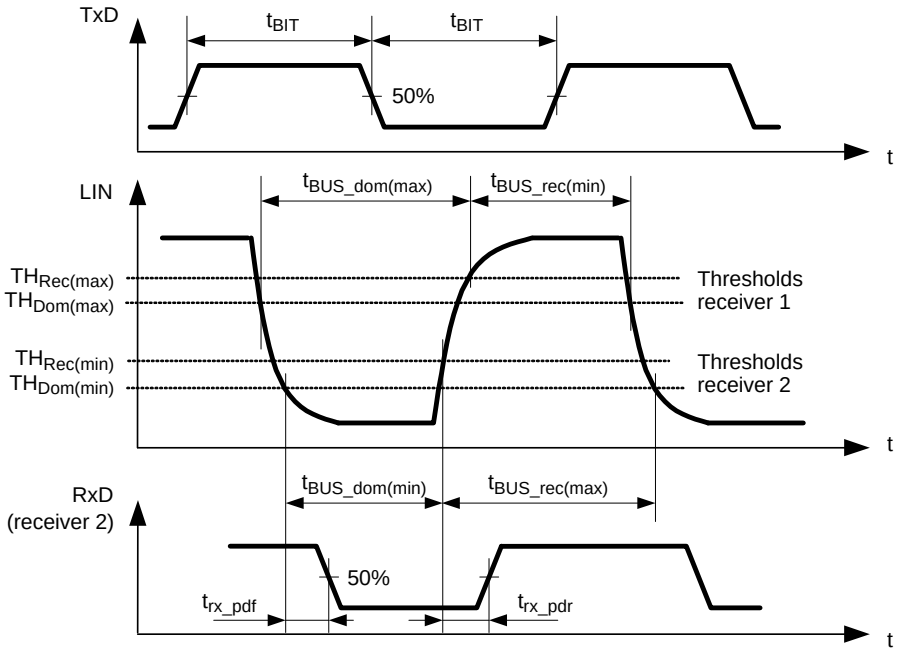



**Figure 4. Timing Diagram for AC Characteristics According to LIN 2.0 & 2.1**

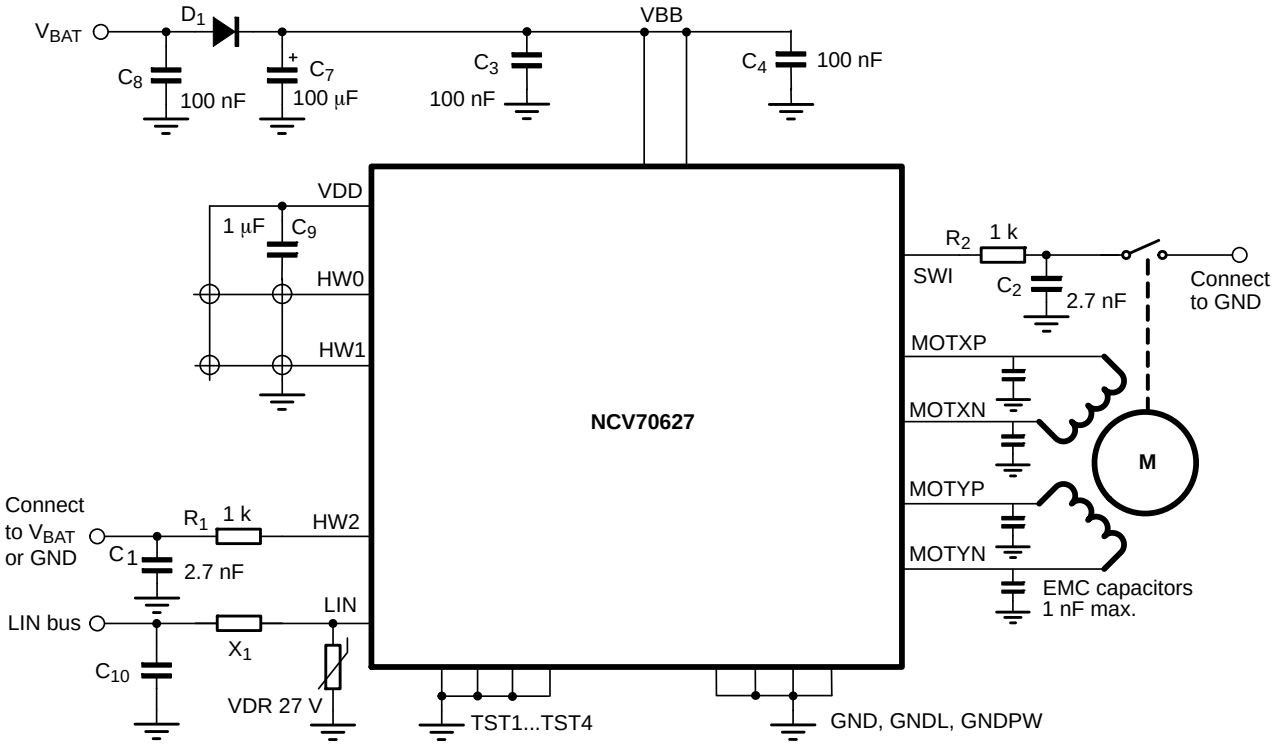



**Figure 5. Typical Application**

NOTES: All resistors are ± 5%, 1/4 W
$C_1$, $C_2$ minimum value is 2.7 nF, maximum value is 10 nF
Depending on the application, the ESR value and working voltage of $C_7$ must be carefully chosen
$C_3$ and $C_4$ must be close to pins $V_{BB}$ and coupled GND directly
$C_9$ must be a ceramic capacitor to assure low ESR
$C_{10}$ is placed for system level EMC reasons; value depends on EMC requirements of the application, recommended 200 pF
$X_1$ is placed for system level EMC and ESD reasons. Use e.g. BLM18AG601SN1D 600 OHM or resistor 50 Ω

## Positioning Parameters

### Stepping Modes

One of four possible stepping modes can be programmed:

- Half−stepping
- 1/4 micro−stepping
- 1/8 micro−stepping
- 1/16 micro−stepping

### Maximum Velocity

For each stepping mode, the maximum velocity Vmax can be programmed to 16 possible values given in the table below.

The accuracy of Vmax is derived from the internal oscillator. Under special circumstances it is possible to change the Vmax parameter while a motion is ongoing. All 16 entries for the Vmax parameter are divided into four groups. When changing Vmax during a motion the application must take care that the new Vmax parameter stays within the same group.

**Table 9. MAXIMUM VELOCITY SELECTION TABLE**

| Vmax Index | | | | Stepping Mode | | | |
|---|---|---|---|---|---|---|---|
| Hex | Dec | Vmax (full step/s) | Group | Half−stepping (half−step/s) | 1/4th Micro−stepping (micro−step/s) | 1/8th Micro−stepping (micro−step/s) | 1/16th Micro−stepping (micro−step/s) |
| 0 | 0 | 99 | A | 197 | 395 | 790 | 1579 |
| 1 | 1 | 136 | B | 273 | 546 | 1091 | 2182 |
| 2 | 2 | 167 | | 334 | 668 | 1335 | 2670 |
| 3 | 3 | 197 | | 395 | 790 | 1579 | 3159 |
| 4 | 4 | 213 | | 425 | 851 | 1701 | 3403 |
| 5 | 5 | 228 | | 456 | 912 | 1823 | 3647 |
| 6 | 6 | 243 | | 486 | 973 | 1945 | 3891 |
| 7 | 7 | 273 | C | 546 | 1091 | 2182 | 4364 |
| 8 | 8 | 303 | | 607 | 1213 | 2426 | 4852 |
| 9 | 9 | 334 | | 668 | 1335 | 2670 | 5341 |
| A | 10 | 364 | | 729 | 1457 | 2914 | 5829 |
| B | 11 | 395 | | 790 | 1579 | 3159 | 6317 |
| C | 12 | 456 | | 912 | 1823 | 3647 | 7294 |
| D | 13 | 546 | D | 1091 | 2182 | 4364 | 8728 |
| E | 14 | 729 | | 1457 | 2914 | 5829 | 11658 |
| F | 15 | 973 | | 1945 | 3891 | 7782 | 15564 |

### Minimum Velocity

Once the maximum velocity is chosen, 16 possible values can be programmed for the minimum velocity Vmin. The table below provides the obtainable values in full−step/s. The accuracy of Vmin is derived from the internal oscillator. It is not recommended to change the Vmin while a motion is ongoing.

**Table 10. OBTAINABLE VALUES IN FULL−STEP/s FOR THE MINIMUM VELOCITY**

| Vmin Index | | Vmax Factor | Vmax (Full−step/s) | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | A | B | | | | | | C | | | | | | D | | |
| Hex | Dec | | 99 | 136 | 167 | 197 | 213 | 228 | 243 | 273 | 303 | 334 | 364 | 395 | 456 | 546 | 729 | 973 |
| 0 | 0 | 1 | 99 | 136 | 167 | 197 | 213 | 228 | 243 | 273 | 303 | 334 | 364 | 395 | 456 | 546 | 729 | 973 |
| 1 | 1 | 1/32 | 3 | 4 | 5 | 6 | 6 | 7 | 7 | 8 | 8 | 10 | 10 | 11 | 13 | 15 | 19 | 27 |
| 2 | 2 | 2/32 | 6 | 8 | 10 | 11 | 12 | 13 | 14 | 15 | 17 | 19 | 21 | 23 | 27 | 31 | 42 | 57 |
| 3 | 3 | 3/32 | 9 | 12 | 15 | 18 | 19 | 21 | 22 | 25 | 27 | 31 | 32 | 36 | 42 | 50 | 65 | 88 |
| 4 | 4 | 4/32 | 12 | 16 | 20 | 24 | 26 | 28 | 30 | 32 | 36 | 40 | 44 | 48 | 55 | 65 | 88 | 118 |
| 5 | 5 | 5/32 | 15 | 21 | 26 | 31 | 32 | 35 | 37 | 42 | 46 | 51 | 55 | 61 | 71 | 84 | 111 | 149 |
| 6 | 6 | 6/32 | 18 | 25 | 31 | 36 | 39 | 42 | 45 | 50 | 55 | 61 | 67 | 72 | 84 | 99 | 134 | 179 |
| 7 | 7 | 7/32 | 21 | 30 | 36 | 43 | 46 | 50 | 52 | 59 | 65 | 72 | 78 | 86 | 99 | 118 | 156 | 210 |
| 8 | 8 | 8/32 | 24 | 33 | 41 | 49 | 52 | 56 | 60 | 67 | 74 | 82 | 90 | 97 | 113 | 134 | 179 | 240 |
| 9 | 9 | 9/32 | 28 | 38 | 47 | 55 | 59 | 64 | 68 | 76 | 84 | 93 | 101 | 111 | 128 | 153 | 202 | 271 |
| A | 10 | 10/32 | 31 | 42 | 51 | 61 | 66 | 71 | 75 | 84 | 93 | 103 | 113 | 122 | 141 | 168 | 225 | 301 |
| B | 11 | 11/32 | 34 | 47 | 57 | 68 | 72 | 78 | 83 | 93 | 103 | 114 | 124 | 135 | 156 | 187 | 248 | 332 |
| C | 12 | 12/32 | 37 | 51 | 62 | 73 | 79 | 85 | 91 | 101 | 113 | 124 | 135 | 147 | 170 | 202 | 271 | 362 |
| D | 13 | 13/32 | 40 | 55 | 68 | 80 | 86 | 93 | 98 | 111 | 122 | 135 | 147 | 160 | 185 | 221 | 294 | 393 |
| E | 14 | 14/32 | 43 | 59 | 72 | 86 | 93 | 99 | 106 | 118 | 132 | 145 | 158 | 172 | 198 | 237 | 317 | 423 |
| F | 15 | 15/32 | 46 | 64 | 78 | 93 | 99 | 107 | 113 | 128 | 141 | 156 | 170 | 185 | 214 | 256 | 340 | 454 |

NOTES: The Vmax factor is an approximation.
In case of motion without acceleration (**AccShape = 1**) the length of the steps = 1/**Vmin.** In case of accelerated motion (**AccShape** = 0) the length of the first step is shorter than 1/**Vmin** depending of **Vmin, Vmax** and **Acc**.

### Acceleration and Deceleration

Sixteen possible values can be programmed for Acc (acceleration and deceleration between Vmin and Vmax). The table below provides the obtainable values in full−step/s$^2$. One observes restrictions for some combinations of acceleration index and maximum speed. It is not recommended to change the Acc value while a motion is ongoing.

The accuracy of Acc is derived from the internal oscillator.

**Table 11. ACCELERATION AND DECELERATION SELECTION TABLE**

| Vmax (FS/s) → | | 99 | 136 | 167 | 197 | 213 | 228 | 243 | 273 | 303 | 334 | 364 | 395 | 456 | 546 | 729 | 973 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓ Acc Index | | | | | | | | | | | | | | | | | |
| Hex | Dec | Acceleration (Full−step/s$^2$) | | | | | | | | | | | | | | | |
| 0 | 0 | 49 | | | | | | | 106 | | | | | | 473 | | |
| 1 | 1 | 218 | | | | | | | | | | | | | 735 | | |
| 2 | 2 | 1004 | | | | | | | | | | | | | | | |
| 3 | 3 | 3609 | | | | | | | | | | | | | | | |
| 4 | 4 | 6228 | | | | | | | | | | | | | | | |
| 5 | 5 | 8848 | | | | | | | | | | | | | | | |
| 6 | 6 | 11409 | | | | | | | | | | | | | | | |
| 7 | 7 | 13970 | | | | | | | | | | | | | | | |
| 8 | 8 | 16531 | | | | | | | | | | | | | | | |
| 9 | 9 | 14785 | 19092 | | | | | | | | | | | | | | |
| A | 10 | | 21886 | | | | | | | | | | | | | | |
| B | 11 | | 24447 | | | | | | | | | | | | | | |
| C | 12 | | 27008 | | | | | | | | | | | | | | |
| D | 13 | | 29570 | | | | | | | | | | | | | | |
| E | 14 | | 29570 | | | | | | 34925 | | | | | | | | |
| F | 15 | | | | | | | | 40047 | | | | | | | | |

The formula to compute the number of equivalent full−steps during acceleration phase is:

$$Nstep = \frac{Vmax^2 - Vmin^2}{2 \times Acc}$$

### Positioning

The position programmed in commands `SetPosition` is given as a number of (micro−) steps. According to the chosen stepping mode, the internal position words is aligned as described in the table below. When using command `SetPositionShort` the position is given in numbers of half steps, while the Secure Position is given in a number of two Full Steps. The position data is aligned automatically.

**Table 12. POSITION WORD ALIGNMENT**

| Stepping Mode | Position Word: `Pos[15:0]` | | | | | | | | | | | | | | | | Shift |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1/16$^{th}$ | **S** | B14 | B13 | B12 | B11 | B10 | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | **LSB** | No shift |
| 1/8$^{th}$ | **S** | B13 | B12 | B11 | B10 | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | **LSB** | 0 | 1−bit left ⇔ ×2 |
| 1/4$^{th}$ | **S** | B12 | B11 | B10 | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | **LSB** | 0 | 0 | 2−bit left ⇔ ×4 |
| Half−stepping | **S** | B11 | B10 | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | **LSB** | 0 | 0 | 0 | 3−bit left ⇔ ×8 |
| Position Short | **S** | **S** | **S** | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | **LSB** | 0 | 0 | 0 | No shift |
| Secure Position | **S** | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | **LSB** | 0 | 0 | 0 | 0 | 0 | No shift |

NOTES: LSB: Least Significant Bit
S: Sign bit

### Position Ranges

A position is coded by using the binary two's complement format. According to the positioning commands used and to the chosen stepping mode, the position range will be as shown in the following table.

**Table 13. POSITION RANGE**

| Command | Stepping Mode | Position Range | Full Range Excursion | Number of Bits in micro stepping |
|---|---|---|---|---|
| SetPosition | Half−stepping | −4096 to +4095 | 8192 half−steps | 13 |
| | 1/4th micro−stepping | −8192 to +8191 | 16384 micro−steps | 14 |
| | 1/8th micro−stepping | −16384 to +16383 | 32768 micro−steps | 15 |
| | 1/16th micro−stepping | −32768 to +32767 | 65536 micro−steps | 16 |
| SetPositionShort | Half−stepping | −1024 to +1023 | 2048 half−steps | 11 |
| | 1/4th micro−stepping | −2048 to +2047 | 4096 micro−steps | 12 |
| | 1/8th micro−stepping | −4096 to +4095 | 8192 micro−steps | 13 |
| | 1/16th micro−stepping | −8192 to +8191 | 16384 micro−steps | 14 |

When using the command SetPosition, although coded on 16 bits, the position word is shifted to the left by a certain number of bits, according to the stepping mode. SetPositonShort is only coded on 11 bits.

### Secure Position

A secure position can be programmed. It is mapped to the positioned full range but coded in 11−bits, thus having a lower resolution than normal positions, as shown in the following table. See also command GotoSecurePosition and LIN lost behavior.

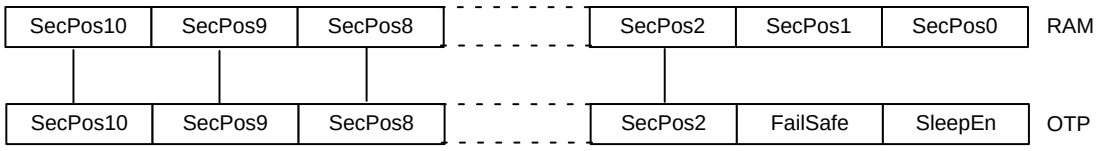
**Table 14. SECURE POSITION**

| Stepping Mode | Secure Position Resolution |
|---|---|
| Half−stepping | 4 half−steps |
| 1/4th micro−stepping | 8 micro−steps (1/4th) |
| 1/8th micro−stepping | 16 micro−steps (1/8th) |
| 1/16th micro−stepping | 32 micro−steps (1/16th) |

### Important

NOTES: For the FailSafe functionality and SetDualPosition command, the secure position is disabled in case the programmed value has the code "10000000000" (0x400 or most negative position). For the GotoSecurePosition command there is no disabling possible. By receiving this command the secure positioning is always executed, even when the secure position has the value 0x400.

The resolution of the secure position is limited to 9 bit at start−up. The OTP register is copied in RAM as illustrated below. The RAM bits SecPos1 and SecPos0 are set to 0.




### Shaft

A shaft bit, which can be programmed in OTP or with command SetMotorParam, defines whether a positive motion is a clockwise (CW) or counter−clockwise rotation (CCW) (an outer or an inner motion for linear actuators):

- Shaft = 0 ⇒ MOTXP is used as positive pin of the X coil, while MOTXN is the negative one.
- Shaft = 1 ⇒ opposite situation

## Structural Description

Refer to the Block Diagram in Figure 1.

### Stepper Motordriver

The Motordriver receives the control signals from the control logic. The main features are:

- Two H−bridges, designed to drive a stepper motor with two separated coils. Each coil (X and Y) is driven by one H−bridge, and the driver controls the currents flowing through the coils. The rotational position of the rotor, in unloaded condition, is defined by the ratio of current flowing in X and Y. The torque of the stepper motor when unloaded is controlled by the magnitude of the currents in X and Y.
- The control block for the H−bridges, including the PWM control, the synchronous rectification and the internal current sensing circuitry.
- Two pre−scale 4−bit DAC's to set the maximum magnitude of the current through X and Y.
- Two DAC's to set the correct current ratio through X and Y.
- A boost function that increases the current during cold conditions.

Battery voltage monitoring is also performed by this block, which provides the required information to the control logic part. The same applies for detection and reporting of an electrical problem that could occur on the coils.

### Control Logic (Position Controller and Main Control)

The control logic block stores the information provided by the LIN interface (in a RAM or an OTP memory) and digitally controls the positioning of the stepper motor in terms of speed and acceleration, by feeding the right signals to the motor driver state machine.

It will take into account the successive positioning commands to properly initiate or stop the stepper motor in order to reach the set point in a minimum time.

It also receives feedback from the motor driver part in order to manage possible problems and decide on internal actions and reporting to the LIN interface.

### Motion Detection

Motion detection is based on the back−emf generated internally in the running motor. When the motor is blocked, e.g. when it hits the end position, the velocity, and as a result also the generated back−emf, is disturbed. The NCV70627 senses the back−emf and compares the value with an independent threshold level. If the back−emf becomes lower than the threshold, the running motor is stopped.

### LIN Interface

The LIN interface implements the physical layer and the MAC and LLC layers according to the OSI reference model. It provides and gets information to and from the control logic block, in order to drive the stepper motor, to configure the way this motor must be driven, or to get information such as actual position or diagnosis (temperature, battery voltage, electrical status...) and pass it to the LIN master node.

### Miscellaneous

The NCV70627 also contains the following:

- An internal oscillator, needed for the LIN protocol handler as well as the control logic and the PWM control of the motor driver.
- An internal trimmed voltage source for precise referencing.
- A protection block featuring a thermal shutdown and a power−on−reset circuit.
- A 3.3 V regulator (from the battery supply) to supply the internal logic circuitry.

## Functions Description

This chapter describes the following functional blocks in more detail:

- Position controller
- Main control and register, OTP memory + ROM
- Motor driver

The Motion detection and LIN controller are discussed in separate chapters.

## Position Controller

### Positioning and Motion Control

A positioning command will produce a motion as illustrated in Figure 6. A motion starts with an acceleration phase from minimum velocity (Vmin) to maximum velocity (Vmax) and ends with a symmetrical deceleration. This is defined by the control logic according to the position required by the application and the parameters programmed by the application during the configuration phase. The current in the coils is also programmable.

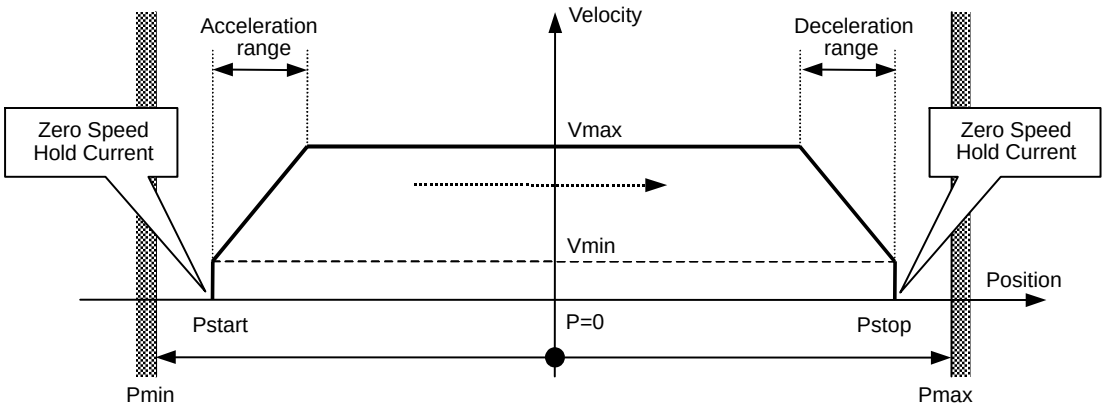


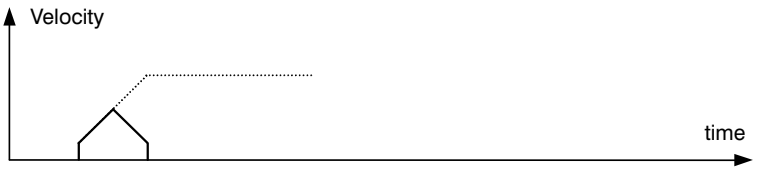
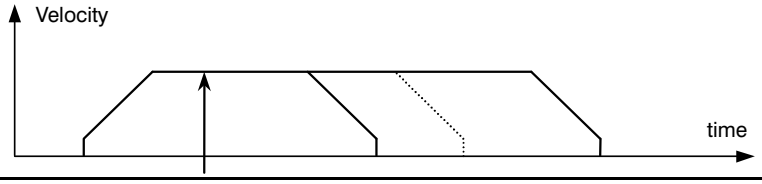
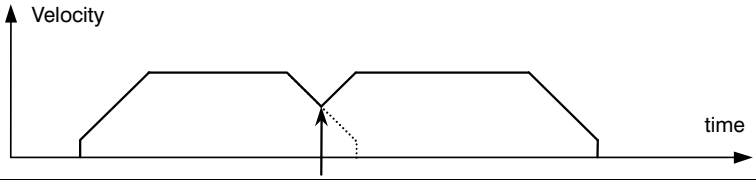
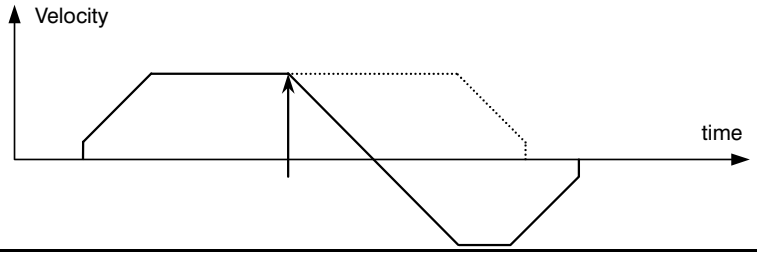
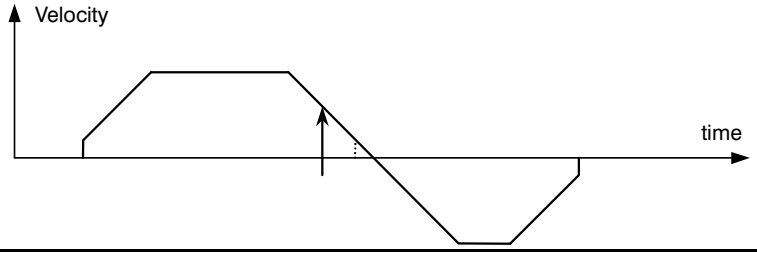
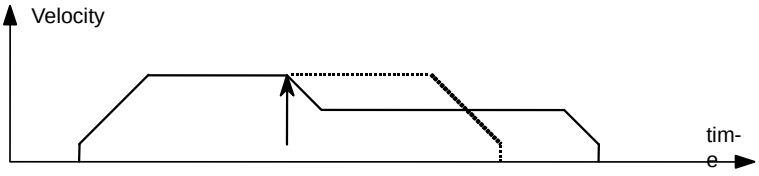

**Figure 6. Position and Motion Control**

**Table 15. POSITION RELATED PARAMETERS**

| Parameter | Reference |
|---|---|
| Pmax – Pmin | See Positioning |
| Zero Speed Hold Current | See Ihold |
| Maximum Current | See Irun |
| Acceleration and Deceleration | See Acceleration and Deceleration |
| Vmin | See Minimum Velocity |
| Vmax | See Maximum Velocity |
| Stabilization Time | See Stabilization Time |

Different positioning examples are shown in the next table.

**Table 16. POSITIONING EXAMPLES**

| | |
|---|---|
| Short motion. |  |
| New positioning command in same direction, shorter or longer, while a motion is running at maximum velocity. |  |
| New positioning command in same direction while in deceleration phase (Note 22)<br>Note: there is no wait time between the deceleration phase and the new acceleration phase. |  |
| New positioning command in reverse direction while motion is running at maximum velocity. |  |
| New positioning command in reverse direction while in deceleration phase. |  |
| New velocity Vmax programming while motion is running. |  |

22. Reaching the end position is always guaranteed, however velocity rounding errors might occur. The device is automatically compensating the position error. The velocity rounding error will be removed at Vmin (e.g. at end of acceleration or when AccShape=1) by a corrective motion action.

### Dual Positioning

A SetDualPosition command allows the user to perform a positioning using two different velocities. The first motion is done with the specified Vmin and Vmax velocities in the SetDualPosition command, with the acceleration (deceleration) parameter already in RAM, to a position Pos1[15:0] also specified in SetDualPosition.

Then a second relative motion to a physical position Pos1[15:0] + Pos2[15:0] is done at the specified Vmin velocity in the SetDualPosition command (no acceleration). Once the second motion is achieved, the ActPos register is reset to zero, whereas TagPos register is not changed.

When the Secure position is enabled, after the dual positioning, the secure positioning is executed. The figure below gives a detailed overview of the dual positioning function. After the dual positioning is executed an internal flag is set to indicate the NCV70627 is referenced.

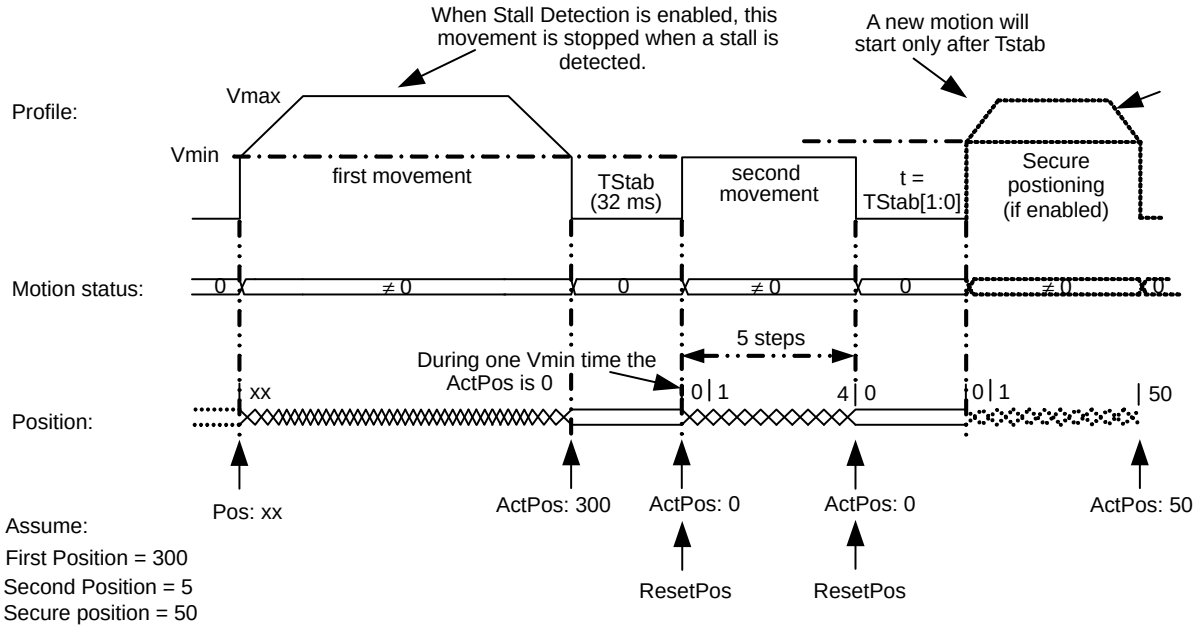



**Figure 7. Dual Position**

**Remark:** This operation cannot be interrupted or influenced by any further command unless the occurrence of the conditions driving to a motor shutdown or by a `HardStop` command. Sending a `SetDualPosition` command while a motion is already ongoing is not recommended.

23. The priority encoder is describing the management of states and commands.
24. A DualPosition sequence starts by setting TagPos buffer register to SecPos value, provided secure position is enabled otherwise TagPos is reset to zero. If a SetPosition(Short) command is issued during a DualPosition sequence, it will be kept in the position buffer memory and executed afterwards. This applies also for the commands Sleep, SetPosParam and GotoSecurePosition.
25. Commands such as GetActualPos or GetStatus will be executed while a Dual Positioning is running. This applies also for a dynamic ID assignment LIN frame.
26. The Pos1, Pos2, Vmax and Vmin values programmed in a `SetDualPosition` command apply only for this sequence. All other motion parameters are used from the RAM registers (programmed for instance by a former SetMotorParam command). After the DualPosition motion is completed, the former Vmin and Vmax become active again.
27. Commands ResetPosition, SetDualPosition, and SoftStop will be ignored while a DualPosition sequence is ongoing, and will not be executed afterwards.
28. Recommendation: a SetMotorParam command should not be sent during a `SetDualPosition` sequence: all the motion parameters defined in the command, except Vmin and Vmax, become active immediately.
29. When during the Dual positioning an under voltage UV2 or UV3 happens, the motor will stop (hardstop for UV2 or softstop for UV3). The device will go into the under−voltage and autarkic operational handler function (refer to battery voltage management and autarkic function). Especially for the dual positioning it should be stated that after passing the UV1 level the motion is continued with the parameters Vmax, Vmin and Acceleration from the SetMotorParam command and not from the SetDualPosition command.
30. After the first motion of the dual positioning there is always a fixed stabilization time of 32 ms applied afterwards. After the second motion the programmed stabilization time TStab[1..0] is applied.

### Position Periodicity

Depending on the stepping mode the position can range from −4096 to +4095 in half−step to −32768 to +32767 in 1/16th micro−stepping mode. One can project all these positions lying on a circle. When executing the command `SetPosition`, the position controller will set the movement direction in such a way that the traveled distance is minimal.

The figure below illustrates that the moving direction going from `ActPos` = +30000 to `TagPos` = –30000 is clockwise.

If a counter clockwise motion is required in this example, several consecutive `SetPosition` commands can be used.

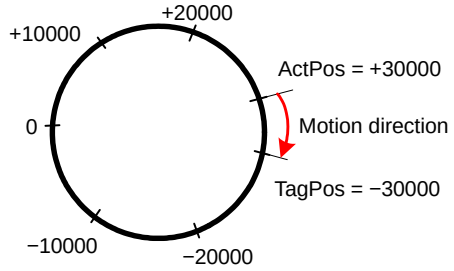



**Figure 8. Motion Direction is Function of Difference between ActPos and TagPos**

### Hardwired Address HW2

In the drawing below, a simplified schematic diagram is shown of the HW2 comparator circuit.

The HW2 pin is sensed via 2 switches. The DriveHS and DriveLS control lines are alternatively closing the top and bottom switch connecting HW2 pin with a current to resistor converter. Closing $S_{TOP}$ (DriveHS = 1) will sense a current to GND. In that case the top I → R converter output is low, via the closed passing switch $S_{PASS_T}$ this signal is fed to the "R" comparator which output HW2_Cmp is high. Closing bottom switch $S_{BOT}$ (DriveLS = 1) will sense a current to $V_{BAT}$. The corresponding I → R converter output is low and via $S_{PASS_B}$ fed to the comparator. The output HW2_Cmp will be high.

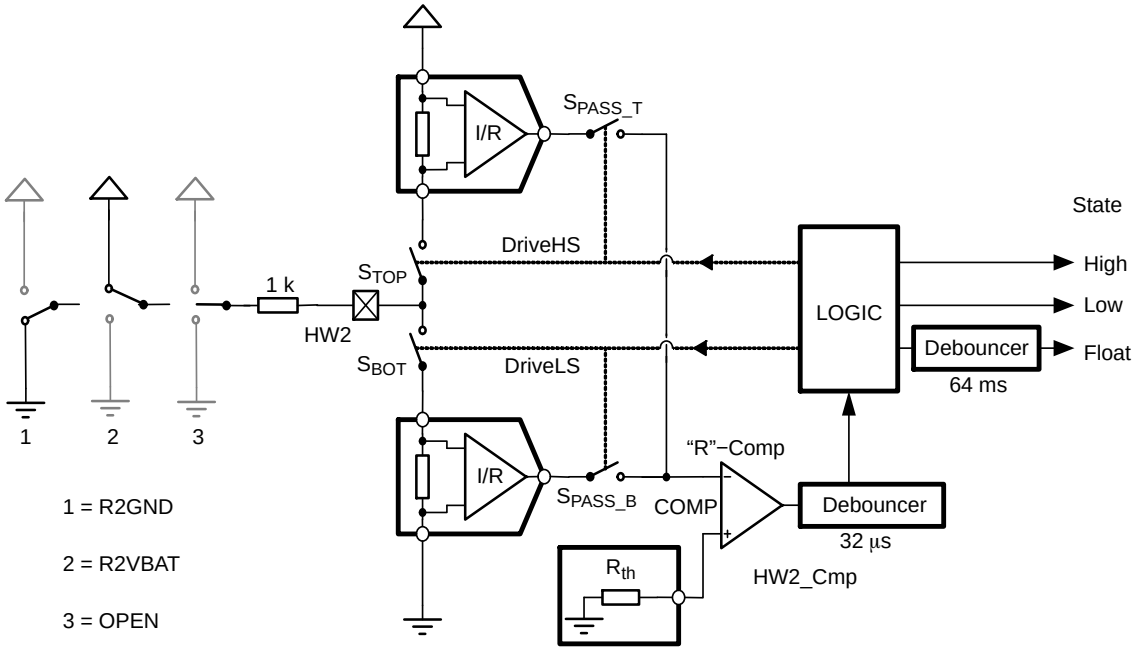



**Figure 9.**

3 cases can be distinguished (see also Figure 9 above):

- HW2 is connected to ground: R2GND or drawing 1
- HW2 is connected to VBAT: R2VBAT or drawing 2
- HW2 is floating: OPEN or drawing 3

**Table 17. STATE DIAGRAM OF THE HW2 COMPARATOR**

| Previous State | DriveLS | DriveHS | HW2_Cmp | New State | Condition | Drawing |
|---|---|---|---|---|---|---|
| Float | 1 | 0 | 0 | Float | R2GND or OPEN | 1 or 3 |
| Float | 1 | 0 | 1 | High | R2VBAT | 2 |
| Float | 0 | 1 | 0 | Float | R2VBAT or OPEN | 2 or 3 |
| Float | 0 | 1 | 1 | Low | R2GND | 1 |
| Low | 1 | 0 | 0 | Low | R2GND or OPEN | 1 or 3 |
| Low | 1 | 0 | 1 | High | R2VBAT | 2 |
| Low | 0 | 1 | 0 | Float | R2VBAT or OPEN | 2 or 3 |
| Low | 0 | 1 | 1 | Low | R2GND | 1 |
| High | 1 | 0 | 0 | Float | R2GND or OPEN | 1 or 3 |
| High | 1 | 0 | 1 | High | R2VBAT | 2 |
| High | 0 | 1 | 0 | High | R2VBAT or OPEN | 2 or 3 |
| High | 0 | 1 | 1 | Low | R2GND | 1 |

The logic is controlling the correct sequence in closing the switches and in interpreting the 64 μs debounced HW2_Cmp output accordingly. The output of this small state−machine is corresponding to:

- High or address = 1
- Low or address = 0
- Floating

As illustrated in the table above (Table 17), the state is depending on the previous state, the condition of the 2 switch controls (DriveLS and DriveHS) and the output of HW2_Cmp. Figure 10 shows an example of a practical case where a connection to VBAT is interrupted.

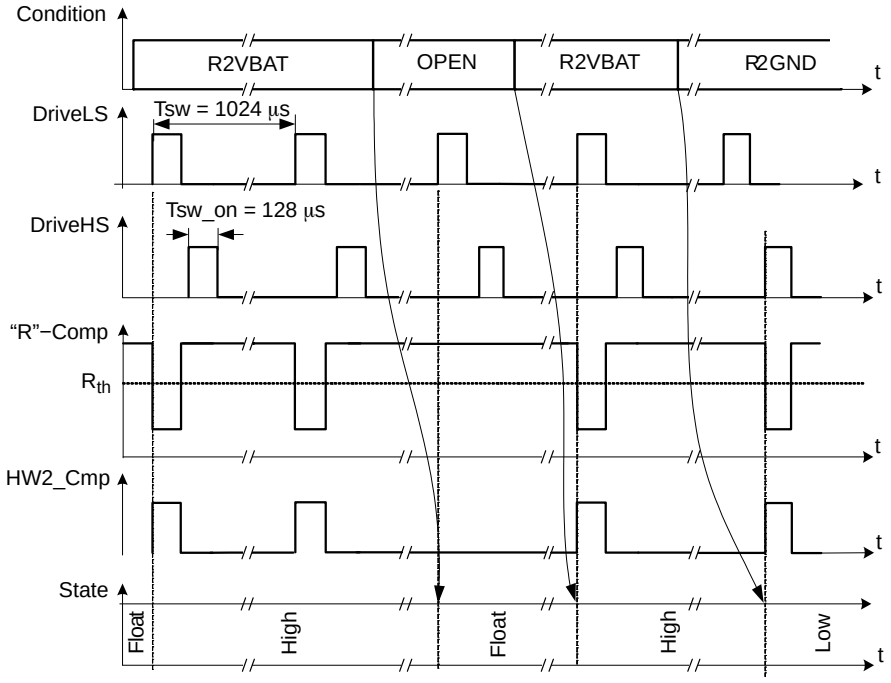



**Figure 10. Timing Diagram Showing the Change in State for HW2 Comparator**

### R2VBAT

A resistor is connected between VBAT and HW2. Every 1024 μs $S_{BOT}$ is closed and a current is sensed. The output of the I ⇒ R converter is low and the HW2_Cmp output is high. Assuming the previous state was floating, the internal logic will interpret this as a change of state and the new state will be high (see also Table 17). The next time $S_{BOT}$ is closed the same conditions are observed. The previous state was high so based on Table 17 the new state remains unchanged. This high state will be interpreted as HW2 address = 1.

### OPEN

In case the HW2 connection is lost (broken wire, bad contact in connector) the next time $S_{BOT}$ is closed, this will be sensed. There will be no current, the output of the corresponding I ⇒ R converter is high and the HW2_Cmp will be low. The previous state was high. Based in Table 17 one can see that the state changes to float. This will trigger a motion to secure position after a debounce time of 64 ms, which prevents false triggering in case of micro−interruptions of the power supply.

### R2GND

If a resistor is connected between HW2 and the GND, a current is sensed every 1024 μs when $S_{TOP}$ is closed. The output of the top I ⇒ R converter is low and as a result the HW2_Cmp output switches to high. Again based on the stated diagram in Table 17 one can see that the state will change to Low. This low state will be interpreted as HW2 address = 0.

### External Switch SWI

As illustrated in Figure 11 the SWI comparator is almost identical to HW2. The major difference is in the limited number of states. Only open or closed is recognized leading to respectively ESW = 0 and ESW = 1.

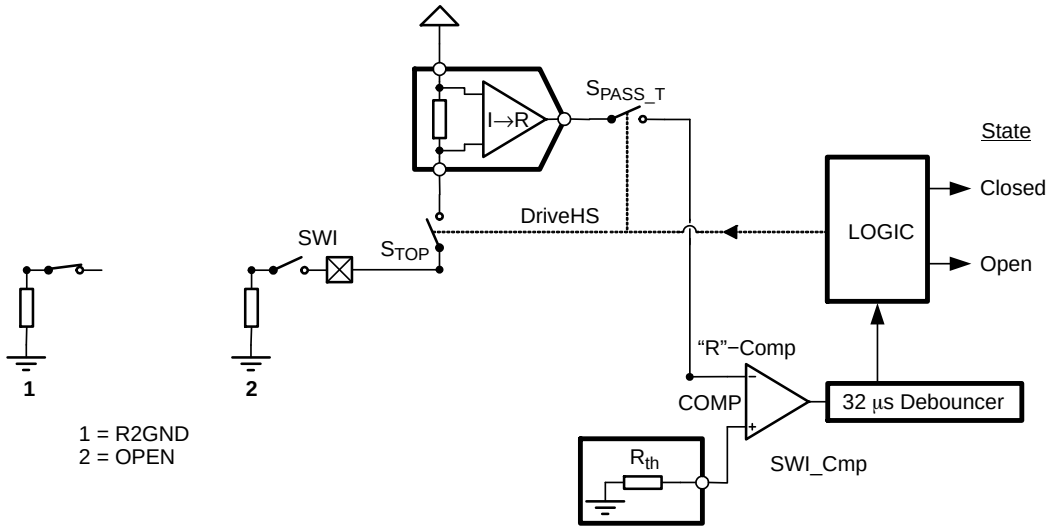



**Figure 11. Simplified Schematic Diagram of the SWI Comparator**

As illustrated in the drawing above, a change in state is always synchronized with DriveHS or DriveLS. The same synchronization is valid for updating the internal position register. This means that after every current pulse (or closing of $S_{TOP}$ or $S_{BOT}$) the state of the position switch together with the corresponding position is memorized.

The GetActualPos command reads back the <ActPos> register and the status of ESW. In this way the master node may get synchronous information about the state of the switch together with the position of the motor. See Table 18 below.

**Table 18. GetActualPos LIN COMMAND**

| Reading Frame | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Structure | | | | | | | |
| Byte | Content | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 1 | 0 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | ESW | AD[6:0] | | | | | | |
| 2 | Data 2 | ActPos[15:8] | | | | | | | |
| 3 | Data 3 | ActPos[7:0] | | | | | | | |
| 4 | Data 4 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |
| 5 | Checksum | Checksum over data | | | | | | | |

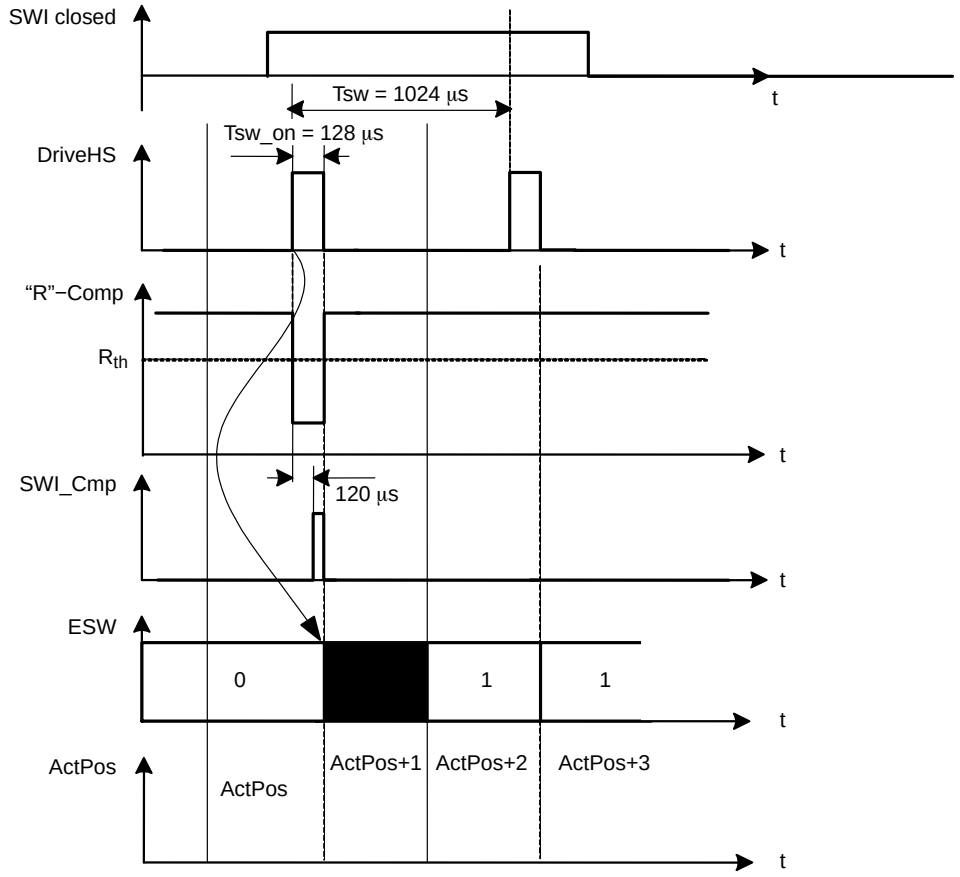



**Figure 12. Timing Diagram Showing the Change in States for SWI Comparator**

## Main Control and Register, OTP memory + ROM

### Power−up Phase

Power−up phase of the NCV70627 will not exceed 10 ms. After this phase, the NCV70627 is in standby mode, ready to receive LIN messages and execute the associated commands. After power−up, the registers and flags are in the reset state, while some of them are being loaded with the OTP memory content (see Table 21: RAM Registers).

### Reset

After power−up, or after a reset occurrence (e.g. a micro−cut on pin $V_{BB}$ has made $V_{DD}$ to go below VddReset level), the H−bridges will be in high−impedance mode, and the registers and flags will be in a predetermined position. This is documented in Table 21: RAM Registers and Table 22: Flags Table.

### Soft−stop

A soft−stop is an immediate interruption of a motion, but with a deceleration phase. At the end of this action, the register <TagPos> is loaded with the value contained in register <ActPos>, (see Table 21: Ram Registers). The circuit is then ready to execute a new positioning command, provided thermal and electrical conditions allow for it.

### Sleep Mode

When entering sleep mode, the stepper−motor can be driven to its secure position. After which, the circuit is completely powered down, apart from the LIN receiver, which remains active to detect a dominant state on the bus. In case sleep mode is entered while a motion is ongoing, a transition will occur towards secure position as described in Positioning and Motion Control provided <SecPos> is enabled. Otherwise, <SoftStop> is performed.

Sleep mode can be entered in the following cases:

- The circuit receives a LIN frame with identifier **0x3C** and first data byte containing **0x00**, as required by LIN specification rev 1.3 and <SleepEn> bit = 1. See also Sleep in the LIN Application Command section.
- In case the <SleepEn> bit = 1 and the LIN bus remains inactive (or is lost) during more than 25000 time slots (1.30 s at 19.2 kbit/s), a time−out signal switches the circuit to sleep mode.

The circuit will return to normal mode if a valid LIN frame is received (this valid frame can be addressed to another slave).

### Thermal Shutdown Mode

When thermal shutdown occurs, the circuit performs a <SoftStop> command and goes to motor shutdown mode (see Figure 13: State Diagram Temperature Management).

**Temperature Management**

The NCV70627 monitors temperature by means of two thresholds and one shutdown level, as illustrated in the state diagram and illustration of Figure 13: State Diagram Temperature Management below. The only condition to reset flags <TW> and <TSD> (respectively thermal warning and thermal shutdown) is to be at a temperature lower than Ttw and to get the occurrence of a GetStatus or a GetFullStatus LIN frame.

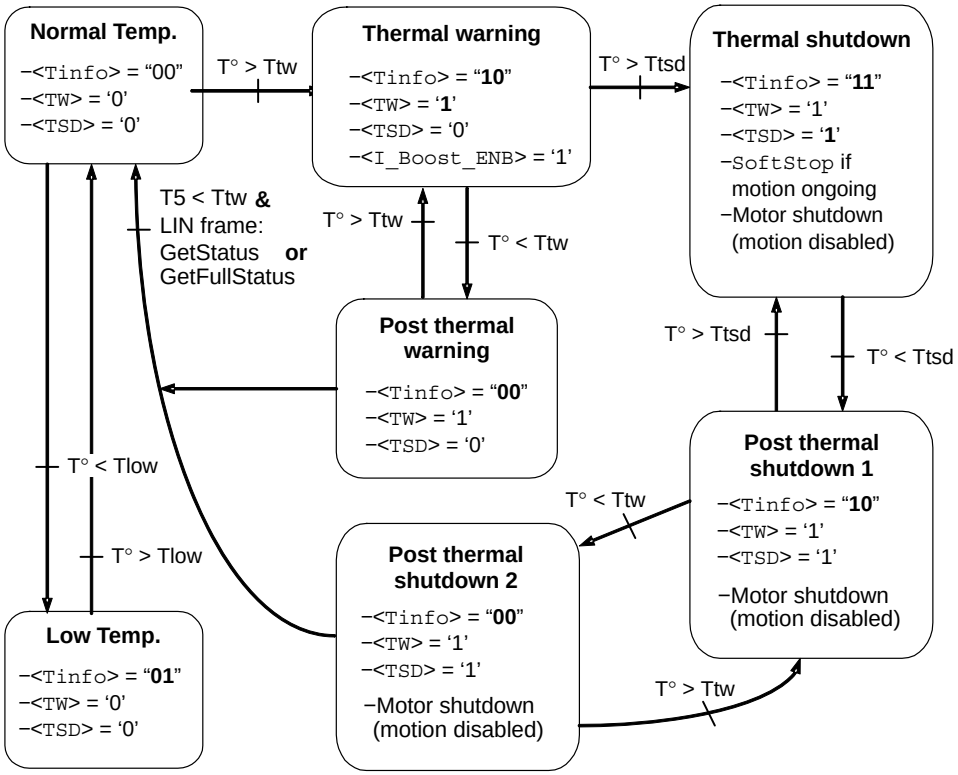



**Figure 13. State Diagram Temperature Management**

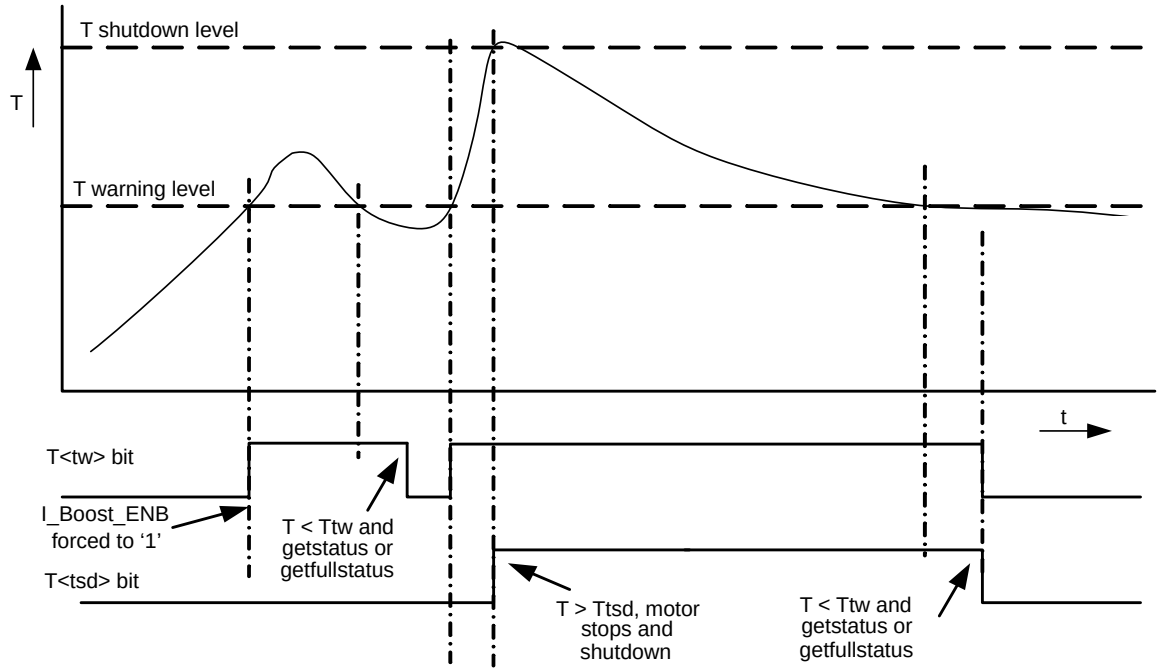



**Figure 14. Illustration of Thermal Management Situation**

## Under−Voltage Condition and Autarkic Functionality

### Battery Voltage Management

The NCV70627 monitors the $V_{BB}$ voltage by means of two under voltage threshold UV3 and UV2 and one shutdown level. The only condition to go back to normal operation is to recover by a $V_{BB}$ voltage higher than UV1. The flags <UV2> and <UV3> can only be cleared by receiving the header of a GetStatus or a GetFullStatus command after the $V_{BB}$ voltage higher than UV1.

The UV3 and UV1 levels are programmable by a LIN command. There are 8 levels available for the UV3 threshold voltage. The UV1 level is ratio metric coupled with UV3. UV2 has only a fixed threshold level. Refer to the DC parameter table for the different under voltage levels.

When the battery voltage drops below UV3, the <UV3> flag will be set and a Soft Stop is performed to stop the motion. If during this decelerated motion the battery voltage does not go under the UV2 level, the NCV70627 will go to state <StoppedUnder UV1>" and the original Target Position (TagPos) is saved while the motor is kept in position by the Hold current*. As soon as the $V_{BB}$ voltage rises above the UV1 level the NCV70627 will continue the motion the (TagPos) and will go to the normal <Stopped> state afterwards.

When during a motion the battery voltage drops below the UV2 level, the NCV70627 will stop immediately by a Hard Stop and directly enters the state <HardUnder> followed by <ShutUnder>. The motor is placed in HiZ and the flags <UV2> and <Steploss> are set (see Figure 15).

**Note*:** In this situation the <Steploss> flag is not set.

### Remarks:

If $V_{BB}$ voltage drops below the UV2 level while the NCV70627 is in the motion "stabilization phase", only the <UV2> flag is set; the <Steploss> flag is not set.

When the NCV70627 is in a stopped states <Stopped> or <StoppedUnder UV1> and the $V_{BB}$ voltage drops below UV2 level, the device will directly go to the state <ShutUnder>, but does not raise the <Steppploss> flag.

At the UV3 comparator output, there is implemented an unsymmetrical debouncer which will filter immediate actions during unwanted spikes at the battery supply. For transitions, when supply voltage $V_{BB}$ drops below UV3 level, a 32 μs debouncer is implemented that is derived from the internal oscillator. For transitions when supply voltage $V_{BB}$ rises above UV1 level, the NVC70627 reacts after 96 μs debounce time typically (OTP bit UV3debT is not set). This time is increased to 256 μs when OTP bit UV3debT is zapped to "1". Zapping can be done via the SetOTPparam command.

### Autarkic Function

From above described states the device can enter the state <ShutUnder>. When in the <ShutUnder> state, the device will perform the Autarkic Function:

- If in this state $V_{BB}$ becomes > UV1 within 15 seconds, the NCV70627 still will resume the motion to the saved (TagPos) and will go to the <Stopped> state afterwards. It accepts updates of the target position by means of the commands SetPosition, SetPositionShort, SetPosParam and GotoSecurePosition, even if the <UV2> flag and <Steploss> flags are NOT cleared.
- If however the $V_{BB}$ voltage remains below UV2 level voltage level for more than 15 seconds, the device will enter <Shutdown> state and the target position is overwritten by Actual Position. This state can be exited only if $V_{BB}$ is > UV1 voltage level and an incoming command GetStatus or GetFullStatus is received.

### Important Notes:

1. In the case of Autarkic positioning, care needs to be taken because accumulated steploss can cause a significant deviation between physical and stored actual position.
2. The SetDualPosition command will only be executed after clearing the <UV2> and <Steploss> flags.
3. RAM reset occurs when Vdd < VddReset (digital Power−On−Reset level).
4. The Autarkic function remains active as long as $V_{DD}$ > VddReset.

## OTP Register

### OTP Memory Structure

The table below shows how the parameters to be stored in the OTP memory are located.

**Table 19. OTP MEMORY STRUCTURE**

| Address | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0x00 | SecPosA | TSD2 | TSD1 | TSD0 | IREF3 | IREF2 | IREF1 | IREF0 |
| 0x01 | LIN_BR | ADM | BG5 | BG4 | BG3 | BG2 | BG1 | BG0 |
| 0x02 | AbsThr3 | AbsThr2 | AbsThr1 | AbsThr0 | PA3 | PA2 | PA1 | PA0 |
| 0x03 | Irun3 | Irun2 | Irun1 | Irun0 | Ihold3 | Ihold2 | Ihold1 | Ihold0 |
| 0x04 | Vmax3 | Vmax2 | Vmax1 | Vmax0 | Vmin3 | Vmin2 | Vmin1 | Vmin0 |
| 0x05 | SecPos10 | SecPos9 | SecPos8 | Shaft | Acc3 | Acc2 | Acc1 | Acc0 |
| 0x06 | SecPos7 | SecPos6 | SecPos5 | SecPos4 | SecPos3 | SecPos2 | Failsafe | SleepEn |
| 0x07 | UV3debT | UV3Thr2 | UV3Thr1 | UV3Thr0 | StepMode1 | StepMode0 | LOCKBT | LOCKBG |
| 0x08 | SecPos10A | SecPos9A | SecPos8A | OSC4 | OSC3 | OSC2 | OSC1 | OSC0 |
| 0x09 | SecPos7A | SecPos6A | SecPos5A | SecPos4A | SecPos3A | SecPos2A | FailsafeA | SleepEnA |

Parameters stored at address 0x00 and 0x01 and bit <LOCKBT> are already programmed in the OTP memory at circuit delivery. They correspond to the calibration of the circuit and are just documented here as an indication.

Each OTP bit is at ‘0’ when not zapped. Zapping a bit will set it to ‘1’. Thus only bits having to be at ‘1’ must be zapped. Zapping of a bit already at ‘1’ is disabled. Each OTP byte will be programmed separately (see command SetOTPparam). Once OTP programming is completed, bit <LOCKBG> can be zapped to disable future zapping, otherwise any OTP bit at ‘0’ could still be zapped by using a SetOTPparam command.

**Table 20. OTP OVERWRITE PROTECTION**

| Lock Bit | Protected Bytes |
|---|---|
| LOCKBT (factory zapped before delivery) | 0x00[6:0], 0x01[5:0], 0x08[4:0] |
| LOCKBG | 0x00 to 0x09 |

The command used to load the application parameters via the LIN bus in the RAM prior to an OTP Memory programming is SetMotorParam. This allows for a functional verification before using a SetOTPparam command to program and zap separately one OTP memory byte. A GetOTPparam command issued after each SetOTPparam command allows verifying the correct byte zapping.

**Note:** Zapped bits will become active only after a power cycle. After programming the LIN bits the power cycle has to be performed first to guarantee further communication with the device at the new address.

### Application Parameters Stored in OTP Memory

Except for the physical address <PA[3:0]> these parameters, although programmed in a non−volatile memory can still be overwritten in RAM by a LIN SetMotorParam writing operation.

**PA[3:0]** In combination with HW[2:0] it forms the physical address AD[6:0] of the stepper−motor. Up to 128 stepper−motors can theoretically be connected to the same LIN bus.

**AbsThr[3:0]** Absolute threshold used for the motion detection

| Index | AbsThr | | | | AbsThr level (V) (*) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Disable |
| 1 | 0 | 0 | 0 | 1 | 0.6 |
| 2 | 0 | 0 | 1 | 0 | 1.3 |
| 3 | 0 | 0 | 1 | 1 | 1.9 |
| 4 | 0 | 1 | 0 | 0 | 2.6 |
| 5 | 0 | 1 | 0 | 1 | 3.2 |
| 6 | 0 | 1 | 1 | 0 | 3.9 |
| 7 | 0 | 1 | 1 | 1 | 4.5 |
| 8 | 1 | 0 | 0 | 0 | 5.1 |
| 9 | 1 | 0 | 0 | 1 | 5.8 |
| A | 1 | 0 | 1 | 0 | 6.4 |
| B | 1 | 0 | 1 | 1 | 7.1 |
| C | 1 | 1 | 0 | 0 | 7.7 |
| D | 1 | 1 | 0 | 1 | 8.3 |
| E | 1 | 1 | 1 | 0 | 9.0 |
| F | 1 | 1 | 1 | 1 | 9.6 |

(*) Not tested in production. Values are approximations.

**UV3Thr [2:0]** Under voltage threshold voltage for UV3 and UV1.

| Index | UV3Thr | | | UV3 Level | UV1 Level |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 5.90 | 6.62 |
| 1 | 0 | 0 | 1 | 6.30 | 7.07 |
| 2 | 0 | 1 | 0 | 6.70 | 7.52 |
| 3 | 0 | 1 | 1 | 7.10 | 7.97 |
| 4 | 1 | 0 | 0 | 7.50 | 8.41 |
| 5 | 1 | 0 | 1 | 7.90 | 8.86 |
| 6 | 1 | 1 | 0 | 8.30 | 9.31 |
| 7 | 1 | 1 | 1 | 8.70 | 9.76 |

**Irun[3:0]** Current amplitude value to be fed to each coil of the stepper−motor. The table below provides the 16 possible values for <IRUN>.

| Index | Irun | | | | Run Current (mA) | Run Boost Current (mA) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 59 | 81 |
| 1 | 0 | 0 | 0 | 1 | 71 | 98 |
| 2 | 0 | 0 | 1 | 0 | 84 | 116 |
| 3 | 0 | 0 | 1 | 1 | 100 | 138 |
| 4 | 0 | 1 | 0 | 0 | 119 | 164 |
| 5 | 0 | 1 | 0 | 1 | 141 | 194 |
| 6 | 0 | 1 | 1 | 0 | 168 | 231 |
| 7 | 0 | 1 | 1 | 1 | 200 | 275 |
| 8 | 1 | 0 | 0 | 0 | 238 | 327 |
| 9 | 1 | 0 | 0 | 1 | 283 | 389 |
| A | 1 | 0 | 1 | 0 | 336 | 462 |
| B | 1 | 0 | 1 | 1 | 400 | 550 |
| C | 1 | 1 | 0 | 0 | 476 | 655 |
| D | 1 | 1 | 0 | 1 | 566 | 778 |
| E | 1 | 1 | 1 | 0 | 673 | 925 |
| F | 1 | 1 | 1 | 1 | 800 | 1100 |

**Ihold[3:0]** Hold current for each coil of the stepper−motor. The table below provides the 16 possible values for <IHOLD>.

| Index | Ihold | | | | Hold Current (mA) | Hold Boost Current (mA) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 59 | 81 |
| 1 | 0 | 0 | 0 | 1 | 71 | 98 |
| 2 | 0 | 0 | 1 | 0 | 84 | 116 |
| 3 | 0 | 0 | 1 | 1 | 100 | 138 |
| 4 | 0 | 1 | 0 | 0 | 119 | 164 |
| 5 | 0 | 1 | 0 | 1 | 141 | 194 |
| 6 | 0 | 1 | 1 | 0 | 168 | 231 |
| 7 | 0 | 1 | 1 | 1 | 200 | 275 |
| 8 | 1 | 0 | 0 | 0 | 238 | 327 |
| 9 | 1 | 0 | 0 | 1 | 283 | 389 |
| A | 1 | 0 | 1 | 0 | 336 | 462 |
| B | 1 | 0 | 1 | 1 | 400 | 550 |
| C | 1 | 1 | 0 | 0 | 476 | 655 |
| D | 1 | 1 | 0 | 1 | 566 | 778 |
| E | 1 | 1 | 1 | 0 | 673 | 925 |
| F | 1 | 1 | 1 | 1 | 0 | 0 |

**Note:** When the motor is stopped, the current is reduced from <IRUN> to <IHOLD>. In the case of 0 mA hold current (1111 in the hold current table), the following sequence is applied:

1. The current is first reduced to 59 mA or 81 mA during I_Boost function (corresponding to 0000 value in the table).
2. The PWM regulator is switched off; the bottom transistors of the bridges are grounded.

**Step Mode** Setting of step modes.

| Step Mode | | Step Mode |
|---|---|---|
| 0 | 0 | 1/2 stepping |
| 0 | 1 | 1/4 stepping |
| 1 | 0 | 1/8 stepping |
| 1 | 1 | 1/16 stepping |

**Shaft** This bit distinguishes between a clock−wise or counter−clock−wise rotation.

**SecPos[10:2]** Secure Position of the stepper−motor. This is the position to which the motor is driven in case of a LIN communication loss or when the LIN error−counter overflows. If <SecPos[10:2]> = “100 0000 00xx”, secure positioning is disabled for the FailSafe function and the SetDualPosition command while it is not disabled for the GotoSecurePosition and even is still executed for the position “100 0000 00xx”.

**Note:** The Secure Position is coded on 11 bits only, providing actually the most significant bits of the position, the non coded least significant bits being set to ‘0’. The Secure Position in OTP has only 9 bits. The two least significant bits are loaded as ‘0’ to RAM when copied from OTP.

**SecPosA** If <SecPosA> = 0 then <SecPos[10:2]>, <Failsafe> and <SleepEn> stored in bytes 0x05 and 0x06 are used during operation
If <SecPosA> = 1 then <SecPos[10:2]>, <Failsafe> and <SleepEn> stored in bytes 0x08 and 0x09 are used during operation

Programming SecPosA with “1” makes the OTP bytes 0x05 and 0x06 obsolete. In this case the OTP bytes at 0x08 and 0x09 will be read at the positions of bytes 0x05 and 0x06 when reading the OTP via the GetOTPparam command.

**Vmax[3:0]** Maximum velocity

| Index | Vmax | | | | Vmax(full step/s) | Group |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 99 | A |
| 1 | 0 | 0 | 0 | 1 | 136 | B |
| 2 | 0 | 0 | 1 | 0 | 167 | |
| 3 | 0 | 0 | 1 | 1 | 197 | |
| 4 | 0 | 1 | 0 | 0 | 213 | |
| 5 | 0 | 1 | 0 | 1 | 228 | |
| 6 | 0 | 1 | 1 | 0 | 243 | |
| 7 | 0 | 1 | 1 | 1 | 273 | C |
| 8 | 1 | 0 | 0 | 0 | 303 | |
| 9 | 1 | 0 | 0 | 1 | 334 | |
| A | 1 | 0 | 1 | 0 | 364 | |
| B | 1 | 0 | 1 | 1 | 395 | |
| C | 1 | 1 | 0 | 0 | 456 | |
| D | 1 | 1 | 0 | 1 | 546 | D |
| E | 1 | 1 | 1 | 0 | 729 | |
| F | 1 | 1 | 1 | 1 | 973 | |

**Vmin[3:0]** Minimum velocity.

| Index | Vmin | | | | Vmax Factor |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1/32 |
| 2 | 0 | 0 | 1 | 0 | 2/32 |
| 3 | 0 | 0 | 1 | 1 | 3/32 |
| 4 | 0 | 1 | 0 | 0 | 4/32 |
| 5 | 0 | 1 | 0 | 1 | 5/32 |
| 6 | 0 | 1 | 1 | 0 | 6/32 |
| 7 | 0 | 1 | 1 | 1 | 7/32 |
| 8 | 1 | 0 | 0 | 0 | 8/32 |
| 9 | 1 | 0 | 0 | 1 | 9/32 |
| A | 1 | 0 | 1 | 0 | 10/32 |
| B | 1 | 0 | 1 | 1 | 11/32 |
| C | 1 | 1 | 0 | 0 | 12/32 |
| D | 1 | 1 | 0 | 1 | 13/32 |
| E | 1 | 1 | 1 | 0 | 14/32 |
| F | 1 | 1 | 1 | 1 | 15/32 |

**Acc[3:0]** Acceleration and deceleration between Vmax and Vmin.

| Index | Acc | | | | Acceleration (Full−step/s$^2$) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 49 (*) |
| 1 | 0 | 0 | 0 | 1 | 218 (*) |
| 2 | 0 | 0 | 1 | 0 | 1004 . |
| 3 | 0 | 0 | 1 | 1 | 3609 . |
| 4 | 0 | 1 | 0 | 0 | 6228 . |
| 5 | 0 | 1 | 0 | 1 | 8848 . |
| 6 | 0 | 1 | 1 | 0 | 11409 . |
| 7 | 0 | 1 | 1 | 1 | 13970 . |
| 8 | 1 | 0 | 0 | 0 | 16531 . |
| 9 | 1 | 0 | 0 | 1 | 19092 (*) |
| A | 1 | 0 | 1 | 0 | 21886 (*) |
| B | 1 | 0 | 1 | 1 | 24447 (*) |
| C | 1 | 1 | 0 | 0 | 27008 (*) |
| D | 1 | 1 | 0 | 1 | 29570 (*) |
| E | 1 | 1 | 1 | 0 | 34925 (*) |
| F | 1 | 1 | 1 | 1 | 40047 (*) |

(*) restriction on speed

**SleepEn** IF <SleepEn> = 1 −> NCV70627 always goes to low−power sleep mode incase of LIN timeout.
IF <SleepEn> = 0, there is no more automatic transition to low−current sleep mode (i.e. stay in stop mode with applied hold current, unless there are failures). Exception to this rule are the states <Standby> and <Shutdown>, in which the device can enter sleep regardless of the state of SleepEn.

**Note:** The `<SleepEn>` function acts for the LIN command “SLEEP” too. When <SleepEn> = 1 and the Sleep command is received the NCV70627 will go into Sleep. In case the `<SleepEn>` = 0 the NCV70627 will go into stop mode.

**FailSafe**
Description: see section LIN Lost Behavior.

**LIN_BR** Setting of LIN Baud rate.

| <LIN_BR> | Baud rate |
|---|---|
| 0 | 19200 Baud |
| 1 | 9600 Baud |

**ADM** <ADM> controls how the OTP bits and hardwired LIN address bits are combined into the LIN node address (see also LIN Address section).

**UV3DepT** Debounce time after passing the UV1 level of the rising battery voltage slope. The debouce time is specified in the AC parameter table.

**Table 21. RAM REGISTERS**

| Register | Mnemonic | Length (bit) | Related Commands | Comment | Reset State |
|---|---|---|---|---|---|
| Actual position | ActPos | 16 | GetActualPos<br>GetFullStatus<br>GotoSecurePos<br>ResetPosition | 16−bit signed | |
| Last programmed Position | Pos/TagPos | 16/11 | GetFullStatus<br>GotoSecurePos<br>ResetPosition<br>SetPosition<br>SetPositionShort<br>SetPosParam | 16−bit signed or<br>11−bit signed for half stepping<br>(see Positioning) | |
| Acceleration shape | AccShape | 1 | GetFullStatus<br>SetMotorParam | '0' ⇒ normal acceleration from Vmin to Vmax<br>'1' ⇒ motion at Vmin without acceleration | '0' |
| Coil peak current | Irun | 4 | GetFullStatus<br>SetMotorParam | Operating current<br>See look−up table Irun | From OTP memory |
| Coil hold current | Ihold | 4 | GetFullStatus<br>SetMotorParam | Standstill current<br>See look−up table Ihold | |
| Minimum Velocity | Vmin | 4 | GetFullStatus<br>SetMotorParam<br>SetPosParam | See Section Minimum Velocity<br>See look−up table Vmin | |
| Maximum Velocity | Vmax | 4 | GetFullStatus<br>SetMotorParam<br>SetPosParam | See Section Maximum Velocity<br>See look−up table Vmax | |
| Shaft | Shaft | 1 | GetFullStatus<br>SetMotorParam | Direction of movement | |
| Acceleration/ deceleration | Acc | 4 | GetFullStatus<br>SetMotorParam<br>SetPosParam | See Section Acceleration<br>See look−up table Acc | |
| Secure Position | SecPos | 11 | GetFullStatus<br>SetMotorParam | Target position when LIN connection fails; 11 MSB's of 16−bit position (LSB's fixed to '0') | |
| Stepping mode | StepMode | 2 | GetFullStatus<br>SetStallParam | See Section Stepping Modes<br>See look−up table StepMode | |
| Stall detection absolute threshold | AbsThr | 4 | GetFullStatus<br>SetStallParam<br>SetPosParam | The B−emf voltage threshold level at which stall is detected. | |
| Under voltage UV3 | UV3Thr | 3 | GetFullStatus<br>SetStallParam | Under voltage UV3 and UV1 level | |
| Sleep Enable | SleepEn | 1 | SetOTPParam | Enables entering sleep mode after LIN lost. See also LIN lost behavior | |
| Fail Safe | FailSafe | 1 | SetOTPParam | Triggers autonomous motion after LIN lost at POR. See also LIN lost behavior | |
| Stall detection delay | FS2StallEn | 3 | GetFullStatus<br>SetStallParam | Delays the stall detection after acceleration | '000' |
| Stall detection sampling | MinSamples | 3 | GetFullStatus<br>SetStallParam | Duration of the zero current step in number of PWM cycles. | '000' |
| PWM Jitter | PWMJEn | 1 | GetFullStatus<br>SetStallParam | '1' means jitter is added | '0' |
| 100% duty cycle Stall Enable | DC100StEn | 1 | GetFullStatus<br>SetStallParam | '1' means stall detection is enabled in case PWM regulator runs at $\delta$ = 100% | '0' |
| PWM frequency | PWMFreq | 1 | GetFullStatus<br>SetMotorParam | '0' means ~ 22.8 KHz,<br>'1' means ~ 45.6 KHz | '0' |

**Table 22. FLAGS TABLE**

| Flag | Mnemonic | Length (bit) | Related Commands | Comment | Reset State |
|---|---|---|---|---|---|
| LIN Timeout Error | TimE | 1 | GetFullStatus | '1' if no data (dominant state) was received for more than T_timeout (~1.3 s) | '0' |
| LIN Data Error | DataE | 1 | GetFullStatus | '1' when one of the three errors occurred: checksum error, stop bit error or length error | '0' |
| LIN Header Error | HeadE | 1 | GetFullStatus | '1' when one of the two errors occurred: parity error or synchronization error | '0' |
| LIN Bit Error | BitE | 1 | GetFullStatus | '1' when received bit value is different from the one being transmitted | '0' |
| Overall LIN Error | LIN_E | 1 | GetActualPos<br>GetFullStatus<br>GetStatus | Or function of the TimE, DataE, HeadE and BitE | '0' |
| Electrical defect | ElDef | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | <OVC1> or <OVC2> or 'open−load on coil X' or 'open−load on coil XY<br>Resets only after Get(Full)Status | '0' |
| External switch status | ESW | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | '0' = open<br>'1' = close | '0' |
| Electrical flag | HS | 1 | Internal use | <UV2> or <ElDef> or <VDDreset> | '0' |
| Motion status | Motion | 3 | GetFullStatus | "000" = Stop, last movement was inner (CCW) motion<br>"100" = Stop, last movement was outer (CW) motion<br>"001" = inner (CCW) motion acceleration<br>"010" = inner (CCW) motion deceleration<br>"011" = inner (CCW) motion max. speed<br>"101" = outer (CW) motion acceleration<br>"110" = outer (CW) motion deceleration<br>"111" = outer (CW) motion max. speed | "000" |
| Over current in coil X | OVC1 | 1 | GetFullStatus | '1' = over current; reset only after GetFullStatus | '0' |
| Over current in coil Y | OVC2 | 1 | GetFullStatus | '1' = over current; reset only after GetFullStatus | '0' |
| Secure position enabled | SecEn | 1 | Internal use | '0' if <SecPos> = "100 0000 0000"<br>'1' otherwise | n.a. |
| Circuit going to Sleep mode | Sleep | 1 | Internal use | '1' = Sleep mode<br>reset by LIN command | '0' |
| Step loss | StepLoss | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | '1' = step loss due to under voltage, over current, open circuit or stall; Resets only after GetFullStatus or GetActualPos | '1' |
| Device ID Code | Device ID | 4 | GetActualPos | Contains the unique device ID | '3' |
| Absolute Stall | AbsStall | 1 | GetFullStatus | '1' = Vbemf < AbsThr | '0' |
| Stall | Stall | 1 | GetFullStatus<br>GetStatus | '1' = Vbemf < AbsThr | '0' |
| Motor stop | Stop | 1 | Internal use | | '0' |
| Temperature info | Tinfo | 2 | GetActualPos<br>GetStatus<br>GetFullStatus | "00" = normal temperature range<br>"01" = low temperature warning<br>"10" = high temperature warning<br>"11" = motor shutdown | "00" |
| Thermal shutdown | TSD | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | '1' = shutdown ($T_j > T_{tsd}$)<br>Resets only after Get(Full)Status<br>and if <Tinfo> = "00" | '0' |
| Thermal warning | TW | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | '1' = over temperature ($T_j > T_{tw}$)<br>Resets only after Get(Full)Status<br>and if <Tinfo> = "00" | '0' |

**Table 22. FLAGS TABLE**

| Flag | Mnemonic | Length (bit) | Related Commands | Comment | Reset State |
|---|---|---|---|---|---|
| Battery decelerated stop voltage | UV3 | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | '0' = $V_{BB}$ > UV3<br>'1' = $V_{BB}$ ≤ UV3<br>Resets only after reception of header of<br>Get(Full)Status and if $V_{BB}$ > UV1 | '0' |
| Battery hard stop voltage | UV2 | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | '0' = $V_{BB}$ > UV2<br>'1' = $V_{BB}$ ≤ UV2<br>Resets only after reception of header of<br>Get(Full)Status and if $V_{BB}$ > UV1 | '0' |
| Overall UV flag | UV | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | Is the OR function of UV2 and UV3<br>Resets only after reception of header of<br>Get(Full)Status and if $V_{BB}$ > UV1 | '0' |
| Digital supply reset | VddReset | 1 | GetActualPos<br>GetStatus<br>GetFullStatus | Set at '1' after power−up of the circuit. If this was due to a supply micro−cut, it warns that the RAM contents may have been lost; can be reset to '0' with a Get-Status or a Get(Full)Status command | '1' |

### Priority Encoder

The table below describes the simplified state management performed by the main control block.

**Table 23. PRIORITY ENCODER** (See table notes on the following page.)

| **State →** | **Standby** | **Stopped** | **GotoPos** | **Dual Position** | **SoftStop** | **HardStop** | **SoftStopped** | **ShutDown** | **Sleep** | **HardUnder** | **ShutUnder** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Command ↓** | | **Motor Stopped, Ihold in Coils** | **Motor Motion Ongoing** | **No Influence on RAM and TagPos** | **Motor Decelerating** | **Motor Forced to Stop** | | **Motor Stopped, H−bridges in Hi−Z** | **No Power (Note 31)** | | |
| GetActualPos | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | | LIN in-frame response | | LIN in-frame response | LIN in-frame response |
| GetOTPparam | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | | LIN in-frame response | | LIN in-frame response | LIN in-frame response |
| GetFullStatus or GetStatus [ attempt to clear <TSD> and <HS> flags] | LIN in-frame response; if (<TSD> or <HS>) = '0' then → **Stopped** | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | LIN in-frame response | | LIN in-frame response; if (<TSD> or <HS>) = '0' then → **Stopped** | | LIN in-frame response | |
| SetMotorParam [Master takes care about proper update] | RAM update | RAM update | RAM update | RAM update | RAM update | RAM update | RAM update | RAM update | | RAM update | RAM update |
| ResetPosition | | <TagPos> and <ActPos> reset | | | | | <TagPos> kept | <TagPos> and <ActPos> reset | | | <TagPos> and <ActPos> reset |
| SetPosition | | <TagPos> updated; → **GotoPos** | <TagPos> updated | <TagPos> updated | | | | | | | |
| SetPositionShort | | <TagPos> updated; → **GotoPos** | <TagPos> updated | <TagPos> updated | | | | | | | |
| GotoSec Position | | If <SecEn> = '1' then <TagPos> = <SecPos>; → **GotoPos** | If <SecEn> = '1' then <TagPos> = <SecPos> | If <SecEn> = '1' then <TagPos> = <SecPos> | | | | | | | |
| DualPosition | | → Dual Position | | | | | | | | | |
| SoftStop | | | → SoftStop | | | | | | | | |
| Sleep or LIN timeout [ ⇒ <Sleep> = '1', reset by any LIN command received later] | → Sleep | (Note 38) | If <SecEn> = '1' then <TagPos> = <SecPos> else → **SoftStop** | If <SecEn> = '1' then <Tag Pos> = <SecPos>; will be evaluated after DualPosition | No action; <Sleep> flag will be evaluated when motor stops | No action; <Sleep> flag will be evaluated when motor stops | No action; <Sleep> flag will be evaluated when motor stops | No action; <Sleep> flag will be evaluated when motor stops | | No action; <Sleep> flag will be evaluated when motor stops | |
| HardStop | | | → Hard Stop | → Hard Stop | → Hard Stop | | | | | | |
| $V_{BB}$ < UV2 and t > 15 seconds | | → Hard Under | → Hard Under | → Hard Stop | → Hard Under | | | | | | |
| $V_{BB}$ < UV2 and t < 15 seconds | | | | | | | | | | | → Stopped |
| <ElDef> = '1' ⇒ <HS> = '1' | | → Shutdown | → HardStop; <Step Loss> = '1' | → Hard Stop; <Step Loss> = '1' | → Hard Stop; <Step Loss> = '1' | | → Shutdown | | | | → Shutdown |

**Table 23. PRIORITY ENCODER** (See table notes on the following page.)

| State → | Standby | Stopped | GotoPos | Dual Position | SoftStop | HardStop | SoftStopped | ShutDown | Sleep | HardUnder | ShutUnder |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Command ↓** | | **Motor Stopped, Ihold in Coils** | **Motor Motion Ongoing** | **No Influence on RAM and TagPos** | **Motor Decelerating** | **Motor Forced to Stop** | | **Motor Stopped, H−bridges in Hi−Z** | **No Power (Note 31)** | | |
| Thermal shutdown [<TSD> = '1'] | | → Shutdown | → SoftStop | → SoftStop | | | → Shutdown | | | | → Shutdown |
| Motion finished | | n.a. | → Stopped | → Stopped | → Stopped; <TagPos> = <Act Pos> | → Stopped; <TagPos> = <ActPos> | | n.a. | n.a. | | |

**With the Following Color Code:**

Command Ignored | Transition to Another State | Master is responsible for proper update (see Note 36)

31. Leaving <Sleep> state is equivalent to power−on−reset.
32. After power−on−reset, the <Standby> state is entered.
33. A DualPosition sequence runs with a separate set of RAM registers. The parameters that are not specified in a DualPosition command are loaded with the values stored in RAM at the moment the DualPosition sequence starts. <AccShape> is forced to '1' during second motion. <AccShape> at '0' will be taken into account after the DualPosition sequence. A GetFullStatus command will return the default parameters for <Vmax> and <Vmin> stored in RAM.
34. The <Sleep> flag is set to '1' when a LIN timeout or a Sleep command occurs. It is reset by the next LIN command (<Sleep> is cancelled if not activated yet).
35. Shutdown state can be left only when <TSD> and <HS> flags are reset.
36. Flags can be reset only after the master could read them via a GetStatus or GetFullStatus command, and provided the physical conditions allow for it (normal temperature, correct battery voltage and no electrical defect).
37. A SetMotorParam command sent while a motion is ongoing (state <GotoPos>) should not attempt to modify <Acc> and <Vmin> values. This can be done during a DualPosition sequence since this motion uses its own parameters, the new parameters will be taken into account at the next SetPosition or SetPositionShort command.
38. Some transitions like <GotoPos> → <Sleep> are actually done via several states: <GotoPos> → <SoftStop> → <Stopped> → <Sleep> (see diagram below).
39. Two transitions are possible from state <Stopped> when <Sleep> = '1':
    1) Transition to state <Sleep> if (<SecEn> = '0') or ((<SecEn> = '1') and (<ActPos> = <SecPos>)) or <Stop> = '1'
    2) Otherwise transition to state <GotoPos>, with <TagPos> = <SecPos>
40. <SecEn> = '1' when register <SecPos> is loaded with a value different from the most negative value (i.e. different from 0x400 = "100 0000 0000").
41. <Stop> flag allows distinguishing whether state <Stopped> was entered after HardStop/SoftStop or not. <Stop> is set to '1' when leaving state <HardStop> or <SoftStop> and is reset during first clock edge occurring in state <Stopped>.
42. Command for dynamic assignment of Ids is decoded in all states except <Sleep> and has no effect on the current state.
43. While in state <Stopped>, if <ActPos> → <TagPos> there is a transition to state <GotoPos>. This transition has the lowest priority, meaning that <Sleep>, <Stop>, <TSD>, etceteras are first evaluated for possible transitions.
44. If <StepLoss> is active, then SetPosition, SetPositionShort and GotoSecurePosition commands **are not** ignored. <StepLoss> can only be cleared by a GetStatus or GetFullStatus command.

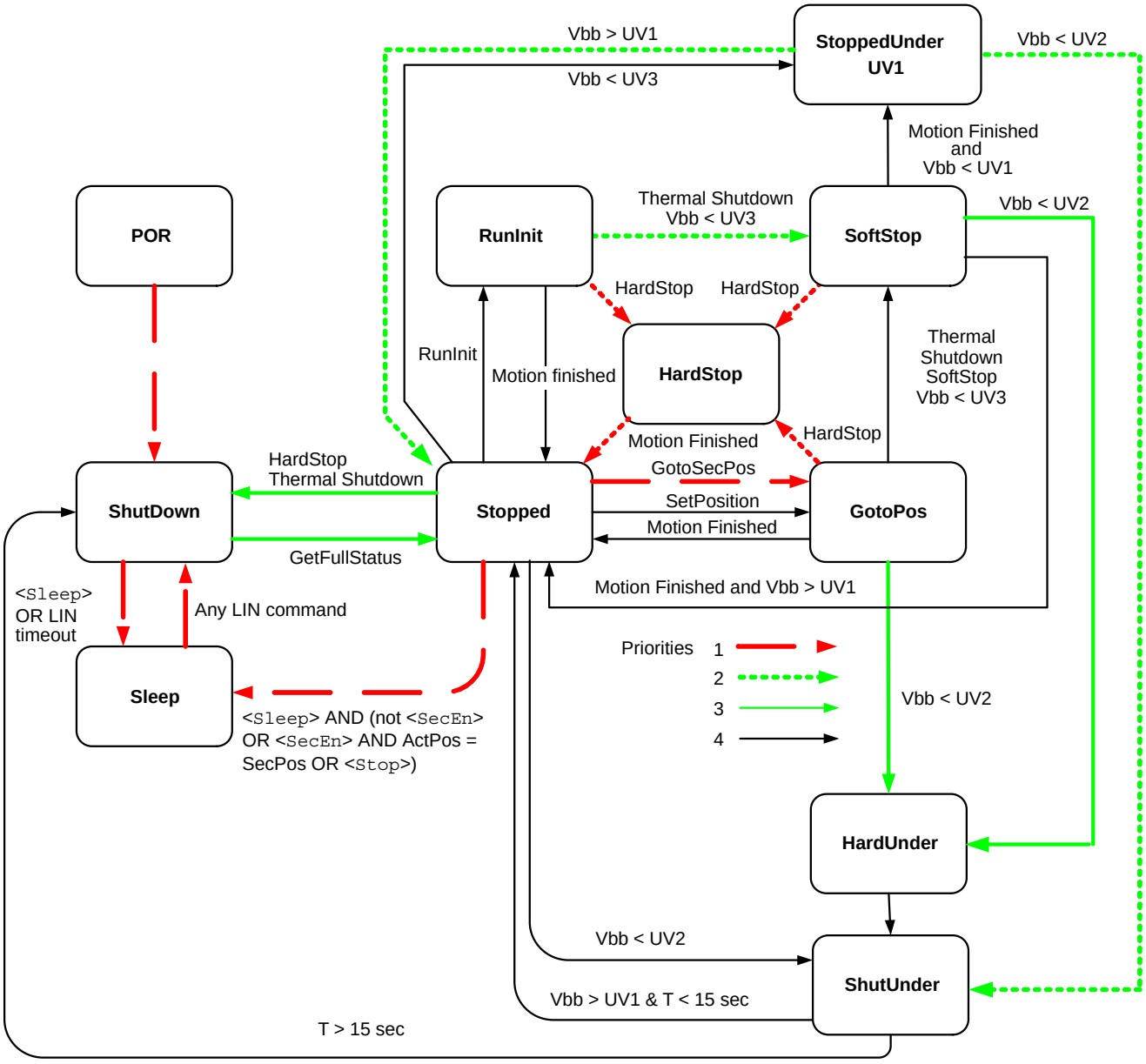



**Figure 15. Simplified State Diagram**

**Remark:** IF <SleepEn> = 0, then the arrow from stopped state to sleep state does not exist.

## Motordriver

### Current Waveforms in the Coils

Figure 16 below illustrates the current fed to the motor coils by the motor driver in half−step mode.

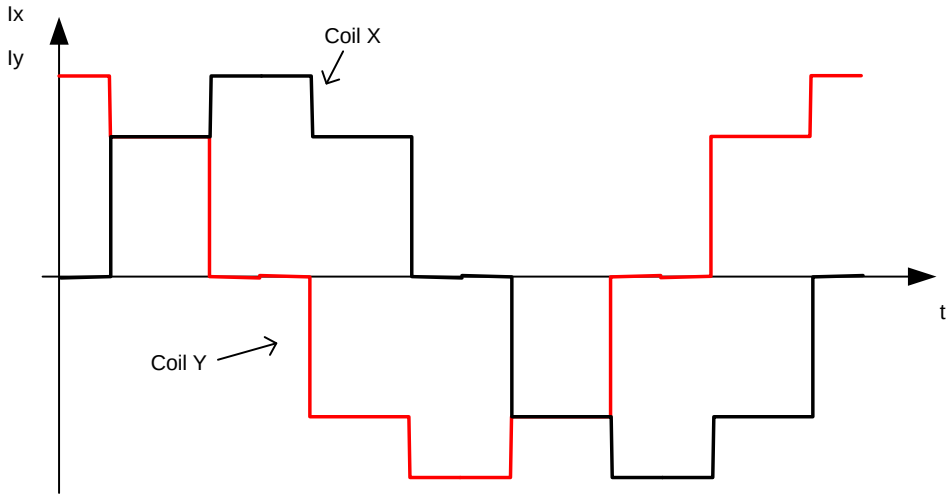



**Figure 16. Current Waveforms in Motor Coils X and Y in Halfstep Mode**

Whereas Figure 17 below shows the current fed to the coils in 1/16$^{th}$ micro stepping (1 electrical period).

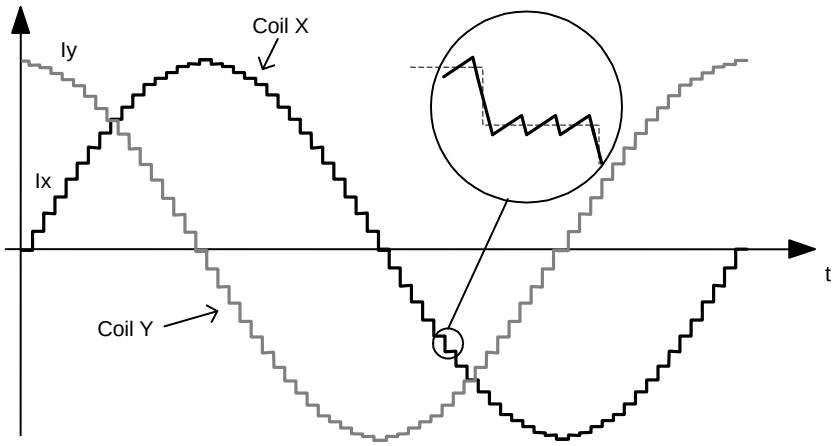



**Figure 17. Current Waveforms in Motor Coils X and Y in 1/16th Micro−Step Mode**

### Motor Current Boost Function

Under certain conditions it can happen that the normal motor currents are not sufficiently high enough to achieve the proper torque for bursting out the motor axis (Especially under cold conditions). For this reason the NCV70627 can be forced to boost mode by setting the <I_BOOST_ENB> bit to ‘0’ via the SetMotorParam command. The boost function increases the current as described in the Irun and Ihold tables. It can only be activated if the junction temperature is lower than $t_{low.}$ When the temperature rises above $t_{tw,,}$ the <I_BOOST_ENB> bit is automatically set back to ‘1’ causing that the current is switched back to the normal current set point values.

### PWM Regulation

In order to force a given current (determined by `<Irun>` or `<Ihold>` and the current position of the rotor) through the motor coil while ensuring high energy transfer efficiency, a regulation based on PWM principle is used. The regulation loop performs a comparison of the sensed output current to an internal reference, and features a digital regulation generating the PWM signal that drives the output switches. The zoom over one micro−step in the Figure 17 above shows how the PWM circuit performs this regulation. To reduce the current ripple, a higher PWM frequency is selectable. The RAM register PWMfreq is used for this.

**Table 24. PWM FREQUENCY SELECTION**

| PWMfreq | Applied PWM Frequency |
|---|---|
| 0 | 22,8 kHz |
| 1 | 45,6 kHz |

### PWM Jitter

To lower the power spectrum for the fundamental and higher harmonics of the PWM frequency, jitter can be added to the PWM clock. The RAM register <PWMJEn> is used for this.

**Table 25. PWM JITTER SELECTION**

| PWMJEn | Status |
|---|---|
| 0 | Single PWM frequency |
| 1 | Added jitter to PWM frequency |

### Motor Starting Phase

At motion start, the currents in the coils are directly switched from **<Ihold>** to **<Irun>** with a new sine/cosine ratio corresponding to the first half (or micro−) step of the motion.

### Motor Stopping Phase

At the end of the deceleration phase, the currents are maintained in the coils at their actual DC level (hence keeping the sine/cosine ratio between coils) during the stabilization time $t_{stab}$ (see AC Table). The currents are then set to the hold values, respectively **Ihold** x sin(TagPos) and **Ihold** x cos(TagPos), as illustrated below. A new positioning order can then be executed. The stabilization time $t_{stab}$ is programmable via a LIN command. There are 4 values possible that can be set dependant the requirement of the motor application.

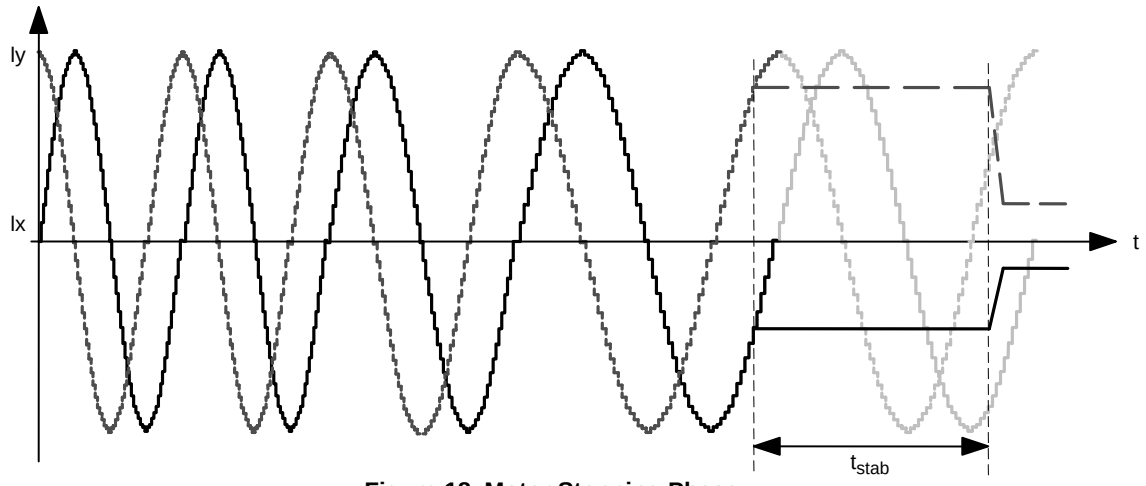



**Figure 18. Motor Stopping Phase**

### Electrical Defect on Coils, Detection and Confirmation

The principle relies on the detection of a voltage drop on at least one transistor of the H−bridge. Then the decision is taken to open the transistors of the defective bridge.

This allows the detection the following short circuits:

- External coil short circuit
- Short between one terminal of the coil and Vbat or Gnd

One cannot detect an internal short in the motor.

Open circuits are detected by 100% PWM duty cycle value during one electrical period with duration, determined by Vmin.

**Table 26. ELECTRICAL DEFECT DETECTION**

| Pins | Fault Mode |
|---|---|
| Yi or Xi | Short−circuit to GND |
| Yi or Xi | Short−circuit to Vbat |
| Yi or Xi | Open |
| Y1 and Y2 | Short circuited |
| X1 and X2 | Short circuited |
| Xi and Yi | Short circuited |

### Motor Shutdown Mode

A motor shutdown occurs when:

- The chip temperature rises above the thermal shutdown threshold Ttsd (see Thermal Shutdown Mode).
- The battery voltage goes below UV2 for longer than 15 seconds (see Under−Voltage Condition and Autarkic Functionality).
- Flag <ElDef> = ‘1’, meaning an electrical problem is detected on one or both coils, e.g. a short circuit.

A motor shutdown leads to the following:

- H−bridges in high impedance mode.
- The <TagPos> register is loaded with the <ActPos>, except in autarkic states.
- The LIN interface remains active, being able to receive orders or send status.

The conditions to get out of a motor shutdown mode are:

- Reception of a GetStatus or GetFullStatus command AND
- The four above causes are no longer detected

This leads to H−bridges going in Ihold mode. Hence, the circuit is ready to execute any positioning command.

This can be illustrated in the following sequence given as an application example. The master can check whether there is a problem or not and decide which application strategy to adopt.

**Table 27. EXAMPLE OF POSSIBLE SEQUENCE USED TO DETECT AND DETERMINE CAUSE OF MOTOR SHUTDOWN**

| Tj ≥Ttsd or $V_{BB}$ ≤ UV2 (>15s) or <ElDef> = '1' ↓ | SetPosition frame ↓ | GetFullStatus or GetStatus frame ↓ | GetFullStatus or GetStatus frame ↓... |
|---|---|---|---|
| − The circuit is driven in motor shutdown mode<br>− The application is not aware of this | − The position set−point is updated by the LIN Master<br>− Motor shutdown mode ⇒ no motion<br>− The application is still unaware | − The application is aware of a problem | − Possible confirmation of the problem |
| | | − Reset <TSD> or <UV2> or <StepLoss> or <ElDef> by the application<br>− Possible new detection of over temperature or low voltage or electrical problem ⇒ Circuit sets <TW> or <TSD> or <UV2> or <StepLoss> or <ElDef> again at '1' | |

**Important:** While in shutdown mode, since there is no hold current in the coils, the mechanical load can cause a step loss, which indeed cannot be flagged by the NCV70627.

If the LIN communication is lost while in shutdown mode, the circuit enters the sleep mode immediately.

**Note:** The Priority Encoder is describing the management of states and commands.

**Warning:** The application should limit the number of consecutive GetStatus or GetFullStatus commands to try to get the NCV70627 out of shutdown mode when this proves to be unsuccessful, e.g. there is a permanent defect. The reliability of the circuit could be altered since Get(Full)Status attempts to disable the protection of the H−bridges.

### Motion Detection

Motion detection is based on the back emf generated internally in the running motor. When the motor is blocked, e.g. when it hits the end−stop, the velocity and as a result also the generated back emf, is disturbed. The NCV70627 senses the back emf and compares the value with an absolute threshold (AbsThr[3:0]). Instructions for correct use of this level in combination with three additional parameters (<MinSamples>, <FS2StallEn> and <DC100StEn>) are available in a dedicated Application Note "Robust Motion Control with AMIS−3062x Stepper Motor Drivers".

When the motor is blocked and the velocity is zero after the acceleration phase, the back emf is low or zero. When this value is below the Absolute threshold, <Stall> is set.

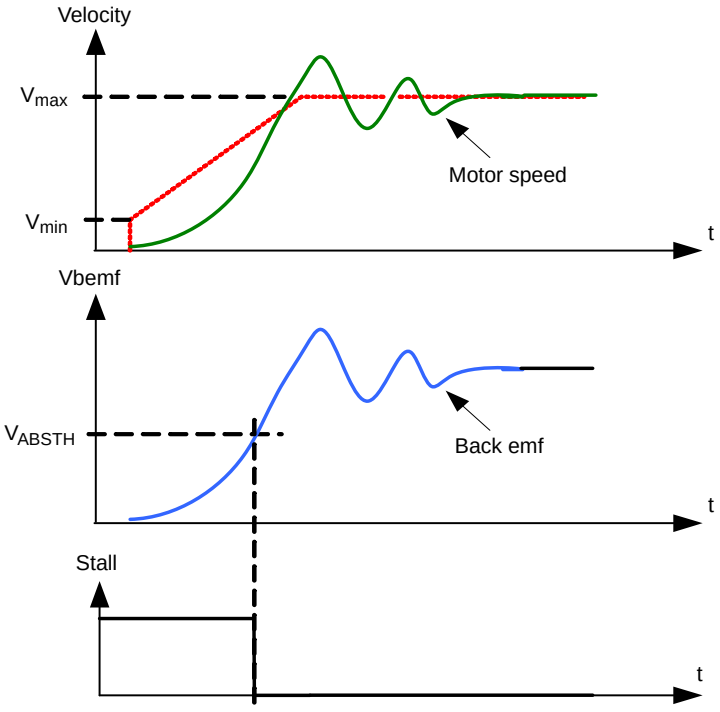



**Figure 19. Triggering of the Stall Flag as Function of the Measured Backemf**

By design, the motion will only be detected when the motor is running at the maximum velocity, not during acceleration or deceleration.

If the motor is positioning when Stall is detected, an (internal) HardStop of the motor is generated and the <StepLoss> and <Stall> flags are set. These flags can only be reset by sending a GetFullStatus command.

If Stall appears during DualPosition then the first phase is cancelled (via internal hardstop) and after timeout Tstab (see AC table) the second phase at Vmin starts.

When the <Stall> flag is set, the position controller will generate an internal HardStop. As a consequence also the <Steploss> flag will be set. The position in the internal counter will be copied to the <ActPos> register. All flags can be read out with the GetStatus or GetFullStatus command.

**Important Remark**
**(limited to motion detection flags / parameters):**

Using GetFullStatus will read **AND** clear the following flags: <Steploss>, <Stall> and <AbsStall>. New positioning is possible and the <ActPos> register will be further updated.

Using GetStatus will read **AND** clear **ONLY** the <Steploss> flag. The <Stall> and <AbsStall> flags are **NOT** cleared. New positioning is possible and the <ActPos> register will be further updated.

Motion detection is disabled when the RAM registers <AbsThr[3:0]> is zero. The level can be programmed using the LIN command SetStallParam in the register <AbsThr[3:0]>. Also the OTP register <AbsThr[3:0]> can be set using the LIN command SetOTPParam. These values are copied in the RAM registers during power on reset.

**Table 28. ABSOLUTE THRESHOLD SETTINGS**

| AbsThr Index | AbsThr Level (V) (*) |
|---|---|
| 0 | Disabled |
| 1 | 0.64 |
| 2 | 1.28 |
| 3 | 1.92 |
| 4 | 2.56 |
| 5 | 3.19 |
| 6 | 3.83 |
| 7 | 4.47 |
| 8 | 5.11 |
| 9 | 5.75 |
| A | 6.38 |
| B | 7.03 |
| C | 7.67 |
| D | 8.30 |
| E | 8.94 |
| F | 9.58 |

NOTE: (*) Not tested in production. Values are typical levels with spread of 0,48V.

**MinSamples**

<MinSamples[2:0]> is a programmable delay timer. After the zero crossing is detected, the delay counter is started. After the delay time−out ($t_{delay}$) the back−emf sample is taken. For more information please refer to the Application Note “Robust Motion Control with AMIS−3062x Stepper Motor Drivers”.

**Table 29. BACK EMF SAMPLE DELAY TIME**

| Index | MinSamples[2:0] | $t_{DELAY}$ (µs) |
|---|---|---|
| 0 | 000 | 87 |
| 1 | 001 | 130 |
| 2 | 010 | 174 |
| 3 | 011 | 217 |
| 4 | 100 | 304 |
| 5 | 101 | 391 |
| 6 | 110 | 521 |
| 7 | 111 | 694 |

**FS2StallEn**

If <AbsThr> <> 0 (i.e. motion detection is enabled), then stall detection will be activated AFTER the acceleration ramp + an additional number of full−steps, according to the following table:

**Table 30.**
**ACTIVATION DELAY OF MOTION DETECTION**

| Index | FS2StallEn[2:0] | Delay (Full Steps) |
|---|---|---|
| 0 | 000 | 0 |
| 1 | 001 | 1 |
| 2 | 010 | 2 |
| 3 | 011 | 3 |
| 4 | 100 | 4 |
| 5 | 101 | 5 |
| 6 | 110 | 6 |
| 7 | 111 | 7 |

**DC100StEn**

When a motor with large bemf is operated at high speed and low supply voltage, then the PWM duty cycle can be as high as 100%. This indicates that the supply is too low to generate the required torque and might also result in erroneously triggering the stall detection. The bit <DC100StEn> enables stall detection when duty cycle is 100%. For more information please refer to the Application Note “Robust Motion Control with AMIS−3062x Stepper Motor Drivers”.

## Lin Controller

### General Description

The LIN (local interconnect network) is a serial communications protocol that efficiently supports the control of mechatronics nodes in distributed automotive applications. The physical interface implemented in the NCV70627 is compliant to the LIN rev. 2.0 & 2.1 specifications. It features a slave node, thus allowing for:

- single−master / multiple−slave communication
- self synchronization without quartz or ceramics resonator in the slave nodes
- guaranteed latency times for signal transmission
- single−signal−wire communication
- transmission speed selectable between 9.6 and 19.2 kbit/s
- selectable length of Message Frame: 2, 4, and 8 bytes
- configuration flexibility
- data checksum (classic checksum, cf. LIN1.3) security and error detection
- detection of defective nodes in the network

It includes the analog physical layer and the digital protocol handler.

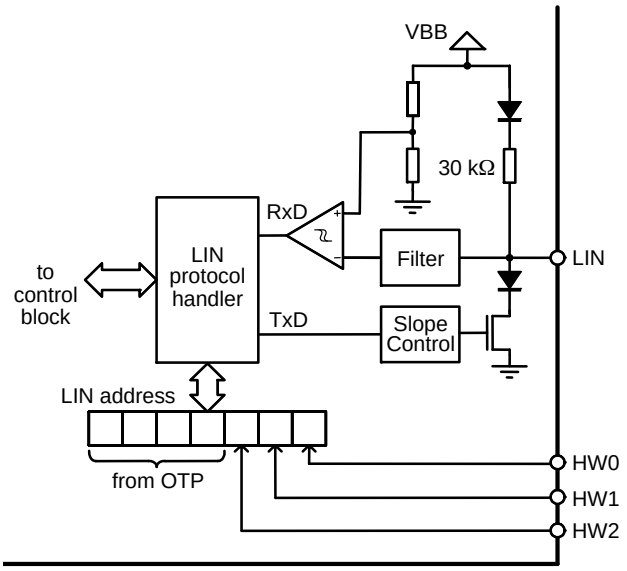



**Figure 20. LIN Interface**

The analog circuitry implements a low side driver with a pull−up resistor as a transmitter, and a resistive divider with a comparator as a receiver. The specification of the line driver/receiver follows the ISO 9141 standard with some enhancements regarding the EMI behavior.

### Slave Operational Range for Proper Self Synchronization

The LIN interface will synchronize properly in the following conditions:

- Vbat ≥ 8 V
- Ground shift between master node and slave node < ±1 V

It is highly recommended to use the same type of reverse battery voltage protection diode for the Master and the Slave nodes.

## Functional Description

### Analog Part

The transmitter is a low−side driver with a pull−up resistor and slope control. The receiver mainly consists of a comparator with a threshold equal to $V_{BB}/2$. Figure 4 shows the characteristics of the transmitted and received signal. See AC Parameters for timing values.

### Protocol Handler

This block implements:

- Bit synchronization
- Bit timing
- The MAC layer
- The LLC layer
- The supervisor

### Error Status Register

The LIN interface implements a register containing an error status of the LIN communication. This register is as follows:

**Table 31. LIN ERROR REGISTER**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Not used | Not used | Not used | Not used | **Time** out error | **Data** error Flag | **Header** error Flag | **Bit** error Flag |

With:

Data error flag: (= Checksum error + StopBit error + Length error)

Header error flag: (= Parity error + SynchField error)

Time out flag: The message frame is not fully completed within the maximum length

Bit error flag: Difference in bit sent and bit monitored on the LIN bus

A `GetFullStatus` frame will reset the error status register.

### Physical Address of the Circuit

The circuit must be provided with a physical address in order to discriminate this circuit from other ones on the LIN bus. This address is coded on 7 bits, yielding the theoretical possibility of 128 different circuits on the same bus. It is a combination of 4 OTP memory bits and of the 3 hardwired address bits (pins HW[2:0]). However the maximum number of nodes in a LIN network is also limited by the physical properties of the bus line. It is recommended to limit the number of nodes in a LIN network to not exceed 16. Otherwise the reduced network impedance may prohibit a fault free communication under worst case conditions. Every additional node lowers the network impedance by approximately 3%.

The node address is a combination of 4 OTP memory bits and 3 hardwired address bits (pins HW[2:0]). Depending on the Addressing Mode (ADM–bit in OTP) the bits of the address are combined as illustrated below.

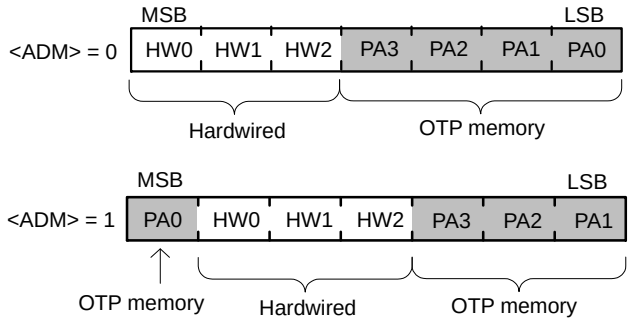



**Figure 21. Combination of OTP and Hardwired Address Bits in Function of ADM (Address Mode)**

NOTE: Pins HW0 and HW1 are 3.3 V digital inputs, whereas pin HW2 is compliant with a 12 V level, e.g. it can be connected to Vbat or Gnd via a terminal of the PCB. For SetPositionShort it is recommended to set HW0, HW1 and HW2 to '1'.

### LIN Frames

The LIN frames can be divided in writing and reading frames. A frame is composed of an 8−bit Identifier followed by 2, 4 or 8 data−bytes and a checksum byte.

**Note:** The checksum is conform LIN1.3, classic checksum calculation over only data bytes. (Checksum is an inverted 8−bit sum with carry over all data bytes.)

Writing frames will be used to:

- Program the OTP Memory;
- Configure the component with the stepper−motor parameters (current, speed, stepping−mode, etc.);
- Provide set−point position for the stepper−motor;
- Control the motion state machine.

Whereas reading frames will be used to:

- Get the actual position of the stepper−motor;
- Get status information such as error flags;
- Verify the right programming and configuration of the component.

### Writing Frames

The LIN master sends commands and/or information to the slave nodes by means of a writing frame. According to the LIN specification, identifiers are to be used to determine a specific action. If a physical addressing is needed, then some bits of the data field can be dedicated to this, as illustrated in the example below.

| Identifier Byte | | | | | | | | Data Byte 1 | | | | | | | | Data Byte 2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID0 | ID1 | ID2 | ID3 | ID4 | ID5 | ID6 | ID7 | | | | | | | | | | | | | | | | |
| | | | | | | | | phys. address | | | | command parameters (e.g. position) | | | | | | | | | | | |

<ID6> and <ID7> are used for parity check over <ID0> to <ID5>, conform LIN1.3 specification. <ID6> = <ID0> ⊗ <ID1> ⊗ <ID2> ⊗ <ID4> (even parity) and <ID7> = NOT(<ID1> ⊗ <ID3> ⊗ <ID4> ⊗ <ID5>) (odd parity).

Another possibility is to determine the specific action within the data field in order to use less identifiers. One can for example use the reserved identifier 0x3C and take advantage of the 8 byte data field to provide a physical address, a command and the needed parameters for the action, as illustrated in the example below.

| ID | | Data Byte 1 | | Data Byte 2 | Data Byte 3 | Data Byte 4 | Data Byte 5 | Data Byte 6 | Data Byte 7 | Data Byte 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x3C | 00 | | 1 | | | | | | | |
| | | AppCmd | | command | physical address | parameters | | | | |

NOTE: Bit 7 of Data byte 1 must be at '1' since the LIN specification requires that contents from 0x00 to 0x7F must be reserved for broadcast messages (0x00 being for the "Sleep" message). See also LIN command Sleep

The writing frames used with the NCV70627 are the following:

**Type #1:** General purpose 2 or 4 data bytes writing frame with a dynamically assigned identifier. This type is dedicated to short writing actions when the bus load can be an issue. They are used to provide direct command to one (<Broad> = '1') or all the slave nodes (<Broad> = '0'). If <Broad> = '1', the physical address of the slave node is provided by the 7 remaining bits of DATA2. DATA1 will contain the command code (see Dynamic assignment of Identifiers), while, if present, DATA3 to DATA4 will contain the command parameters, as shown below.

| ID | | | | | | | | Data1 | Data2 | | Data3... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ID0 | ID1 | ID2 | ID3 | ID4 | ID5 | ID6 | ID7 | command | Physical address | Broad | Parameters... |

NOTE: <ID4> and <ID5> indicate the number of data bytes.

| ID5 | ID4 | Ndata (number of data fields) |
|---|---|---|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 1 | 0 | 4 |
| 1 | 1 | 8 |

**Type #2:** two, four or eight data bytes writing frame with an identifier dynamically assigned to an application command, regardless of the physical address of the circuit.

**Type #3:** two data bytes writing frame with an identifier dynamically assigned to a particular slave node together with an application command. This type of frame requires that there are as many dynamically assigned identifiers as there are NCV70627 circuits using this command connected to the LIN bus.

**Type #4:** eight data bytes writing frame with 0x3C identifier.

### Reading Frames

A reading frame uses an in−frame response mechanism. That is: the master initiates the frame (synchronization field + identifier field), and one slave sends back the data field together with the check field. Hence, two types of identifiers can be used for a reading frame:

- Direct ID, which points at a particular slave node, indicating at the same time which kind of information is awaited from this slave node, thus triggering a specific command. This ID provides the fastest access to a read command but is forbidden for any other action.
- Indirect ID, which only specifies a reading command, the physical address of the slave node that must answer having been passed in a previous writing frame, called a preparing frame. Indirect ID gives more flexibility than a direct one, but provides a slower access to a read command.
  1. A reading frame with indirect ID must always be consecutive to a preparing frame. It will otherwise not be taken into account.
  2. A reading frame will always return the physical address of the answering slave node in order to ensure robustness in the communication.

The reading frames, used with the NCV70627, are the following:

**Type #5:** two, four or eight Data bytes reading frame with a direct identifier dynamically assigned to a particular slave node together with an application command. A preparing frame is not needed.

**Type #6:** eight Data bytes reading frame with 0x3D identifier. This is intrinsically an indirect type, needing therefore a preparation frame. It has the advantage to use a reserved identifier. (Note: because of the parity calculation done by the master, the identifier becomes 0x7D as physical data over the bus).

### Preparing Frames

A preparing frame is a frame from the master that warns a particular slave node that it will have to answer in the next frame (being a reading frame). A preparing frame is needed when a reading frame does not use a dynamically assigned direct ID. Preparing and reading frames must be consecutive. A preparing frame will contain the physical address of the LIN slave node that must answer in the reading frame and will also contain a command indicating which kind of information is awaited from the slave.

The preparing frames used with the NCV70627 can be of type #7 or type #8 described below.

**Type #7:** two data bytes writing frame with dynamically assigned identifier. The identifier of the preparing frame has to be assigned to ROM pointer 1000, see Table 35.

**Table 32. PREPARING FRAME #7**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | 1 | CMD[6:0] | | | | | | |
| 2 | Data 2 | 1 | AD[6:0] | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

Where:

(*) According to parity computation

**Type #8:** eight data bytes preparing frame with 0x3C identifier.

**Table 33. PREPARING FRAME #8**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD = ... | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] | | | | | | |
| 3 | Data 3 | 1 | AD[6:0] | | | | | | |
| 4 | Data 4 | Data4[7:0] FF | | | | | | | |
| 5 | Data 5 | Data5[7:0] FF | | | | | | | |
| 6 | Data 6 | Data6[7:0] FF | | | | | | | |
| 7 | Data 7 | Data7[7:0] FF | | | | | | | |
| 8 | Data 8 | Data8[7:0] FF | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

AppCMD: If = ‘0x80’ this indicates that Data 2 contains an application command

CMD[6:0]: Application Command “byte”

AD[6:0]: Slave node physical address

Data[7:0]: Data transmitted

### Dynamic Assignment of Identifiers

The identifier field in the LIN datagram denotes the content of the message. Six identifier bits and two parity bits are used to represent the content. The identifiers 0x3C and 0x3F are reserved for command frames and extended frames. Slave nodes need to be very flexible to adapt itself to a given LIN network in order to avoid conflicts with slave nodes from different manufacturers. Dynamic assignment of the identifiers will fulfill this requirement by writing identifiers into the circuits RAM. ROM pointers are linking commands and dynamic identifiers together. A writing frame with identifier 0x3C issued by the LIN master will write dynamic identifiers into the RAM. One writing frame is able to assign 4 identifiers; therefore 3 frames are needed to assign all identifiers. Each ROM pointer <ROMp_x [3:0]> place the corresponding dynamic identifier <Dyn_ID_x [5:0]> at the correct place in the RAM (see Table below: LIN – Dynamic Identifiers Writing Frame).

When setting <Broad> to zero broadcasting is active and each slave on the LIN bus will store the same dynamic identifiers, otherwise only the slave with the corresponding slave address is programmed.

**Table 34. DYNAMIC IDENTIFIERS WRITING FRAME**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0x3C | | | | | | | |
| 1 | AppCMD | 0x80 | | | | | | | |
| 2 | CMD | 1 | 0x11 | | | | | | |
| 3 | Address | Broad | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| 4 | Data | DynID_1[3:0] | | | | ROMp_1[3:0] | | | |
| 5 | Data | DynID_2[1:0] | | ROMp_2[3:0] | | | | DynID_1[5:4] | |
| 6 | Data | ROMp_3[3:0] | | | | DynID_2[5:2] | | | |
| 7 | Data | ROMp_4[1:0] | | DynID_3[5:0] | | | | | |
| 8 | Data | DynID_4[5:0] | | | | | | ROMp_4[3:2] | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

CMD[6:0]: 0x11, corresponding to dynamic assignment of four LIN identifiers

Broad: If <Broad> = ‘0’ all the circuits connected to the LIN bus will share the same dynamically assigned identifiers.

Dyn_ID_x [5:0]: Dynamically assigned LIN identifier to the application command which ROM pointer is <ROMp_x [3:0]>
One frame allows only assigning of four identifiers. Therefore, additional frames could be needed in order to assign more identifiers (maximum three for the NCV70627).

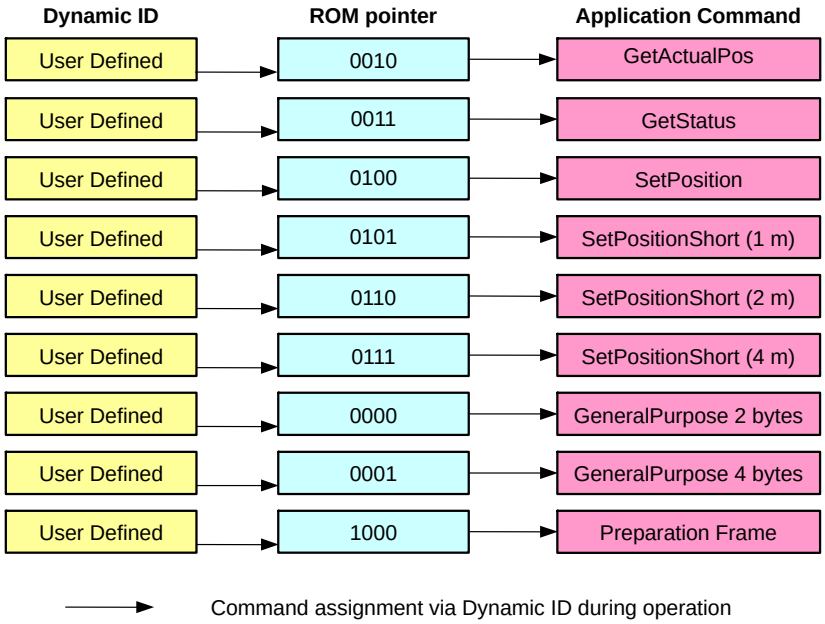



**Figure 22. Principle of Dynamic Command Assignment**

### Commands Table

**Table 35. LIN COMMANDS WITH CORRESPONDING ROM POINTER**

| Command Mnemonic | Command Byte (CMD) | | Dynamic ID (example) | ROM Pointer |
|---|---|---|---|---|
| GetActualPos | 000000 | 0x00 | 100xxx | 0010 |
| GetFullStatus | 000001 | 0x01 | n.a. | |
| GetOTPparam | 000010 | 0x02 | n.a. | |
| GetStatus | 000011 | 0x03 | 000xxx | 0011 |
| GotoSecurePosition | 000100 | 0x04 | n.a. | |
| HardStop | 000101 | 0x05 | n.a. | |
| ResetPosition | 000110 | 0x06 | n.a. | |
| SetDualPosition | 001000 | 0x08 | n.a. | |
| SetMotorParam | 001001 | 0x09 | n.a. | |
| SetOTPparam | 010000 | 0x10 | n.a. | |
| SetStallParam | 010110 | 0x16 | n.a. | |
| SetPosition (16−bit) | 001011 | 0x0B | 10xxxx | 0100 |
| SetPositionShort (1 motor) | 001100 | 0x0C | 001001 | 0101 |
| SetPositionShort (2 motors) | 001101 | 0x0D | 101001 | 0110 |
| SetPositionShort (4 motors) | 001110 | 0x0E | 111001 | 0111 |
| SetPosParam | 101111 | 0x2F | 110xxx | 1001 |
| Sleep | n.a. | | n.a. | |
| SoftStop | 001111 | 0x0F | n.a. | |
| Dynamic ID assignment | 010001 | 0x11 | n.a. | |
| General purpose 2 Data bytes | | | 011000 | 0000 |
| General purpose 4 Data bytes | | | 101000 | 0001 |
| Preparing frame | | | 011010 | 1000 |

NOTE: “xxx” allows addressing physically a slave node. Therefore, these dynamic identifiers cannot be used for more than eight stepper motors. Only ten ROM pointers are needed for the NCV70627.

## LIN Lost Behavior

### Introduction

When the LIN communication is broken for a duration of 25000 consecutive frames (= 1,30s @ 19200 kbit/s) NCV70627 sets an internal flag called “LIN lost”. The functional behavior depends on the state of OTP bits <SleepEn> and <FailSafe>, and if this loss in LIN communication occurred at (or before) power on reset or in normal powered operation.

### Sleep Enable

The OTP bit <SleepEn> enables or disables the entering in low−power sleep mode in case of LIN time−out. Default the entering of the sleep−mode is disabled.

**Table 36. SLEEP ENABLE SELECTION**

| <SleepEn> | Behavior |
|---|---|
| 0 | Entering low−power sleep mode is disabled except from <Standby> and <Shutdown> |
| 1 | Entering low−power sleep mode enabled |

### Fail Safe Motion

The OTP bit <FailSafe> enables or disables an automatic motion to a predefined secure position. See also Autonomous Motion.

**Table 37. FAIL SAFE ENABLE SELECTION**

| <FailSafe> | Behavior |
|---|---|
| 0 | NO reference motion in case of LIN – lost |
| 1 | ENABLES reference motion to a secure position in case of LIN–lost (if the device has not been yet referenced with SetDualPosition) |

NCV70627 is able to perform an Autonomous Motion to a preferred position. This positioning starts after the detection of lost LIN communication and depends on:

− the OTP bit <FailSafe> = 1.

− RAM register <SecPos[10:0]> ≠ 0x400

The functional behavior depends if LIN communication is lost during normal operation (see figure below case A) or at (or before) startup (case B):

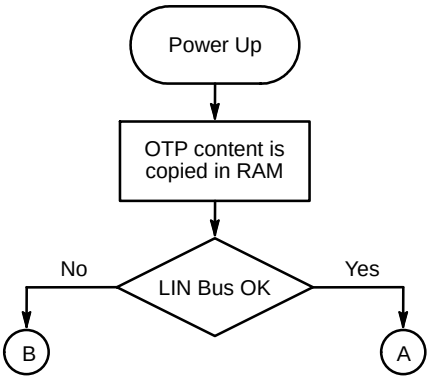



**Figure 23. Flow Chart Power−Up of NCV70627 (Case A: LIN lost during operation and Case B: LIN lost at startup)**

### LIN Lost During Normal Operation

If the LIN communication is lost during normal operation, it is assumed that NCV70627 is referenced (by Dual postioning or Resetposition). In other words the <ActPos> register contains the “real” actual position. At LIN – lost an absolute positioning to the stored secure position SecPos is done. This is further called Secure Positioning.

If OTP bit <FailSafe> = 1, the reaction is the following:

If the device has already been referenced, it is assumed that <ActPos> register contains the “real” actual position. At LIN – lost an absolute positioning to the stored secure position SecPos is done (identical to the case, when OTP bit <FailSafe> = 0).

If the device was not referenced yet, the <ActPos> register does not contain a valid position. At LIN – lost a referencing is started using DualPositioning. A first negative motion of half the positioner range is initiated until the stall position is reached. The motion parameters stored in OTP will be used for this. After this mechanical end−position is reached, <ActPos> will be reset to zero. A second motion of 10 Fullsteps is executed to assure that the motion is really at the end position. After the second motion, a third motion is executed to the Secure Position also stored in OTP; if <SecPos> = 0x400, this second motion is not executed.

Following sequence will be followed. See Figure 22.

1. <SecPos[10:0]> from RAM register will be used. This can be different from OTP register if earlier LIN master communication has updated this. See also Secure Position and command SetMotorParam.

I. If <SecPos[10:0]> = 0x400: No Secure Positioning will be performed

II. If <SecPos[10:0]> ≠ 0x400: Perform a Secure Positioning. This is an absolute positioning (slave knows its ActPos. <SecPos[10:0]> will be copied in <TagPos>)

Depending on <Sleep> NCV70627 will enter the <Stopped> state or the <Sleep> state. See Table 36.

### Important Remarks:

1. The Secure Position has a resolution of 11 bit (2Fs resolution on positions).
2. Same behavior in case of HW2 float (= lost LIN address), except for entering Sleep mode. If HW2 is floating, but there is LIN communication, Sleep mode is not entered. See also Hardwired Address HW2

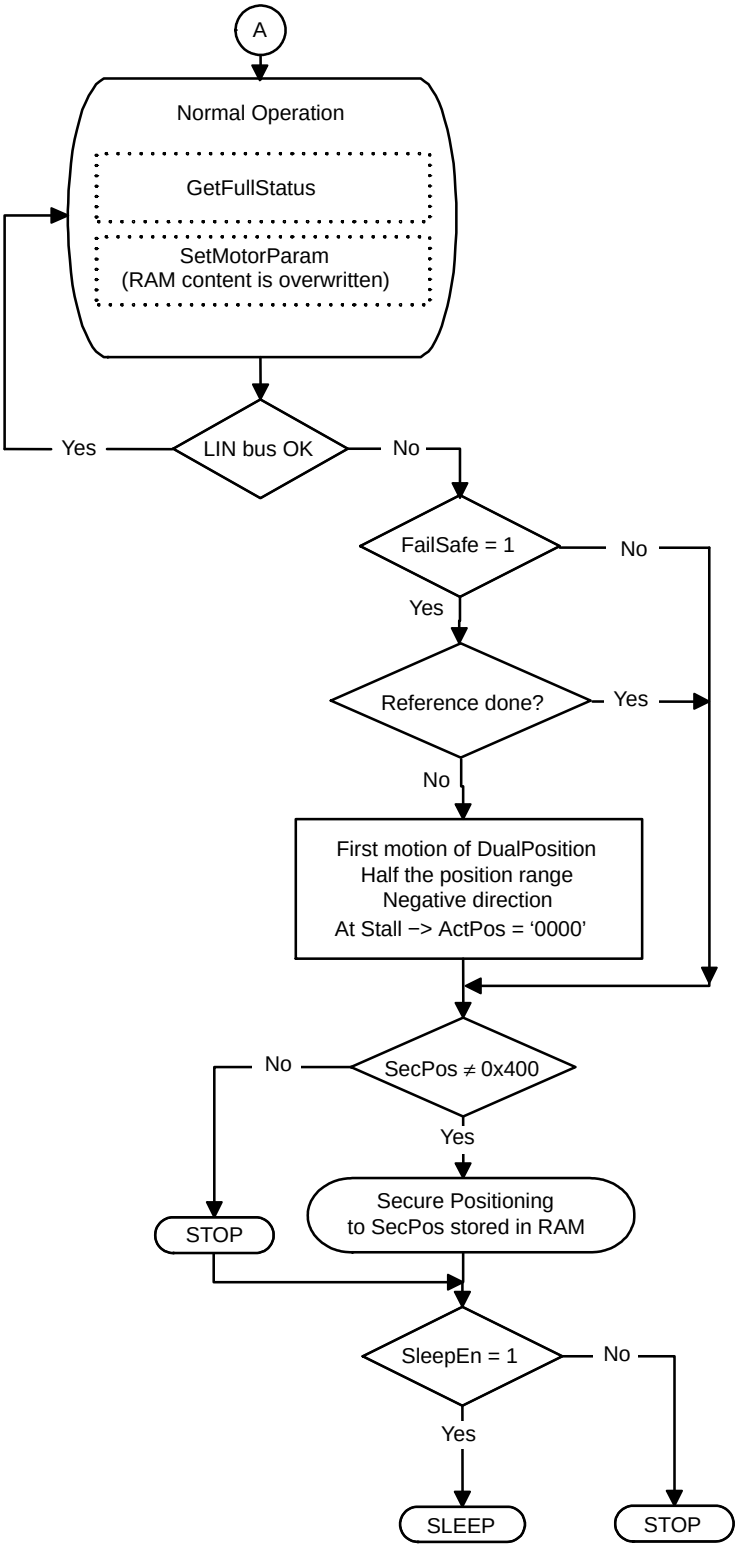



**Figure 24. Case A: LIN Lost During Normal Operation**

### LIN Lost Before or At Power On

If the LIN communication is lost before or at power on, the <ActPos> register does not reflect the “real” actual position. So at LIN − lost a referencing is started using DualPositioning. A first negative motion for half the positioner range is initiated until the stall position is reached. The motion parameters stored in OTP will be used for this. After this mechanical end position is reached, <ActPos> will be reset to zero. A second motion will start to the Secure Position also stored in OTP. More details are given below.

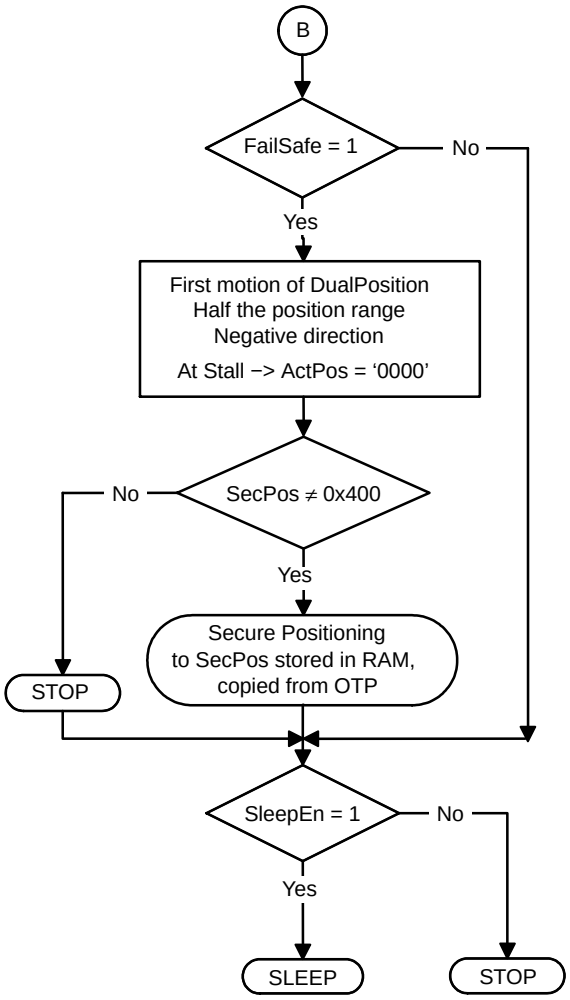



**Figure 25. Case B: LIN Lost at or During Sart−Up**

If LIN is lost before or at power on, following sequence will be followed. See Figure 23.

1. If the LIN communication is lost AND <FailSafe> = 0, secure positioning will be done at absolute position (stored secure position <SecPos>.) Depending on SleepEn NCV70627 will enter the <Stop> state or <Sleep> state. See Table 36.
2. If the LIN communication is lost AND <FailSafe> = 1 a referencing is started using DualPositioning, meaning a negative motion for half the positioner range is initiated until the stall position is reached. The motion parameters stored in OTP will be used for this. After this mechanical end position is reached <ActPos> will be reset to zero. The direction of the motion is given by the Shaft bit.

−If <SecPos[10:0]> = 0x400:
No Second Motion will be performed.

## LIN Application Commands

### Introduction

The LIN Master will have to use commands to manage the different application tasks the NCV70627 can feature. The commands summary is given in Table 38 below.

**Table 38. COMMANDS SUMMARY**

| Command | | Frames | | | Description |
|---|---|---|---|---|---|
| Mnemonic | Code | Prep # | Read # | Write # | |
| **READING COMMAND** | | | | | |
| GetActualPos | 0x00 | 7, 8 | 5, 6 | | Returns the actual position of the motor |
| GetFullStatus | 0x01 | 7, 8 | 6 | | Returns a complete status of the circuit |
| GetOTPparam | 0x02 | 7, 8 | 6 | | Returns the OTP memory content |
| GetStatus | 0x03 | | 5 | | Returns a short status of the circuit |
| **WRITING COMMANDS** | | | | | |
| GotoSecurePosition | 0x04 | | | 1 | Drives the motor to its secure position |
| HardStop | 0x05 | | | 1 | Immediate motor stop |
| ResetPosition | 0x06 | | | 1 | Actual position becomes the zero position |
| SetDualPosition | 0x08 | | | 4 | Drives the motor to 2 different positions with different speeds |
| SetMotorParam | 0x09 | | | 4 | Programs the motion parameters and values for the current in the motor's coils |
| SetOTPparam | 0x10 | | | 4 | Programs (and zaps) a selected byte of the OTP memory |
| SetStallparam | 0x16 | | | 4 | Programs the motion detection parameters |
| SetPosition | 0x0B | | | 1, 3, 4 | Drives the motor to a given position |
| SetPositionShort (1 m.) | 0x0C | | | 2 | Drives the motor to a given position (11 bits 1/2step resolution) |
| SetPositionShort (2 m.) | 0x0D | | | 2 | Drives two motors to 2 given positions (11 bits 1/2step resolution) |
| SetPositionShort (4 m.) | 0x0E | | | 2 | Drives four motors to 4 given positions (11 bits 1/2step resolution) |
| SetPosParam | 0x2F | | | 2 | Drives the motor to a given position and programs some of the motion parameters. |
| **SERVICE COMMANDS** | | | | | |
| Sleep | | | | 1 | Drives circuit into sleep mode if <SleepEn> = 1<br>Drives circuit into stopped mode if if <SleepEn> = 0 |
| SoftStop | 0x0F | | | 1 | Motor stopping with a deceleration phase |

These commands are described hereafter, with their corresponding LIN frames. Refer to LIN Frames for more details on LIN frames, particularly for what concerns dynamic assignment of identifiers. A color coding is used to distinguish between master and slave parts within the frames and to highlight dynamic identifiers. An example is shown below.

**Table 39. COLOR CODE USED IN THE DEFINITION OF LIN FRAMES**

| GetStatus Reading Frame | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Byte | Content | Structure | | | | | | | |
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | ESW | AD[6:0] | | | | | | |
| 2 | Data 2 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |

The Identifier is always sent by the LIN master.

Convention:

The Identifier and Data sent by the master are in gray presented.

The Data sent by the slave is in white presented.

Usually, the NCV70627 makes use of dynamic identifiers for general−purpose 2, 4 or 8 bytes writing frames. If dynamic identifiers are used for other purposes, this is acknowledged.

Some frames implement a <Broad> bit that allows addressing a command to all the NCV70627 circuits connected to the same LIN bus. <Broad> is active when at '0', in which case the physical address provided in the frame is thus not taken into account by the slave nodes.

## Application Commands

### GetActualPos

This command is provided to the circuit by the LIN master to get the actual position of the stepper−motor. This position (<ActPos[15:0]>) is returned in signed two's complement 16−bit format. One should note that according to the programmed stepping mode, the LSB's of <ActPos[15:0]> may have no meaning and should be assumed to be '0', as prescribed in Position Ranges. GetActualPos also provides a quick status of the circuit and the stepper−motor, identical to that obtained by command GetStatus (see further).

**Note:** A GetActualPos command will not attempt to reset any flag.

GetActualPos corresponds to the following LIN reading frames.

1. four data bytes in−frame response with direct ID (type #5)

**Table 40. READING FRAME TYPE #5**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 1 | 0 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | ESW | AD[6:0] | | | | | | |
| 2 | Data 2 | ActPos[15:8] | | | | | | | |
| 3 | Data 3 | ActPos[7:0] | | | | | | | |
| 4 | Data 4 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |
| 5 | Checksum | Checksum over data | | | | | | | |

Where:

(*) According to parity computation

ID[5:0]: Dynamically allocated direct identifier. There should be as many dedicated identifiers to this GetActualPos command as there are stepper−motors connected to the LIN bus.

**Note:** Bit 5 and bit 4 in byte 0 indicate the number of data bytes.

2. The master sends either a type#7 or type#8 preparing frame. After the type#7 or #8 preparing frame, the master sends a reading frame type#6 to retrieve the circuit's in−frame response.

**Table 41. GetActualPos PREPARING FRAME TYPE #7**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | 1 | CMD[6:0] = 0x00 | | | | | | |
| 2 | Data 2 | 1 | AD[6:0] | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

**Table 42. GetActualPos PREPARING FRAME TYPE #6**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | ESW | AD[6:0] | | | | | | |
| 2 | Data 2 | ActPos[15:8] | | | | | | | |
| 3 | Data 3 | ActPos[7:0] | | | | | | | |
| 4 | Data 4 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |
| 5 | Data 5 | Device ID Code | | | | 1 | UV2 | UV3 | LIN_E |
| 6 | Data 6 | 0xFF | | | | | | | |
| 7 | Data 7 | 0xFF | | | | | | | |
| 8 | Data 8 | 0xFF | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:
(*) According to parity computation

**Table 43. GetActualPos PREPARING FRAME TYPE #8**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD =80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x00 | | | | | | |
| 3 | Data 3 | 1 | AD[6:0] | | | | | | |
| 4 | Data 4 | Data4[7:0] FF | | | | | | | |
| 5 | Data 5 | Data5[7:0] FF | | | | | | | |
| 6 | Data 6 | Data6[7:0] FF | | | | | | | |
| 7 | Data 7 | Data7[7:0] FF | | | | | | | |
| 8 | Data 8 | Data8[7:0] FF | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

**Table 44. GetActualPos READING FRAME TYPE #6**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | ESW | AD[6:0] | | | | | | |
| 2 | Data 2 | ActPos[15:8] | | | | | | | |
| 3 | Data 3 | ActPos[7:0] | | | | | | | |
| 4 | Data 4 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |
| 5 | Data 5 | Device ID Code | | | | 1 | UV2 | UV3 | LIN_E |
| 6 | Data 6 | 0xFF | | | | | | | |
| 7 | Data 7 | 0xFF | | | | | | | |
| 8 | Data 8 | 0xFF | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

### GetFullStatus

This command is provided to the circuit by the LIN master to get a complete status of the circuit and the stepper−motor. Refer to RAM Registers and Flags Table to see the meaning of the parameters sent to the LIN master.

**Note:** A GetFullStatus command will attempt to reset flags<TW>,<TSD>,<UV2>,<UV3>,<UV>,<<ElDef>, <StepLoss>, <OVC1>, <OVC2>, <VddReset>, <Stall> and <AbsStall>.

The master sends either type#7 or type#8 preparing frame. GetFullStatus corresponds to 2 successive LIN in−frame responses with **0x3D** indirect ID.

**Note:** It is not mandatory for the LIN master to initiate the second in−frame response if the data in the second response frame is not needed by the application.

1. The master sends a type #7 preparing frame. After the type#7 preparing frame, the master sends a reading frame type#6 to retrieve the circuit's in−frame response.

**Table 45. GetFullStatus PREPARING FRAME TYPE #7**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | 1 | CMD[6:0] = 0x01 | | | | | | |
| 2 | Data 2 | 1 | AD[6:0] | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

**Table 46. GetFullStatus READING FRAME TYPE #6 (1)**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | 1 | AD[6:0] | | | | | | |
| 2 | Data 2 | Irun[3:0] | | | | Ihold[3:0] | | | |
| 3 | Data 3 | Vmax[3:0] | | | | Vmin[3:0] | | | |
| 4 | Data 4 | AccShape | StepMode[1:0] | | Shaft | Acc[3:0] | | | |
| 5 | Data 5 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |
| 6 | Data 6 | Motion[2:0] | | | ESW | OVC1 | OVC2 | Stall | 0 |
| 7 | Data 7 | PWMFreq | I_BOOST_ENB | Tstab[1] | Tstab[0] | TimeE | DataE | HeadE | BitE |
| 8 | Data 8 | AbsThr[3:0] | | | | UV2 | UV3Thr[2:0] | | |
| 9 | Checksum | Checksum over data | | | | | | | |

**Table 47. GetFullStatus READING FRAME TYPE #6 (2)**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | 1 | AD[6:0] | | | | | | |
| 2 | Data 2 | ActPos[15:8] | | | | | | | |
| 3 | Data 3 | ActPos[7:0] | | | | | | | |
| 4 | Data 4 | TagPos[15:8] | | | | | | | |
| 5 | Data 5 | TagPos[7:0] | | | | | | | |
| 6 | Data 6 | SecPos[7:0] | | | | | | | |
| 7 | Data 7 | FS2StallEn[2:0] | | | 1 | DC100 | SecPos[10:8] | | |
| 8 | Data 8 | AbsStall | 0 | 0 | MinSamples[2:0] | | | DC100StEn | PWMJEn |
| 9 | Checksum | Checksum over data | | | | | | | |

Where: (*) According to parity computation

2. The master sends a type #8 preparing frame. After the type#8 preparing frame, the master sends a reading frame type#6 to retrieve the circuit's in−frame response.

**Table 48. GetFullStatus PREPARING FRAME TYPE#8**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD =80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x01 | | | | | | |
| 3 | Data 3 | 1 | AD[6:0] | | | | | | |
| 4 | Data 4 | Data4[7:0] FF | | | | | | | |
| 5 | Data 5 | Data5[7:0] FF | | | | | | | |
| 6 | Data 6 | Data6[7:0] FF | | | | | | | |
| 7 | Data 7 | Data7[7:0] FF | | | | | | | |
| 8 | Data 8 | Data8[7:0] FF | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

**Table 49. GetFullStatus READING FRAME TYPE #6 (1)**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | 1 | AD[6:0] | | | | | | |
| 2 | Data 2 | Irun[3:0] | | | | Ihold[3:0] | | | |
| 3 | Data 3 | Vmax[3:0] | | | | Vmin[3:0] | | | |
| 4 | Data 4 | AccShape | StepMode[1:0] | | Shaft | Acc[3:0] | | | |
| 5 | Data 5 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |
| 6 | Data 6 | Motion[2:0] | | | ESW | OVC1 | OVC2 | Stall | 0 |
| 7 | Data 7 | PWMFreq | I_BOOST_ENB | Tstab[1] | Tstab[0] | TimeE | DataE | HeadE | BitE |
| 8 | Data 8 | AbsThr[3:0] | | | | UV2 | UV3Thr[2:0] | | |
| 6 | Checksum | Checksum over data | | | | | | | |

**Table 50. GetFullStatus READING FRAME TYPE #6 (2)**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | 1 | AD[6:0] | | | | | | |
| 2 | Data 2 | ActPos[15:8] | | | | | | | |
| 3 | Data 3 | ActPos[7:0] | | | | | | | |
| 4 | Data 4 | TagPos[15:8] | | | | | | | |
| 5 | Data 5 | TagPos[7:0] | | | | | | | |
| 6 | Data 6 | SecPos[7:0] | | | | | | | |
| 7 | Data 7 | FS2StallEn[2:0] | | | 1 | DC100 | SecPos[10:8] | | |
| 8 | Data 8 | AbsStall | 0 | 0 | MinSamples[2:0] | | | DC100StEn | PWMJEn |
| 9 | Checksum | Checksum over data | | | | | | | |

**GetOTPparam**

This command is provided to the circuit by the LIN master after a preparing frame (see Preparing frames), to read the content of an OTP memory segment which address was specified in the preparation frame.

`GetOTPparam` corresponds to a LIN in−frame response with **0x3D** indirect ID.

1. The master sends a type #7 preparing frame. After the type#7 preparing frame, the master sends a reading frame type#6 to retrieve the circuit's in−frame response.

**Table 51. GetOTPparam PREPARING FRAME TYPE #7**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | 1 | CMD[6:0] = 0x02 | | | | | | |
| 2 | Data 2 | 1 | AD[6:0] | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

**Table 52. GetOTPparam READING FRAME TYPE #6**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | OTP byte @0x00 | | | | | | | |
| 2 | Data 2 | OTP byte @0x01 | | | | | | | |
| 3 | Data 3 | OTP byte @0x02 | | | | | | | |
| 4 | Data 4 | OTP byte @0x03 | | | | | | | |
| 5 | Data 5 | OTP byte @0x04 | | | | | | | |
| 6 | Data 6 | If SecPosA = 0 OTP byte @0x05, else OTP byte @0x08 | | | | | | | |
| 7 | Data 7 | If SecPosA = 0 OTP byte @0x06, else OTP byte @0x09 | | | | | | | |
| 8 | Data 8 | OTP byte @0x07 | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

(*) According to parity computation

2. The master sends a type #8 preparing frame. After the type#8 preparing frame, the master sends a reading frame type#6 to retrieve the circuit's in−frame response.

**Table 53. GetOTPparam PREPARING FRAME TYPE #8**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD =80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x02 | | | | | | |
| 3 | Data 3 | 1 | AD[6:0] | | | | | | |
| 4 | Data 4 | Data4[7:0] FF | | | | | | | |
| 5 | Data 5 | Data5[7:0] FF | | | | | | | |
| 6 | Data 6 | Data6[7:0] FF | | | | | | | |
| 7 | Data 7 | Data7[7:0] FF | | | | | | | |
| 8 | Data 8 | Data8[7:0] FF | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

**Table 54. GetOTPparam READING FRAME TYPE #6**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | Data 1 | OTP byte @0x00 | | | | | | | |
| 2 | Data 2 | OTP byte @0x01 | | | | | | | |
| 3 | Data 3 | OTP byte @0x02 | | | | | | | |
| 4 | Data 4 | OTP byte @0x03 | | | | | | | |
| 5 | Data 5 | OTP byte @0x04 | | | | | | | |
| 6 | Data 6 | If SecPosA = 0 OTP byte @0x05, else OTP byte @0x08 | | | | | | | |
| 7 | Data 7 | If SecPosA = 0 OTP byte @0x06, else OTP byte @0x09 | | | | | | | |
| 8 | Data 8 | OTP byte @0x07 | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

### GetStatus

This command is provided to the circuit by the LIN master to get a quick status (compared to that of GetFullStatus command) of the circuit and of the stepper−motor. Refer to Flags Table to see the meaning of the parameters sent to the LIN master.

**Note:** A GetStatus command will attempt to reset flags <TW>, <TSD>, <UV>, <ElDef>, <StepLoss> and <VddReset>.

If there is only an open coil detected the < ElDef > flag will be cleared after the GetStatus command. If <ElDef> is set due to a short on one of the coils, the <Eldef> can only be cleared via a GetFullStatus command.

GetStatus corresponds to a 2 data bytes LIN in−frame response with a direct ID (type #5).

**Table 55. GetStatus READING FRAME TYPE #5**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | ESW | AD[6:0] | | | | | | |
| 2 | Data 2 | VddReset | StepLoss | ElDef | UV | TSD | TW | Tinfo[1:0] | |
| 3 | Checksum | Checksum over data | | | | | | | |

Where:

(*) According to parity computation

ID[5:0]: Dynamically allocated direct identifier. There should be as many dedicated identifiers to this GetStatus command as there are stepper−motors connected to the LIN bus.

### GotoSecurePosition

This command is provided by the LIN master to one or all of the stepper−motors to move to the secure position <SecPos[10:0]>. It can also be internally triggered if the LIN bus communication is lost, after an initialization phase, or prior to going into sleep mode. See the priority encoder description for more details. The priority encoder table also acknowledges the cases where a GotoSecurePosition command will be ignored.

**Note:** The dynamic ID allocation has to be assigned to ‘General Purpose 2 Data bytes’ ROM pointer, i.e. ‘0000’. The command is decoded only from the command data.

GotoSecurePosition corresponds to the following LIN writing frame (type #1).

**Table 56. GotoSecurePosition WRITING FRAME TYPE #1**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data | 1 | CMD[6:0] = 0x04 | | | | | | |
| 2 | Data | Broad | AD[6:0] | | | | | | |
| 3 | Checksum | | Checksum over data | | | | | | |

Where:

(*) according to parity computation

Broad: If Broad = '0' all the stepper motors connected to the LIN bus will reach their secure position

**HardStop**

This command will be internally triggered when an electrical problem is detected in one or both coils, leading to shutdown mode. If this occurs while the motor is moving, the <StepLoss> flag is raised to allow warning of the LIN master at the next GetStatus command that steps may have been lost. Once the motor is stopped, <ActPos> register is copied into <TagPos> register to ensure keeping the stop position.

**Note:** The dynamic ID allocation has to be assigned to 'General Purpose 2 Data bytes' ROM pointer, i.e. '0000'. The command is decoded only from the command data.

A hardstop command can also be issued by the LIN master for some safety reasons. It corresponds then to the following two data bytes LIN writing frame (type #1).

**Table 57. HardStop WRITING FRAME TYPE #1**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data | 1 | CMD[6:0] = 0x05 | | | | | | |
| 2 | Data | Broad | AD[6:0] | | | | | | |
| 3 | Checksum | | Checksum over data | | | | | | |

Where:

(*) according to parity computation

Broad: If broad = '0' all stepper motors connected to the LIN bus will stop

**ResetPosition**

This command is provided to the circuit by the LIN master to reset <ActPos> and <TagPos> registers to zero. This can be helpful to prepare for instance a relative positioning. The reset position command sets the internal flag "Reference done".

**Note:** The dynamic ID allocation has to be assigned to 'General Purpose 2 Data bytes' ROM pointer, i.e. '0000'. The command is decoded only from the command data.

ResetPosition corresponds to the following LIN writing frames (type #1).

**Table 58. ResetPosition WRITING FRAME TYPE #1**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data | 1 | CMD[6:0] = 0x06 | | | | | | |
| 2 | Data | Broad | AD[6:0] | | | | | | |
| 3 | Checksum | | Checksum over data | | | | | | |

Where:

(*) according to parity computation

Broad: If broad = '0' all the circuits connected to the LIN bus will reset their <ActPos> and <TagPos> registers

### SetDualPosition

This command is provided to the circuit by the LIN master in order to perform a positioning of the motor using two different velocities. See Dual Positioning. After Dual positioning the internal flag “Reference done” is set.

**Note:** This sequence cannot be interrupted by another positioning command.

`SetDualPosition` corresponds to the following LIN writing frame with **0x3C** identifier (type #4).

**Table 59. SetDualPositioning WRITING FRAME TYPE #4**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD = 0x80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x08 | | | | | | |
| 3 | Data 3 | Broad | AD[6:0] | | | | | | |
| 4 | Data 4 | Vmax[3:0] | | | | Vmin[3:0] | | | |
| 5 | Data 5 | Pos1[15:8] | | | | | | | |
| 6 | Data 6 | Pos1[7:0] | | | | | | | |
| 7 | Data 7 | Pos2[15:8] | | | | | | | |
| 8 | Data 8 | Pos2[7:0] | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

Broad: If broad = ‘0’ all the circuits connected to the LIN bus will run the dual positioning

Vmax[3:0]: Max velocity for first motion

Vmin[3:0]: Min velocity for first motion and velocity for the second motion

Pos1[15:0]: First position to be reached during the first motion

Pos2[15:0]: Relative position of the second motion

### SetStallParam

This command sets the motion detection parameters and the related stepper−motor parameters, such as the minimum and maximum velocity, the run and hold current, acceleration and step mode. See Motion detection for the meaning of the parameters sent by the LIN Master.

`SetStallParam` corresponds to a **0x3C** LIN command (type #4).

**Table 60. SetStallParam WRITING FRAME TYPE #4**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD = 0x80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x16 | | | | | | |
| 3 | Data 3 | Broad | AD[6:0] | | | | | | |
| 4 | Data 4 | Irun[3:0] | | | | Ihold[3:0] | | | |
| 5 | Data 5 | Vmax[3:0] | | | | Vmin[3:0] | | | |
| 6 | Data 6 | MinSamples[2:0] | | | Shaft | Acc[3:0] | | | |
| 7 | Data 7 | AbsThr[3:0] | | | | 0 | UV3Thr[2:0] | | |
| 8 | Data 8 | FS2StallEn[2:0] | | | AccShape | StepMode[1:0] | | DC100StEn | PWMJEn |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

Broad: If Broad = ‘0’ all the circuits connected to the LIN bus will set the parameters in their RAMs as requested

**SetMotorParam**

This command is provided to the circuit by the LIN master to set the values for the stepper motor parameters (listed below) in RAM. Refer to RAM Registers to see the meaning of the parameters sent by the LIN master.

**Important:** If a `SetMotorParam` occurs while a motion is ongoing, it will modify at once the motion parameters (see Position Controller). Therefore the application should not change other parameters than `<Vmax>` and `<Vmin>` while a motion is running, otherwise correct positioning cannot be guaranteed.

`SetMotorParam` corresponds to the following LIN writing frame with **0x3C** identifier (type #4).

**Table 61. SetMotorParam WRITING FRAME TYPE #4**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD = 0x80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x09 | | | | | | |
| 3 | Data 3 | Broad | AD[6:0] | | | | | | |
| 4 | Data 4 | Irun[3:0] | | | | Ihold[3:0] | | | |
| 5 | Data 5 | Vmax[3:0] | | | | Vmin[3:0] | | | |
| 6 | Data 6 | SecPos[10:8] | | | Shaft | Acc[3:0] | | | |
| 7 | Data 7 | SecPos[7:0] | | | | | | | |
| 8 | Data 8 | TStab[1] | PWMfreq | TStab[0] | AccShape | StepMode[1:0] | | I_BOOST_ENB | PWMJEn |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

Broad: If Broad = ‘0’ all the circuits connected to the LIN bus will set the parameters in their RAMs as requested

**SetOTPparam**

This command is provided to the circuit by the LIN master to program the content `D[7:0]` of the OTP memory byte OTPA[3]#, OTPA[2:0] and to zap it.

**Important:** This command must be sent under a specific $V_{BB}$ voltage value. See parameter VBBOTP in DC Parameters. This is a mandatory condition to ensure reliable zapping.

`SetOTPparam` corresponds to a **0x3C** LIN writing frames (type #4).

**Table 62. SetOTPparam WRITING FRAME TYPE #4**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD = 0x80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x10 | | | | | | |
| 3 | Data 3 | Broad | AD[6:0] | | | | | | |
| 4 | Data 4 | 1 | 1 | 1 | 1 | OTPA[3]# | OTPA[2:0] | | |
| 5 | Data 5 | D[7:0] | | | | | | | |
| 6 | Data 6 | 0xFF | | | | | | | |
| 7 | Data 7 | 0xFF | | | | | | | |
| 8 | Data 8 | 0xFF | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

Broad: If Broad = ‘0’ all the circuits connected to the LIN bus will set the parameters in their OTP memories as requested

NOTE: OTPA[3]# is inverted bit.

**SetPosition**

This command is provided to the circuit by the LIN master to drive one or two motors to a given absolute position. See Positioning for more details.

1. Two (2) Data bytes frame with a direct ID (type #3)

The priority encoder table (See Priority Encoder) describes the cases where a `SetPosition` command will be ignored.

SetPosition corresponds to the following LIN write frames.

**Table 63. SetPosition WRITING FRAME TYPE #3**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | Pos[15 :8] | | | | | | | |
| 2 | Data 2 | Pos[7 :0] | | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

Where:

(*) According to parity computation

ID[5:0]: Dynamically allocated direct identifier. There should be as many dedicated identifiers to this SetPosition command as there are stepper−motors connected to the LIN bus.

2. Four (4) Data bytes frame with general purpose identifier (type #1).

**Note:** The dynamic ID allocation has to be assigned to ‘General Purpose 4 Data bytes’ ROM pointer, i.e. ‘0001’.

**Table 64. SetPosition WRITING FRAME TYPE #1**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 1 | 0 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | 1 | CMD[6:0] = 0x0B | | | | | | |
| 2 | Data 2 | Broad | AD[6:0] | | | | | | |
| 3 | Data 3 | Pos[15:8] | | | | | | | |
| 4 | Data 4 | Pos[7:0] | | | | | | | |
| 5 | Checksum | Checksum over data | | | | | | | |

Where:

(*) According to parity computation

Broad: If broad = ‘0’ all the stepper motors connected to the LIN will must go to `Pos[15:0]`.

3. Two (2) motors positioning frame with **0x3C** identifier (type #4)

**Table 65. SetPosition WRITING FRAME TYPE #4**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | AppCMD = 0x80 | | | | | | | |
| 2 | Data 2 | 1 | CMD[6:0] = 0x0B | | | | | | |
| 3 | Data 3 | 1 | AD1[6:0] | | | | | | |
| 4 | Data 4 | Pos1[15:8] | | | | | | | |
| 5 | Data 5 | Pos1[7:0] | | | | | | | |
| 6 | Data 6 | 1 | AD2[6:0] | | | | | | |
| 7 | Data 7 | Pos2[15:8] | | | | | | | |
| 8 | Data 8 | Pos2[7:0] | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

Adn[6:0] : Motor #n physical address (n ∈ [1,2]).

Posn[15:0] : Signed 16−bit position set−point for motor #n.

**SetPositionShort**

This command is provided to the circuit by the LIN Master to drive one, two or four motors to a given absolute position. The Short Position is only 11 bits and mapped as half steps over the position range. The positioner still perform the motion with the programmed `StepMode[1:0]`, but due to the reduced number of bits, the end position is always taken at a multiple of half steps (resolution). See Positioning for more details.

The physical address is coded on 4 bits, hence `SetPositionShort` can only be used with a network implementing a maximum of 16 slave nodes. These 4 bits are corresponding to the bits `PA[3:0]` in OTP memory (address 0x02) See Physical Address of the Circuit. For `SetPositionShort` it is recommended to set HW0, HW1 and HW2 to '1'.

The priority encoder table (See Priority Encoder) describes the cases where a `SetPositionShort` command will be ignored.

`SetPositionShort` corresponds to the following LIN writing frames:

1. Two (2) data bytes frame for one (1) motor, with specific identifier (type #2)

**Table 66. SetPositionShort WRITING FRAME TYPE #2**

| | | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Byte** | **Content** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | Pos[10:8] | | | Broad | AD [3:0] | | | |
| 2 | Data 2 | Pos [7:0] | | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

Where:

(*) According to parity computation

Broad: If broad = '0' all the stepper motors connected to the LIN bus will go to `Pos[10:0]`.

ID[5:0]: Dynamically allocated identifier to two data bytes `SetPositionShort` command.

2. Four (4) data bytes frame for two (2) motors, with specific identifier (type # 2)

**Table 67. SetPositionShort WRITING FRAME TYPE #2**

| | | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Byte** | **Content** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Identifier | * | * | 1 | 0 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | Pos1[10:8] | | | 1 | AD1[3:0] | | | |
| 2 | Data 2 | Pos1[7:0] | | | | | | | |
| 3 | Data 3 | Pos2[10:8] | | | 1 | AD2[3:0] | | | |
| 4 | Data 4 | Pos2[7:0] | | | | | | | |
| 5 | Checksum | Checksum over data | | | | | | | |

Where:

(*) according to parity computation

ID[5:0]: Dynamically allocated identifier to four data bytes `SetPositionShort` command.

Adn[3:0]: Motor #n physical address least significant bits (n ∈ [1,2]).

Posn[10:0]: Signed 11−bit position set point for Motor #n (see RAM Registers)

3. Eight (8) data bytes frame for four (4) motors, with specific identifier (type #2)

**Table 68. SetPositionShort WRITING FRAME TYPE #2**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 1 | 1 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | Pos1[10:8] | | | 1 | AD1[3:0] | | | |
| 2 | Data 2 | Pos1[7:0] | | | | | | | |
| 3 | Data 3 | Pos2[10:8] | | | 1 | AD2[3:0] | | | |
| 4 | Data 4 | Pos2[7:0] | | | | | | | |
| 5 | Data 5 | Pos3[10:8] | | | 1 | AD3[3:0] | | | |
| 6 | Data 6 | Pos3[7:0] | | | | | | | |
| 7 | Data 7 | Pos4[10:8] | | | 1 | AD4[3:0] | | | |
| 8 | Data 8 | Pos4[7:0] | | | | | | | |
| 9 | Checksum | Checksum over data | | | | | | | |

Where:

(*) according to parity computation

ID[5:0]: Dynamically allocated identifier to eight data bytes SetPositionShort command.

Adn[3:0]: Motor #n physical address least significant bits (n ∈ [1,4]).

Posn[10:0]: Signed 11−bit position set point for Motor #n (see RAM Registers)

### SetPosParam

This command is provided to the circuit by the LIN Master to drive one motor to a given absolute position. It also sets some of the values for the stepper motor parameters such as minimum and maximum velocity.

`SetPosParam` corresponds to a four (4) data bytes writing LIN frame with specific dynamically assigned identifier (type # 2).

**Table 69. SetPosParam WRITING FRAME TYPE #2**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 1 | 0 | ID3 | ID2 | ID5 | ID4 |
| 1 | Data 1 | Pos[15:8] | | | | | | | |
| 2 | Data 2 | Pos[7:0] | | | | | | | |
| 3 | Data 3 | Vmax[3:0] | | | | Vmin[3:0] | | | |
| 4 | Data 4 | AbsThr[3:0] | | | | Acc[3:0] | | | |
| 5 | Checksum | Checksum over data | | | | | | | |

Where:

(*) according to parity computation

Broad: If broad = ‘0’ all the stepper motors connected to the LIN bus will stop with deceleration.

ID[5:0]: Dynamically allocated direct identifier to 4 Data bytes `SetPosParam` command. There should be as many dedicated identifiers to this SetPosParam command as there are stepper−motors connected to the LIN bus.

Pos [15:0]: Signed 16−bit position set−point.

### Sleep

This command is provided to the circuit by the LIN master to put all the slave nodes connected to the LIN bus into sleep mode. If this command occurs during a motion of the motor, `TagPos` is reprogrammed to `SecPos` (provided `SecPos` is different from “100 0000 0000”), or a `SoftStop` is executed before going to sleep mode. See LIN 1.3 specification and Sleep Mode. The corresponding LIN frame is a master request command frame (identifier **0x3C**) with data byte 1 containing 0x00 while the followings contain 0xFF.

**Note:** SleepEnable needs to be set to 1 in order to allow the device to go to sleep. If SleepEnable is 0 the device will go into “stopped state”

**Table 70. SLEEP WRITING FRAME**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | Data 1 | 0x00 | | | | | | | |
| 2 | Data 2 | 0xFF | | | | | | | |
| 3 | Data 3 | 0xFF | | | | | | | |
| 4 | Data 4 | 0xFF | | | | | | | |
| 5 | Data 5 | 0xFF | | | | | | | |
| 6 | Data 6 | 0xFF | | | | | | | |
| 7 | Data 7 | 0xFF | | | | | | | |
| 8 | Data 8 | 0xFF | | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

### SoftStop

If a `SoftStop` command occurs during a motion of the stepper motor, it provokes an immediate deceleration to `Vmin` (see Minimum Velocity) followed by a stop, regardless of the position reached. Once the motor is stopped, `TagPos` register is overwritten with value in `ActPos` register to ensure keeping the stop position.

**Note:** The dynamic ID allocation has to be assigned to ‘General Purpose 2 Data bytes’ ROM pointer ‘0000’. The command is decoded only from the command data.

**Note:** A `SoftStop` command occurring during a `DualPosition` sequence is not taken into account.

Command `SoftStop` occurs in the following cases:

- The chip temperature rises above the thermal shutdown threshold (see DC Parameters and Temperature Management);
- The VBB drops under the UV3 level; (see DC Parameters and Battery Voltage Management);
- The LIN master requests a `SoftStop`. Hence `SoftStop` will correspond to the following two data bytes LIN writing frame (type #1).

**Table 71. SoftStop WRITING FRAME TYPE #1**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Identifier | * | * | 0 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 1 | Data 1 | 1 | CMD[6:0] = 0x0F | | | | | | |
| 2 | Data 2 | Broad | AD[6:0] | | | | | | |
| 3 | Checksum | Checksum over data | | | | | | | |

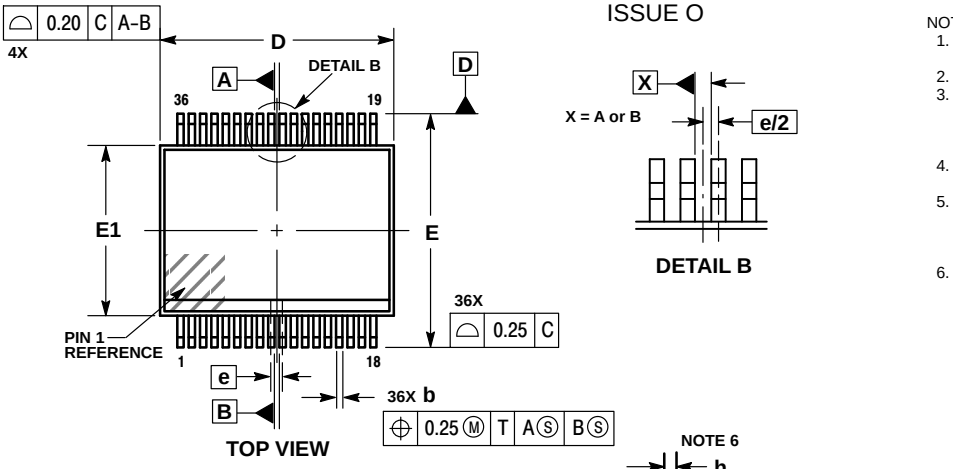
Where:

(*) according to parity computation

Broad: If broad = ‘0’ all the stepper motors connected to the LIN bus will stop with deceleration.

## PACKAGE DIMENSIONS

**SSOP36 EP**
CASE 940AB
ISSUE O

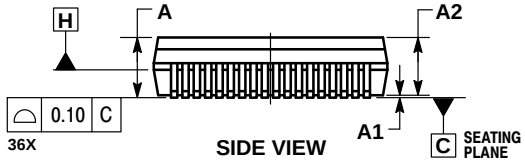


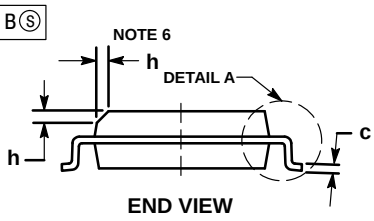


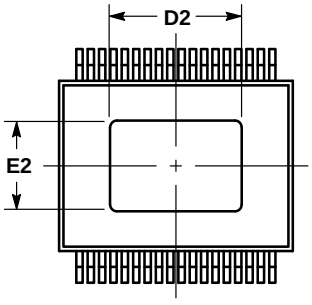


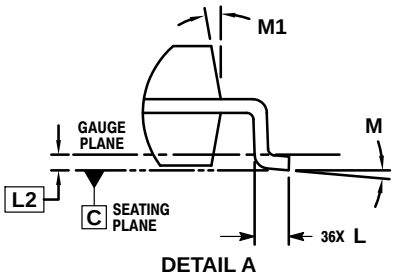
NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. CONTROLLING DIMENSION: MILLIMETERS.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.13 TOTAL IN EXCESS OF THE b DIMENSION AT MMC.
4. DIMENSION b SHALL BE MEASURED BETWEEN 0.10 AND 0.25 FROM THE TIP.
5. DIMENSIONS D AND E1 DO NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. DIMENSIONS D AND E1 SHALL BE DETERMINED AT DATUM H.
6. THIS CHAMFER FEATURE IS OPTIONAL. IF IT IS NOT PRESENT, A PIN ONE IDENTIFIER MUST BE LOACATED WITHIN THE INDICATED AREA.

| | MILLIMETERS | |
|---|---|---|
| DIM | MIN | MAX |
| A | --- | 2.65 |
| A1 | --- | 0.10 |
| A2 | 2.35 | 2.60 |
| b | 0.18 | 0.36 |
| c | 0.23 | 0.32 |
| D | 10.30 BSC | |
| D2 | 5.70 | 5.90 |
| E | 10.30 BSC | |
| E1 | 7.50 BSC | |
| E2 | 3.90 | 4.10 |
| e | 0.50 BSC | |
| h | 0.25 | 0.75 |
| L | 0.50 | 0.90 |
| L2 | 0.25 BSC | |
| M | 0° | 8° |
| M1 | 5° | 15° |

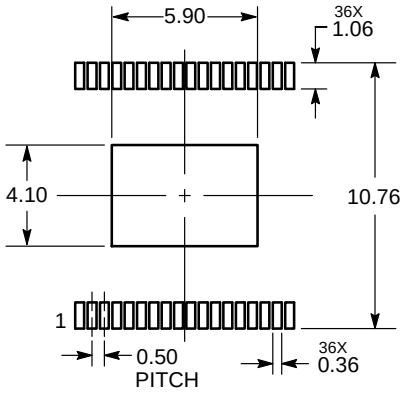


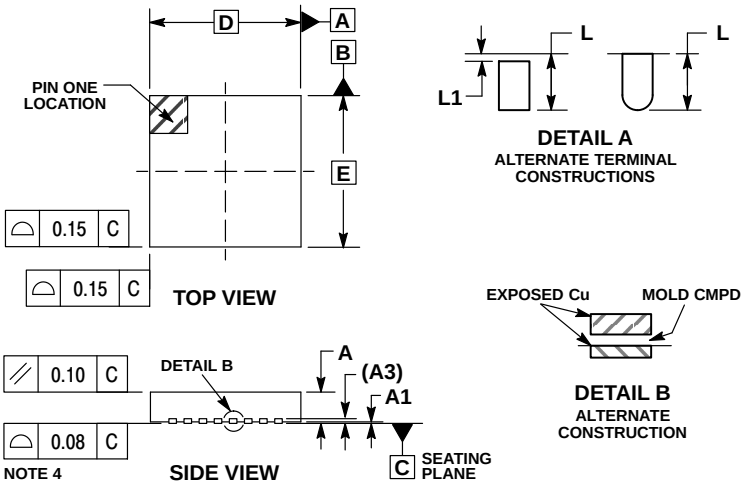

**SOLDERING FOOTPRINT**




DIMENSIONS: MILLIMETERS

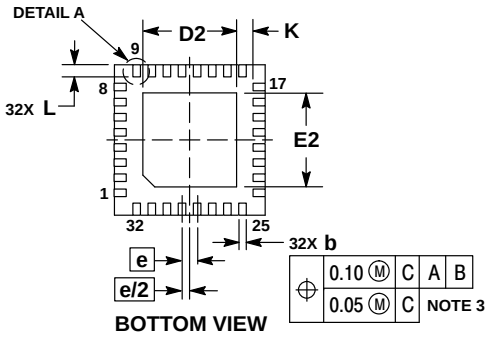
## PACKAGE DIMENSIONS

**QFN32 5x5, 0.5P**
CASE 488AM
ISSUE A

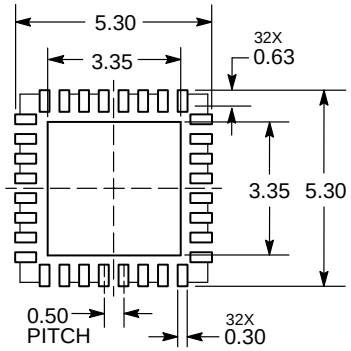



NOTES:
1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M, 1994.
2. CONTROLLING DIMENSION: MILLIMETERS.
3. DIMENSION b APPLIES TO PLATED TERMINAL AND IS MEASURED BETWEEN 0.15 AND 0.30MM FROM THE TERMINAL TIP.
4. COPLANARITY APPLIES TO THE EXPOSED PAD AS WELL AS THE TERMINALS.

| | MILLIMETERS | |
|---|---|---|
| DIM | MIN | MAX |
| A | 0.80 | 1.00 |
| A1 | --- | 0.05 |
| A3 | 0.20 REF | |
| b | 0.18 | 0.30 |
| D | 5.00 BSC | |
| D2 | 2.95 | 3.25 |
| E | 5.00 BSC | |
| E2 | 2.95 | 3.25 |
| e | 0.50 BSC | |
| K | 0.20 | --- |
| L | 0.30 | 0.50 |
| L1 | --- | 0.15 |




### RECOMMENDED SOLDERING FOOTPRINT*




DIMENSION: MILLIMETERS

*For additional information on our Pb−Free strategy and soldering details, please download the ON Semiconductor Soldering and Mounting Techniques Reference Manual, SOLDERRM/D.

FLEXMOS is a trademark of Semiconductor Components Industries, LLC (SCILLC).

**ON Semiconductor** and ON are registered trademarks of Semiconductor Components Industries, LLC (SCILLC). SCILLC owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of SCILLC's product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent−Marking.pdf. SCILLC reserves the right to make changes without further notice to any products herein. SCILLC makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does SCILLC 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 special, consequential or incidental damages. "Typical" parameters which may be provided in SCILLC 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. SCILLC does not convey any license under its patent rights nor the rights of others. SCILLC 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 SCILLC product could create a situation where personal injury or death may occur. Should Buyer purchase or use SCILLC products for any such unintended or unauthorized application, Buyer shall indemnify and hold SCILLC 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 SCILLC was negligent regarding the design or manufacture of the part. SCILLC is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

### PUBLICATION ORDERING INFORMATION

**LITERATURE FULFILLMENT**:
Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
**Phone**: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
**Fax**: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
**Email**: orderlit@onsemi.com

**N. American Technical Support**: 800−282−9855 Toll Free USA/Canada
**Europe, Middle East and Africa Technical Support:**
Phone: 421 33 790 2910
**Japan Customer Focus Center**
Phone: 81−3−5817−1050

**ON Semiconductor Website**: **www.onsemi.com**

**Order Literature**: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative