# AMIS-30624, NCV70624

# I²C Micro-stepping Motor Driver

## INTRODUCTION

The AMIS ĭ ĭ

## APPLICATIONS

The AMISĭ30624/NCV70624 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 mechatronic solutions with the driver chip mounted directly on the motor.

**Table 1. ORDERING INFORMATION**

| Part No. | Peak Current | End Market/Version | Package* | Shipping† |
|---|---|---|---|---|
| AMIS30624C6244G | 800 mA | Industrial High Voltage Version | SOIC−20 (Pb−Free) | Tube/Tray |
| AMIS30624C6244RG | 800 mA | | SOIC−20 (Pb−Free) | Tape & Reel |
| AMIS30624C6245G | 800 mA | | NQFP−32 (7 x 7 mm) (Pb−Free) | Tube/Tray |
| AMIS30624C6245RG | 800 mA | | NQFP−32 (7 x 7 mm) (Pb−Free) | Tape & Reel |
| NCV70624DW010G | 800 mA | Automotive High Temperature Version | SOIC−20 (Pb−Free) | Tube/Tray |
| NCV70624DW010R2G | 800 mA | | SOIC−20 (Pb−Free) | 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 Specification Brochure, BRD8011/D.

## QUICK REFERENCE DATA

**Table 2. ABSOLUTE MAXIMUM RATINGS**

| Parameter | Min | Max | Unit |
|---|---|---|---|
| $V_{BB}$, $V_{HW}$ | | | |

**Table of Contents**



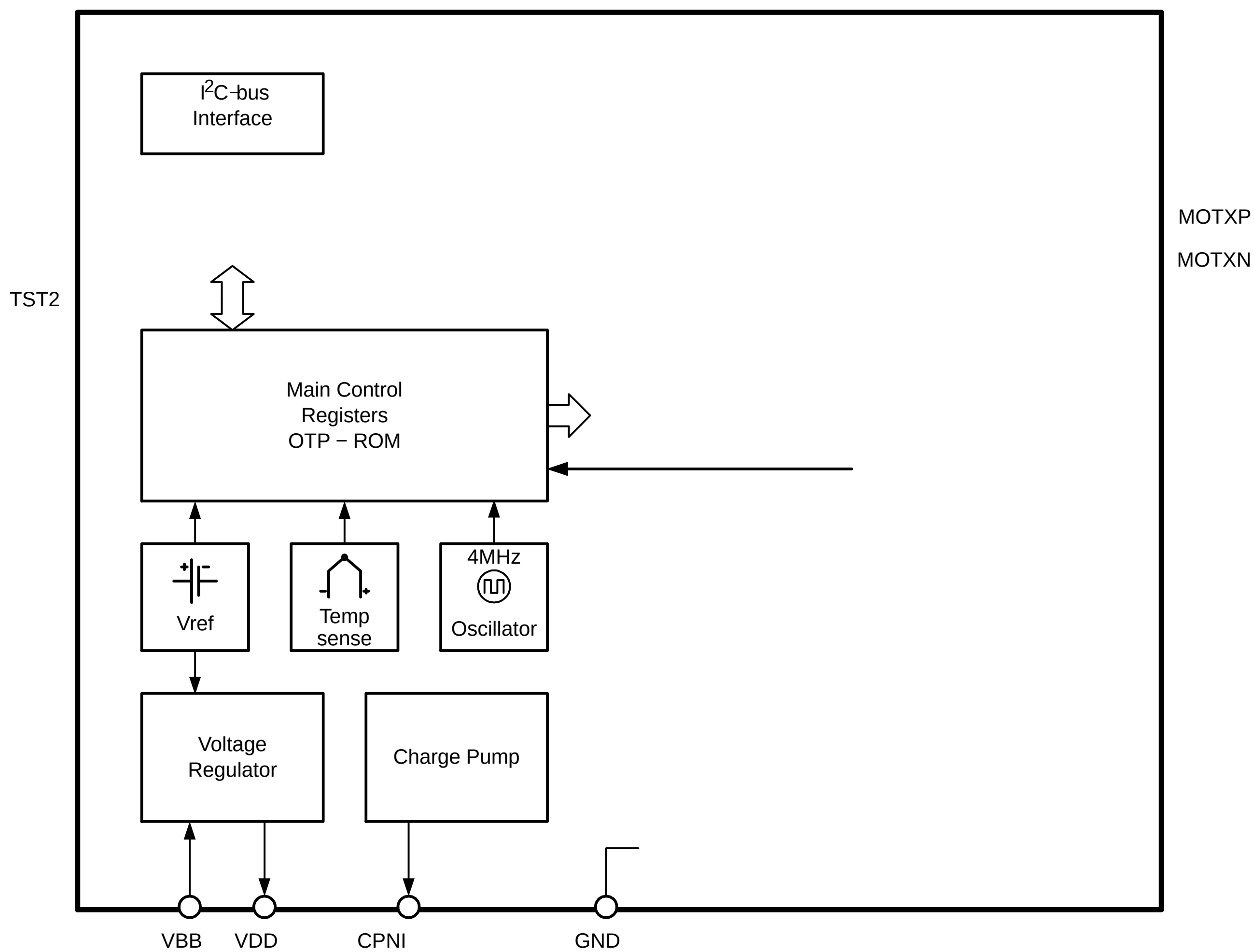
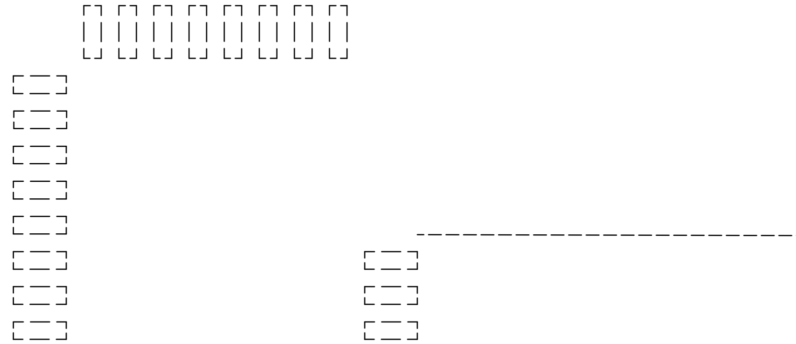



**Figure 1. Block Diagram**



## PACKAGE THERMAL RESISTANCE

The AMISĭ30624/NCV70624 is available in SOICĭ20 or optimized NQFPĭ32 packages. For cooling optimizations, the NQFP has an exposed thermal pad which has to be soldered to the PCB ground plane. The ground plane needs thermal vias to conduct the head to the bottom layer. Figures 3 and 4 give examples for good power distribution solutions.

For precise thermal cooling calculations the major thermal resistances of the devices are given. The thermal media to which the power of the devices has to be given are:

Static environmental air (via the case)

PCB board copper area (via the device pins and exposed pad)

The thermal resistances are presented in Table 5: DC Parameters.

The major thermal resistances of the device are the Rth from the junction to the ambient (Rthja) and the overall Rth from the junction to the leads (Rthjp).

The NQFP device is designed to provide superior thermal performance. Using an exposed die pad on the bottom surface of the package is mainly contributing to this performance. In order to take full advantage of the exposed pad, it is most important that the PCB has features to conduct heat away from the package. A thermal grounded pad with thermal vias can achieve this.

In the table below, one can find the values for the Rthja and Rthjp, simulated according to the JESDĭ51 norm:

| Package | Rth Junction−to−Leads and Exposed Pad − Rthjp | Rth Junction−to−Leads Rthjp | Rth Junction−to−Ambient Rthja (1S0P) | Rth Junction−to−Ambient Rthja (2S2P) |
|---|---|---|---|---|
| SOIC−20 | | 19 | 62 | 39 |
| NQFP−32 | 0,95 | | | |

## DC PARAMETERS

The DC parameters are guaranteed overtemperature and $V_{BB}$ in the operating range, unless otherwise specified. Convention: currents flowing into the circuit are defined as positive.

**Table 5. DC PARAMETERS**

| Symbol | Pin(s) | Parameter | Test Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| **MOTORDRIVER** | | | | | | | |
| $I_{MSmax,Peak}$ | MOTXP MOTXN MOTYP MOTYN | Max current through motor coil in normal operation | $V_{BB}$ = 14 V | | 800 | | mA |
| $I_{MSmax,RMS}$ | | Max rms current through coil in normal operation | $V_{BB}$ = 14 V | | 570 | | mA |
| $I_{MSabs}$ | | Absolute error on coil current (Note 5) | $V_{BB}$ = 14 V | −10 | | 10 | % |
| $I_{MSrel}$ | | Matching of X & Y coil currents | $V_{BB}$ = 14 V | −7 | 0 | 7 | % |
| $R_{DS(on)}$ | | On resistance for each motor pin at $I_{MSmax}$ (Note 6) | $V_{BB}$ = 12 V, $T_j$ = 50 C | | 0.50 | 1 | Ω |
| | | | $V_{BB}$ = 8 V, $T_j$ = 50 C | | 0.55 | 1 | Ω |
| | | | $V_{BB}$ = 12 V, $T_j$ = 150 C | | 0.70 | 1 | Ω |
| | | | $V_{BB}$ = 8 V, $T_j$ = 150 C | | 0.85 | 1 | Ω |
| $I_{MSL}$ | | Pulldown current | HiZ mode, $V_{BB}$ | | | | |

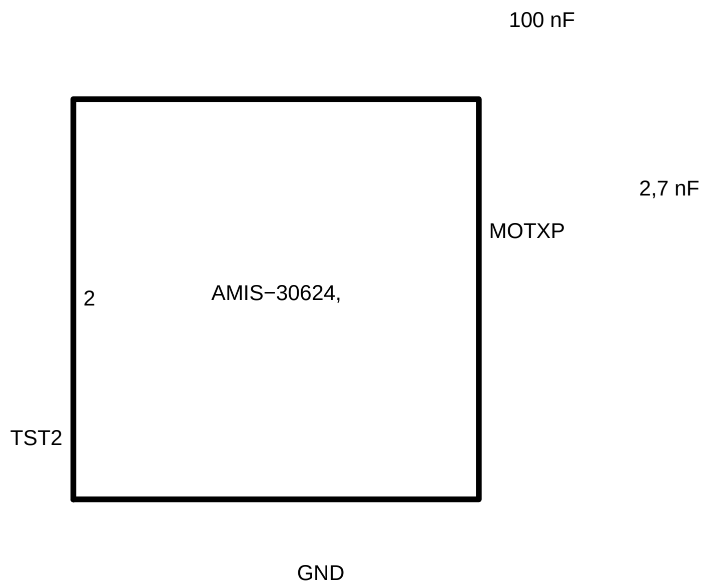
**Table 5. DC PARAMETERS**

| Symbol | Pin(s) | Parameter | Test Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| **SUPPLY AND VOLTAGE REGULATOR** | | | | | | | |
| $V_{DD}$ | $V_{DD}$ | Regulated internal supply (Note 13) | 8 V < $V_{BB}$ < 29 V | 4.75 | 5 | 5.50 | V |
| $V_{ddReset}$ | | Digital supply reset level @ power down (Note 14) | | | | 4.5 | V |
| $I_{ddLim}$ | | Current limitation | Pin shorted to ground $V_{BB}$ = 14 V | | | 45 | mA |
| **SWITCH INPUT AND HARDWIRE ADDRESS INPUT** | | | | | | | |
| $Rt\_{OFF}$ | SWI HW | Switch OPEN resistance (Note 15) | | 10 | | | kΩ |
| $Rt\_{ON}$ | | Switch ON resistance (Note 15) | Switch to GND or $V_{BB}$ | | | 2 | kΩ |
| $V_{bb\_sw}$ | | $V_{BB}$ range for guaranteed operation of SWI and HW | | 6 | | 29 | V |
| $I_{lim\_sw}$ | | Current limitation | Short to GND or $V_{bat}$ $V_{BB}$ = 29 V | 20 | 30 | 45 | mA |
| **TEST PIN** | | | | | | | |
| $V_{ihigh}$ | TST | Input level high | $V_{BB}$ = 14 V | 0.7 * $V_{dd}$ | | | V |
| $V_{ilow}$ | | Input level low | $V_{BB}$ = 14 V | | | 0.3 * $V_{dd}$ | V |
| $HW_{hyst}$ | | Hysteresis | $V_{BB}$ = 14 V | 0.075 * $V_{dd}$ | | | |

**Table 6. AC PARAMETERS**

| Symbol | Pin(s) | Parameter | Test Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| **SWITCH INPUT AND HARDWIRE ADDRESS INPUT** | | | | | | | |
| $T_{sw}$ | SWI HW | Scan pulse period (Note 21) | $V_{BB}$ = 14 V | | 1024 | | μs |
| $T_{sw\_on}$ | | Scan pulse duration (Note 21) | $V_{BB}$ = 14 V | | 128 | | μs |
| **MOTORDRIVER** | | | | | | | |
| $F_{pwm}$ | MOTxx | PWM frequency (Note 21) | PWMfreq = 0 (Note 22) | 20.6 | 22.8 | 25.0 | kHz |
| | | | PWMfreq = 1 (Note 22) | 41.2 | 45.6 | 50.0 | kHz |
| $F_{jit\_depth}$ | | PWM jitter modulation depth | PWMJen = 1 (Note 22) | | 10 | | % |
| $T_{brise}$ | | | | | | | |

## Typical Application



## POSITIONING PARAMETERS

### Stepping Modes

One of four possible stepping modes can be programmed:

Halfĭstepping

1/4 microĭstepping

1/8 microĭstepping

1/16 microĭstepping

### 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

### 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.

**Table 8. 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 |

### 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 11. POSITION RANGE**

| Command |
| --- |

Two DAC's to set the correct current ratio through X and Y.

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 or the charge pump.

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

The control logic block stores the information provided by the $I^2C$ 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 motordriver 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 motordriver part in order to manage possible problems and decide on internal actions and reporting to the $I^2C$ interface.

### Motion Detection

Motion detection is based on the backĭemf 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 backĭ

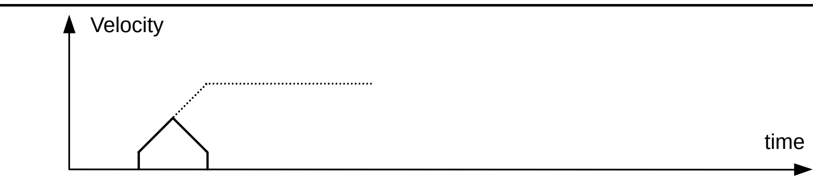
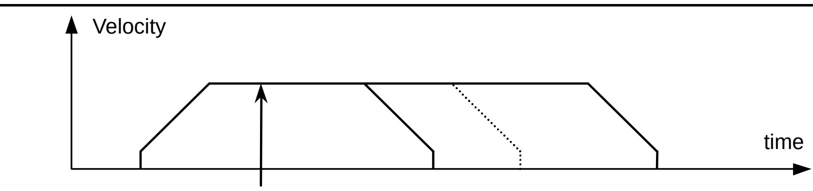
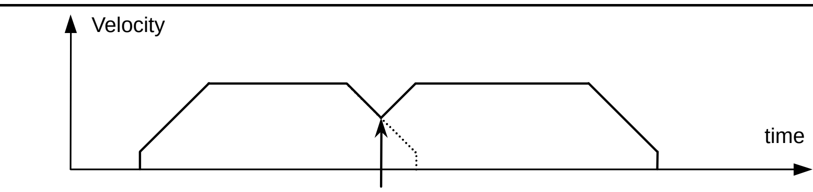
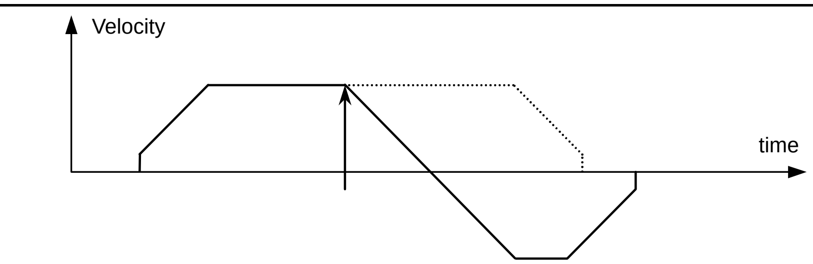
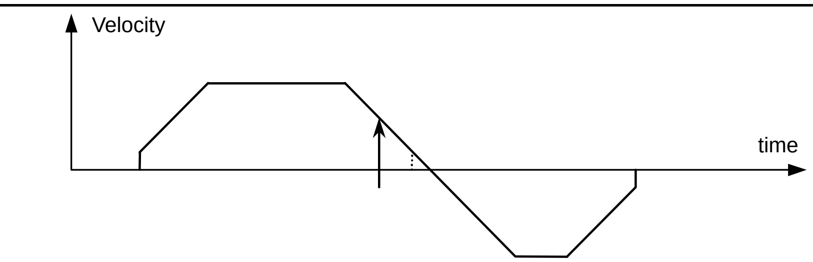
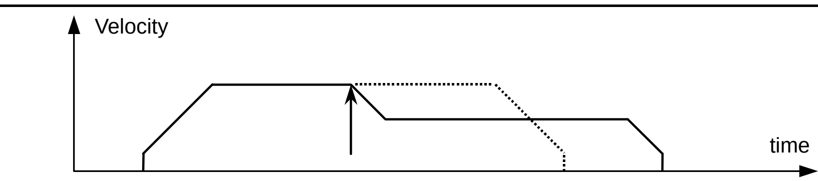
ĭ

**Table 13. 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 |

Different positioning examples are shown in the table below.

**Table 14. 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 23)<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 programming while motion is running. |   |

23. Reaching the end position is always guaranteed, however velocity rounding errors might occur after consecutive accelerations during a deceleration phase. The velocity rounding error will be removed at Vmin (e.g. at end of acceleration or when AccShape=1).

**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

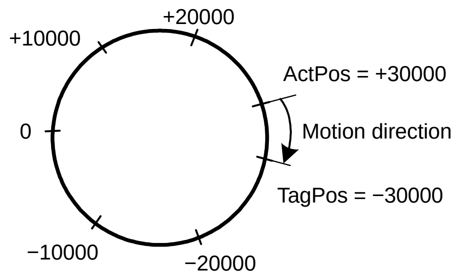



**Figure 9. Motion Direction is Function of Difference between ActPos and TagPos**

**Hardwired Address HW**

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

The HW pin is sensed via 2 switches. The DriveHS and DriveLS control lines are alternatively closing the top and bottom switch connecting HW 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 HW_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 HW_Cmp will be high.

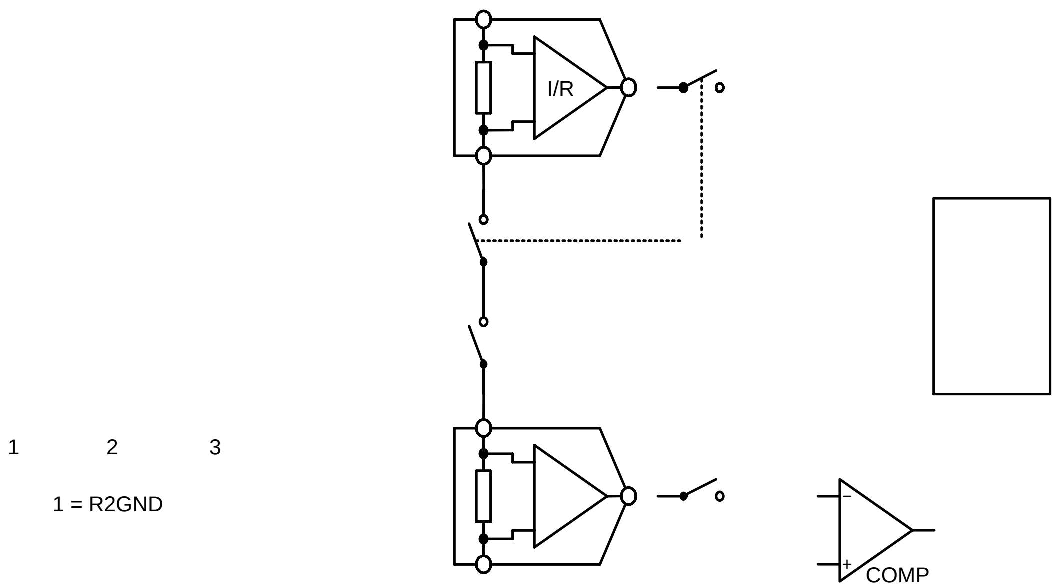


**Table 15. STATE DIAGRAM OF THE HW COMPARATOR**

| Previous State | DriveLS | DriveHS | HW_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 |

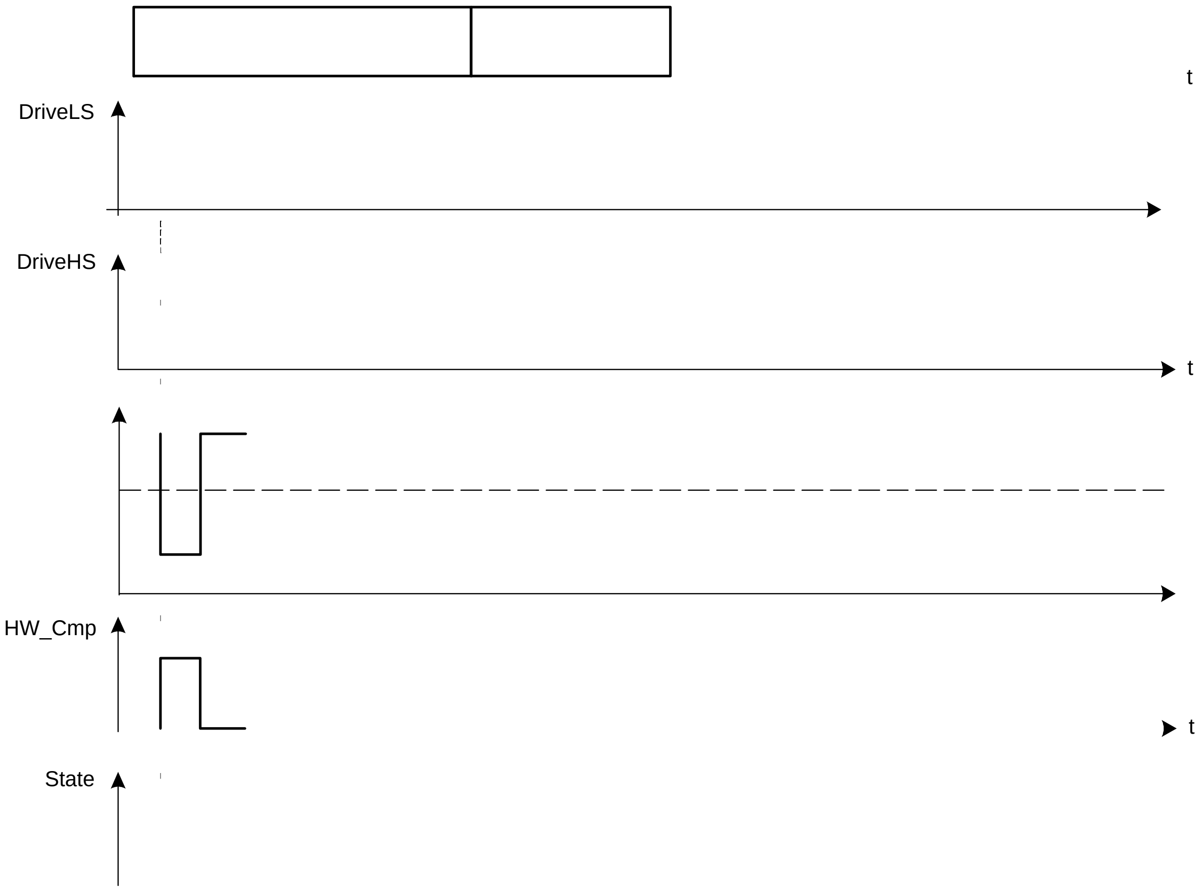
The logic is controlling the correct sequence in closing the switches and in interpreting the 32 μs debounced HW_Cmp output accordingly. The output of this small state ĭmachine is corresponding to:

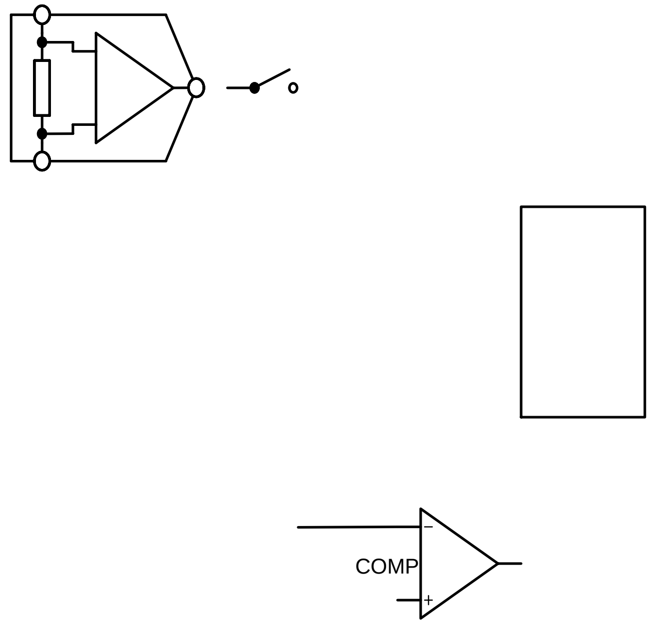
High or address = 1

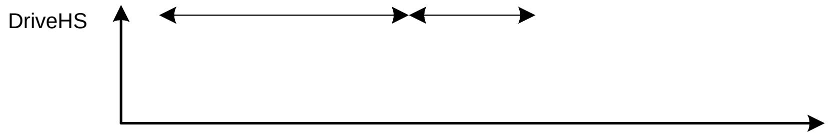
Low or address = 0

Floating

As illustrated in the table above (Table 15), the state is depending on the previous state, the condition of the 2 switch controls (DriveLS and DriveHS) and the output of


DriveLS
DriveHS
HW_Cmp
State
t
t
t


COMP

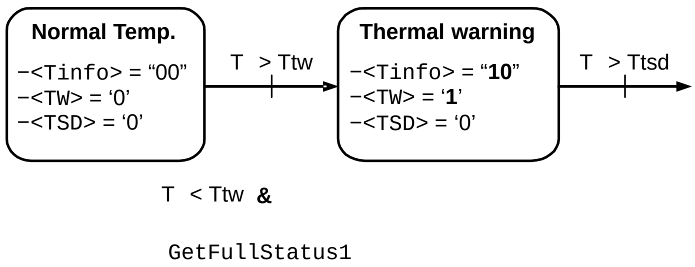

DriveHS

### Thermal Shutdown Mode

When thermal shutdown occurs, the circuit performs a <SoftStop> command and goes to motor shutdown mode (see Figure 14: State Diagram Temperature Management).

### Temperature Management

The AMISĭ30624/NCV70624 monitors temperature by means of two thresholds and one shutdown level, as illustrated in the state diagram and illustration of Figure 14: 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 GetFullStatus1 I²C frame.



## OTP Register

### OTP Memory Structure

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

**Table 17. OTP MEMORY STRUCTURE**

| Address | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0x00 | OSC3 | OSC2 | OSC1 | OSC0 | IREF3 | IREF2 | IREF1 | IREF0 |
| 0x01 | 0 | TSD2 | TSD1 | TSD0 | 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 | | |
| 0x07 | DelThr3 | DelThr2 | DelThr1 | DelThr0 | StepMode1 | StepMode0 | LOCKBT | LOCKBG |

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 18. OTP OVERWRITE PROTECTION**

| Lock Bit | Protected Bytes |
|---|---|
| LOCKBT (factory zapped before delivery) | 0x00 to 0x01 |
| LOCKBG | 0x00 to 0x07 |

The command used to load the application parameters via the I$^2$C bus in the RAM prior to an OTP Memory programming is SetMotorParam

Relative threshold used for the motion detection

| Index | DelThr | | | | DelThr Level (V) (*) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Disable |
| 1 | 0 | 0 | 0 | 1 | 0.25 |
| 2 | 0 | 0 | 1 | 0 | 0.50 |
| 3 | 0 | 0 | 1 | 1 | 0.75 |
| 4 | 0 | 1 | 0 | 0 | 1.00 |
| 5 | 0 | 1 | 0 | 1 | 1.25 |
| 6 | 0 | 1 | 1 | 0 | 1.50 |
| 7 | 0 | 1 | 1 | 1 | 1.75 |
| 8 | 1 | 0 | 0 | 0 | 2.00 |
| 9 | 1 | 0 | 0 | 1 | 2.25 |
| A | 1 | 0 | 1 | 0 | 2.50 |
| B | 1 | 0 | 1 | 1 | 2.75 |
| C | 1 | 1 | 0 | 0 | 3.00 |
| D | 1 | 1 | 0 | 1 | 3.25 |
| E | 1 | 1 | 1 | 0 | 3.50 |
| F | 1 | 1 | 1 | 1 | 3.75 |

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

Current amplitude value to be fed to each coil of the stepperĭmotor. The table below provides the 16 possible values for `<IRUN>`.

| Index | Irun | | | | Run Current (mA) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 59 |
| 1 | 0 | 0 | 0 | 1 | 71 |
| 2 | 0 | 0 | 1 | 0 | 84 |
| 3 | 0 | 0 | 1 | 1 | 100 |
| 4 | 0 | 1 | 0 | 0 | 119 |
| 5 | 0 | 1 | 0 | 1 | 141 |
| 6 | 0 | 1 | 1 | 0 | 168 |
| 7 | 0 | 1 | 1 | 1 | 200 |
| 8 | 1 | 0 | 0 | 0 | 238 |
| 9 | 1 | 0 | 0 | 1 | 283 |
| A | 1 | 0 | 1 | 0 | 336 |
| B | 1 | 0 | 1 | 1 | 400 |
| C | 1 | 1 | 0 | 0 | 476 |
| D | 1 | 1 | 0 | 1 | 566 |
| E | 1 | 1 | 1 | 0 | 673 |
| F | 1 | 1 | 1 | 1 | 800 |

Hold current for each coil of the stepperĭmotor. The table below provides the 16 possible values for `<IHOLD>`.

| Index | Ihold | | | | Hold Current (mA) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 59 |
| 1 | 0 | 0 | 0 | 1 | 71 |
| 2 | 0 | 0 | 1 | 0 | 84 |
| 3 | 0 | 0 | 1 | 1 | 100 |
| 4 | 0 | 1 | 0 | 0 | 119 |
| 5 | 0 | 1 | 0 | 1 | 141 |
| 6 | 0 | 1 | 1 | 0 | 168 |
| 7 | 0 | 1 | 1 | 1 | 200 |
| 8 | 1 | 0 | 0 | 0 | 238 |
| 9 | 1 | 0 | 0 | 1 | 283 |
| A | 1 | 0 | 1 | 0 | 336 |
| B | 1 | 0 | 1 | 1 | 400 |
| C | 1 | 1 | 0 | 0 | 476 |
| D | 1 | 1 | 0 | 1 | 566 |
| E | 1 | 1 | 1 | 0 | 673 |
| F | 1 | 1 | 1 | 1 | 0 |

**Note:** When the motor is stopped, the current is reduced from `<IRUN>` to `<IHOLD>`. In the case of 0 mA hold

**Note:** The Secure Position is coded on 11 bits only,

**Table 19. RAM REGISTERS**

| Register | Mnemonic | Length (bit) | Related Commands | Comment | Reset State |
|---|---|---|---|---|---|
| Actual position | ActPos | 16 | GetFullStatus2<br>GotoSecurePos<br>ResetPosition | 16−bit signed | |
| Last programmed Position | Pos/TagPos | 16/11 | GetFullStatus2<br>GotoSecurePos<br>ResetPosition<br>SetPosition | 16−bit signed or<br>11−bit signed for half stepping<br>(see Positioning) | |
| Acceleration shape | AccShape | 1 | GetFullStatus1<br>SetMotorParam<br>ResetToDefault | ‘0’ normal acceleration from Vmin to Vmax<br>‘1’ | |

**Table 20. FLAGS TABLE**

| Flag | Mnemonic | Length (bit) | Related Commands | Comment | Reset State |
|---|---|---|---|---|---|

### Priority Encoder

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

**Table 21. PRIORITY ENCODER**

| State → | Standby | Stopped | GotoPos | DualPosition | SoftStop | HardStop | ShutDown | HardUnder | ShutUnder |
|---|---|---|---|---|---|---|---|---|---|
| Command ↓ | | Motor Stopped, Ihold in Coils | Motor Motion Ongoing | No Influence on RAM and Tag-Pos | Motor | | | | |

31. <ElFlag> = <CPFail> or <UV2> or <ElDef> or <VDDreset>
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 GetFullStatus1 command will return the default parameters for <Vmax> and <Vmin> stored in RAM.
34. Shutdown state can be left only when <TSD> and <HS> flags are reset.
35. Flags can be reset only after the master could read them via a GetFullStatus1 command, and provided the physical conditions allow for it (normal temperature, correct battery voltage and no electrical or charge pump defect).
36. 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 command.
37. <SecEn> = ‘1’ when register <SecPos> is loaded with a value different from the most negative value (i.e. different from 0x400 = “100 0000 0000”).
38. <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>.
39. While in state <Stopped>, if <ActPos> <TagPos> there is a transition to state <GotoPos>. This transition has the lowest priority, meaning that <Stop>, <TSD>, etceteras are first evaluated for possible transitions.
40. If <StepLoss> is active, then SetPosition and GotoSecurePosition commands **are not** ignored. <StepLoss> can only be cleared by a GetFullStatus1 command.

**Motor Starting Phase**

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 25. Example of Possible Sequence used to Detect and Determine Cause of Motor Shutdown**

| Tj  Tsd or<br>$V_{BB}$  UV2 (>15s) or<br><ElDef> = '1' or<br><CPFail> = '1' (>15s) | SetPosition<br>f ame | GetFullStatus1 f ame | GetFullStatus1 f ame<br>... |
|---|---|---|---|
| − The circuit is driven in motor shutdown mode<br>− The application is <u>not</u> aware of this | − The position set−point is updated by the I²C 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 <TW> or <TSD> or <UV2> or <StepLoss> or <ElDef> or <CPFail> 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> or <CPFail> 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 AMISĭ30624/NCV70624.

**Note:** The <u>Priority Encoder</u> is describing the management of states and commands.

**Warning:** The application should limit the number of consecutive <u>GetFullStatus1</u> commands to try to get the AMISĭ30624/NCV70624 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 <u>GetFullStatus1</u> attempts to disable the protection of the Hĭbridges.

### 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 endĭstop, the velocity and as a result also the generated back emf, is disturbed. The AMISĭ30624/NCV70624 senses the back emf, calculates a moving average and compares the value with two independent threshold levels: Absolute threshold (<u>AbsThr[3:0]</u>) and Delta threshold (<DelThr[3:0]>). Instructions for correct use of these two levels in combination with three additional parameters (<MinSamples>, <FS2StallEn> and <DC100StEn>) are available in a dedicated Application Note "Robust Motion Control with AMIS

**Table 26. TRUTH TABLE**

| Condition | <DelStallLo> | <DelStallHi> | <AbsStall> | <Stall> |
|---|---|---|---|---|
| Vbemf < Average − DelThr | 1 | 0 | 0 | 1 |
| Vbemf > Average + DelThr | 0 | 1 | 0 | 1 |
| Vbemf < AbsThr | 0 | 0 | 1 | 1 |

By design, the motion will only be detected when the

**FS2StallEn**

If <AbsThr> or <DelThr> <> 0 (i.e. motion detection

## I$^2$C BUS DESCRIPTION

### General Description

AMISĭ30624/NCV70624 uses a simple biĭdirectional 2ĭwire bus for efficient interĭic control. This bus is called the Inter IC or I$^2$Cĭbus.

Features include:

- Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCK).
- Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exists at all times; master can operate as masterĭtransmitter or as master receiver.
- Serial, 8ĭbit oriented, biĭdirectional data transfers can be made up to 400 kb/s.
- Onĭchip filtering rejects spikes on the bus data line to preserve data integrity.
- No need to design bus interfaces because I$^2$Cĭbus interface is already integrated onĭchip.
- IC's can be added to or removed from a system without affecting any other circuits on the bus.

### Concept

The I$^2$Cĭbus consists of two wires, serial data (SDA) and serial clock (SCK), carrying information between the devices connected on the bus. Each device connected to the bus is recognized by a unique address and operates as either a transmitter or receiver, depending on the function of the device. AMISĭ30624/NCV70624 can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. AMISĭ30624/NCV70624 is a slave device. See Table 31.

**Table 30. DEFINITION OF I$^2$C–BUS TERMINOLOGY**

| Term | Description |
|---|---|
| Transmitter | The device which sends data on the bus |
| Receiver | The device which receives data from the bus |
| Master | The device which initiates a transfer, generates clock signals and terminates a transfer |
| Slave | The devices addressed by a master |
| Synchronization | Procedure to synchronizer the clock signals of two or more devices |

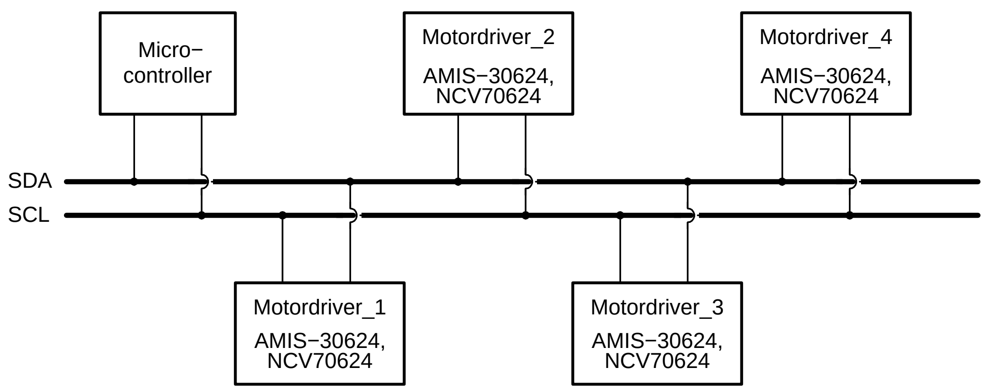



**Figure 21. Example of an I$^2$C−bus Configuration Using One Microcontroller and Four Slaves**

Figure 21 highlights the masterĭslave and receiverĭtransmitter relationships to be found on the I$^2$Cĭbus. It should be noted that these relationships are not permanent but only depend on the direction of data transfer at that time. The transfer of data would proceed as follows:

1. Suppose the microcontroller wants to send information to motordriver_1:
   - Microcontroller (master) addresses motordriver_1 (slave)
   - Microcontroller (masterĭtransmitter) sends data to motordriver_1 (slaveĭreceiver)
   - Microcontroller terminates the transfer
2. If the microcontroller wants to receive information from motordriver_2:
   - Microcontroller (master) addresses motordriver_2 (slave)
   - Microcontroller (masterĭreceiver) receives data from motordriver_2 (slaveĭtransmitter)
   - Microcontroller terminates the transfer

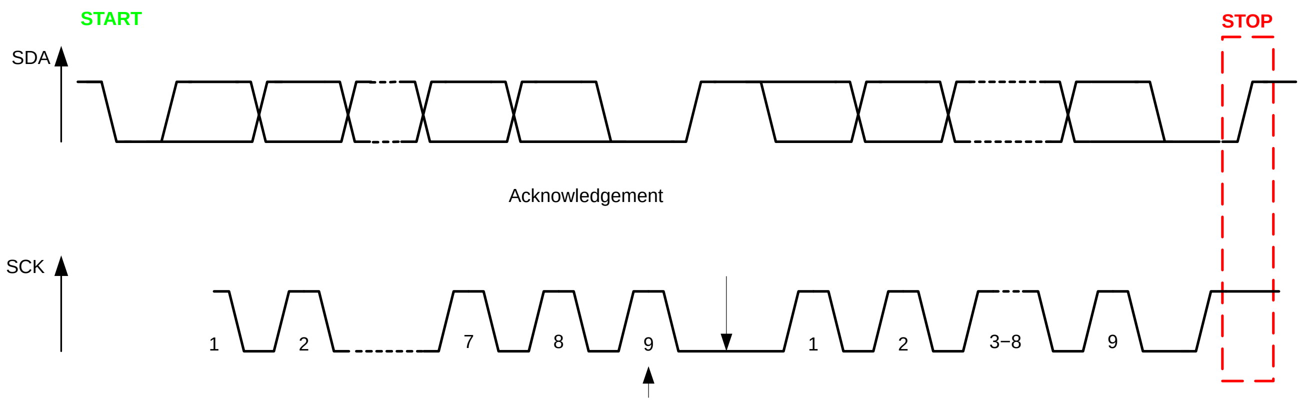
Even in this case the master generates the timing and terminates the transfer.

Generation of the signals on the I$^2$Cĭbus is always the responsibility of the master device. It generates its own clock signal when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow slave device holdingĭdown the clock line.

## Transferring Data

### Byte Format

Every byte put on the SDA line must be 8ĭbits long. The number of bytes that can be transmitted per transfer to AMISĭ30624/NCV70624 is restricted to eight. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first (See Figure 25). If a slave can’t receive or transmit another complete byte of data, it can hold the clock line SCK LOW to force the master into a wait state. Data transfer then continues when the slave is ready for another byte of data and releases clock line SCK.



### Clock Generation

The master generates the clock on the SCK line to transfer messages on the I$^2$C bus. Data is only valid during the HIGH period of the clock.

### Data Formats with 7−bit Addresses

Data transfers follow the format shown in Figure 27. After the START condition (S), a slave address is sent. This address is 7 bit long followed by an eighth bit which is a data

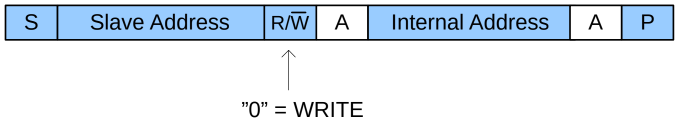



**Figure 29. Master Reading Data from AMIS−30624/NCV70624: First Transmission is Addressing**

2. The second transmission consists of the slave address and the read bit. Then the master can read the data bits on the SDA line on every rising edge of signal SCK. After each byte of data the master has to acknowledge correct data reception by pulling SDA LOW. The last byte is not acknowledged by the master and therefore the slave knows the end of transmission.

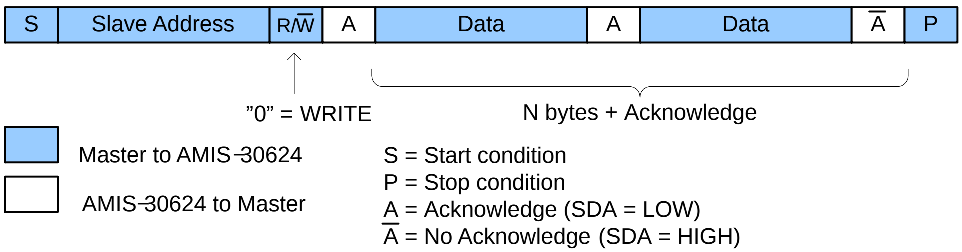



**Figure 30. Master Reading Data from AMIS−30624/NCV70624: Second Transmission is Reading Data**

Notes:
1. Each byte is followed by an acknowledgment bit as indicated by the A or $\overline{A}$ in the sequence.
2. $I^2C$−bus compatible devices must reset their bus logic on receipt of a START condition such that they all anticipate the sending of a slave address, even if these START conditions are not positioned according to the proper format.
3. A START condition immediately followed by a STOP condition (void message) is an illegal format.

## 7−bit Addressing

The addressing procedure for the $I^2C$ĭbus is such that the first byte after the START condition usually determines which slave will be selected by the master. The exception is the general call address which can call all devices. When this address is used all devices should respond with an acknowledge. The second byte of the general call address then defines the action to be taken.

### Definition of Bits in the First Byte

The first seven bits of the first byte make up the slave address. The eighth bit is the least significant bit (LSB). It determines the direction of the message. If the LSB is a "zero" it means that the master will write information to a selected slave. A "one" in this position means that the master will read information from the slave. When an address is sent, each device in a system compares the first seven bits after the START condition with its address. If they match, the device considers itself addressed by the master as a slaveĭreceiver or slaveĭtransmitter, depending on the R/$\overline{W}$ bit.

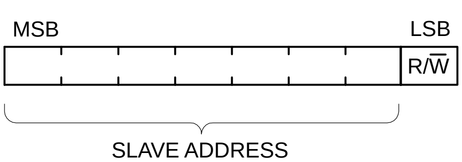



**Figure 31. First Byte after START Procedure**

AMISĭ30624/NCV70624 is provided with a physical address in order to discriminate this circuit from other circuits on the $I^2C$ bus. This address is coded on seven bits (two bits being internally hardwired to '1'), yielding the theoretical possibility of 32 different circuits on the same bus. It is a combination of four OTP memory bits (OTP Memory Structure OPEN) and of the externally hardwired address bits (pin HW). HW must either be connected to ground or to $V_{bat}$. When HW is not connected and is left floating, correct functionality of the positioner is not guaranteed. The motor will be driven to the programmed secure position (See Hardwired Address – OPEN).

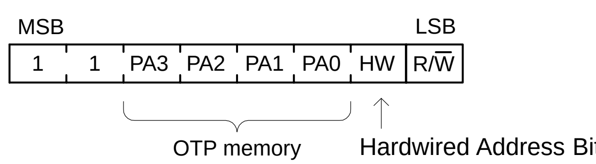



**Figure 32. First Byte after START Procedure**

### General Call Address

The AMISĭ30624/NCV70624 supports also a "general call" address "000 0000", which can address all devices. When this address is used all devices should respond with an acknowledge. The second byte of the general call address then defines the action to be taken.

## I²C APPLICATION COMMANDS

### Introduction

Communications between the AMISĭ30624/NCV70624 and a 2ĭwire serial bus interface master takes place via a large set of commands.

Reading commands are used to:

- Get actual status information, e.g. error flags
- Get actual position of the stepper motor
- Verify the right programming and configuration of the AMISĭ30624/NCV70624.

Writing commands are used to:

- Program the OTP memory
- Configure the positioner with motion parameters (max/min speed, acceleration, stepping mode, etc.)
- Provide target positions to the Stepper motor

The I²Cĭbus master will have to use commands to manage the different application tasks the AMISĭ30624/NCV70624 can feature. The commands summary is given in Table 31.

### Commands Table

**Table 31. I²C COMMANDS WITH CORRESPONDING ROM POINTER**

| Command Mnemonic | Function | Command Byte | |
|---|---|---|---|
| | | Binary | Hexadecimal |
| GetFullStatus1 | Returns complete status of the chip | “1000 0001” | 0x81 |
| GetFullStatus2 | Returns actual, target and secure position | “1111 1100” | 0xFC |
| GetOTPParam | Returns OTP parameter | “1000 0010” | 0x82 |
| GotoSecurePosition | Drives motor to secure position | “1000 0100” | 0x84 |
| HardStop | Immediate full stop | “1000 0101” | 0x85 |
| ResetPosition | Sets actual position to zero | | |

## Application Commands

### GetFullStatus1

This command is provided to the circuit by the master to get a complete status of the circuit and of the stepper motor. Refer to Tables 19 and 20 to see the meaning of the parameters sent back to the $I^2C$ master.

**Note**: A `GetFullStatus1` command will attempt to reset

### GetFullStatus2

This command is provided to the circuit by the master to

### SetDualPosition

This command is provided to the circuit by the $I^2C$ master

### SetMotorParam

This command is provided to the circuit by the I$^2$C master to set the values for the stepper motor parameters (listed below) in RAM. Refer to Table 19 to see the meaning of the parameters sent by the I$^2$C master.

**Important**: If a `SetMotorParam` occurs while a motion is ongoing, it will modify at once the motion parameters (see Position Controller corresponds to the following I$^2$C command frame:). Therefore the application should not change parameters other than `Vmax` while a motion is running, otherwise correct positioning cannot be guaranteed.

SetMotorParam

**Table 45. SetMotorParam COMMAND FRAME**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Address | **1** | **1** | **OTP3** | **OTP2** | **OTP1** | **OTP0** | **HW** | **0** |
| 1 | Command | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** |
| 2 | Data 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | Data 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | Data 3 | Irun[3:0] | | | | Ihold[3:0] | | | |
| 5 | Data 4 | Vmax[3:0] | | | | Vmin[3:0] | | | |
| 6 | Data 5 | SecPos[10:8] | | | Shaft | Acc[3:0] | | | |

### SetPosition

This command is provided to the circuit by the I$^2$C master to drive the motor to a given absolute position. See Positioning (see Priority Encoder) for more details. The priority encoder table acknowledges the cases where a `SetPosition` command will be ignored.

SetPosition corresponds to the following I$^2$C command frame:

**Table 47. SetPosition COMMAND FRAME**

| Byte | Content | Structure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Address | 1 | 1 | OTP3 | OTP2 | OTP1 | OTP0 | HW | 0 |
| 1 | Command | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 2 | Data 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | Data 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | Data 3 | Pos[15:8] | | | | | | | |
| 5 | Data 4 | Pos[7:0] | | | | | | | |

Where:

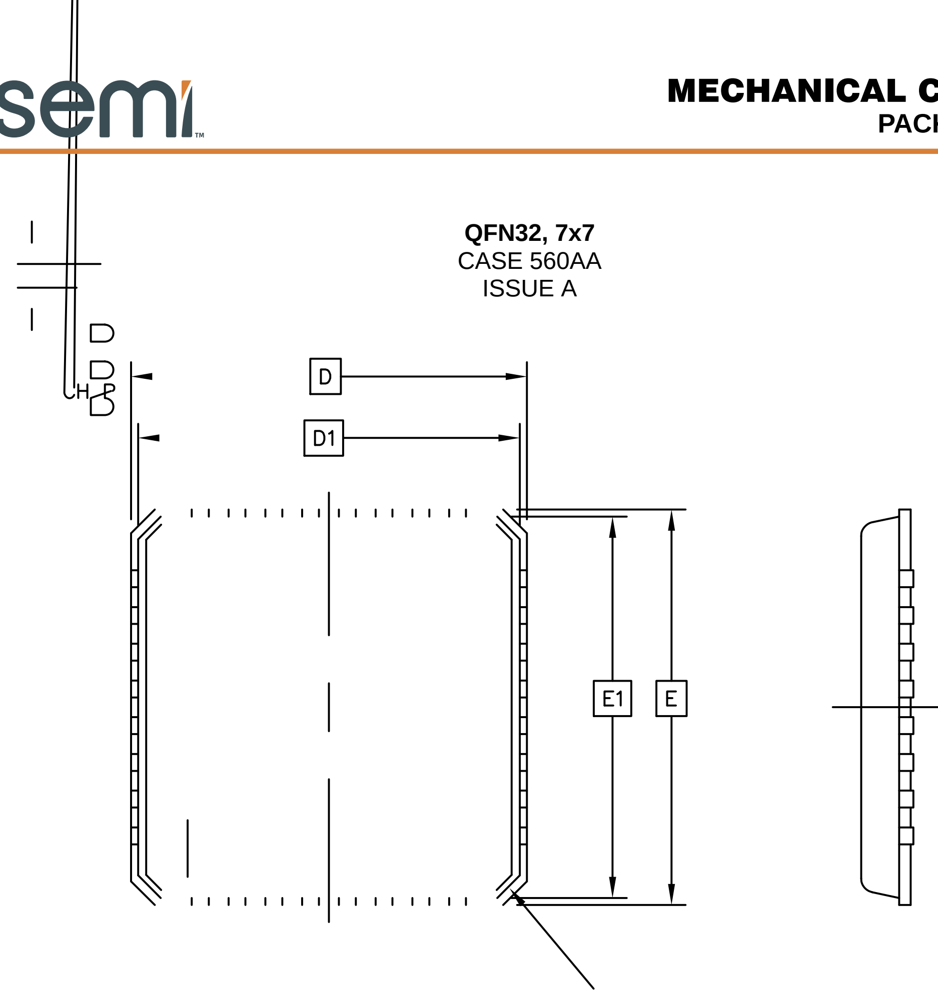
Pos [15:0] Signed 16−bit position set−point for motor.

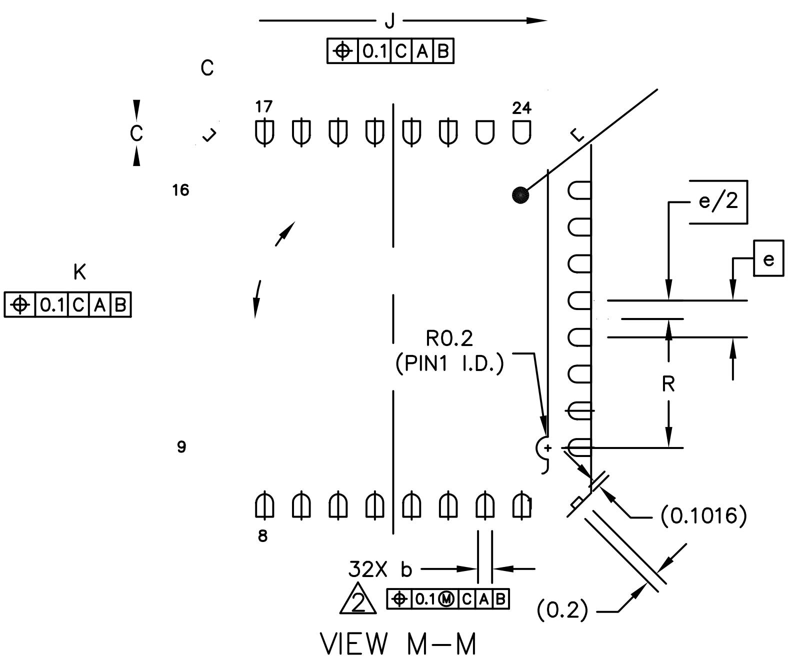
### SoftStop

This command will be internally triggered when the chip temperature rises above the thermal shutdown threshold (see

**QFN32, 7x7**
CASE 560AA
ISSUE A

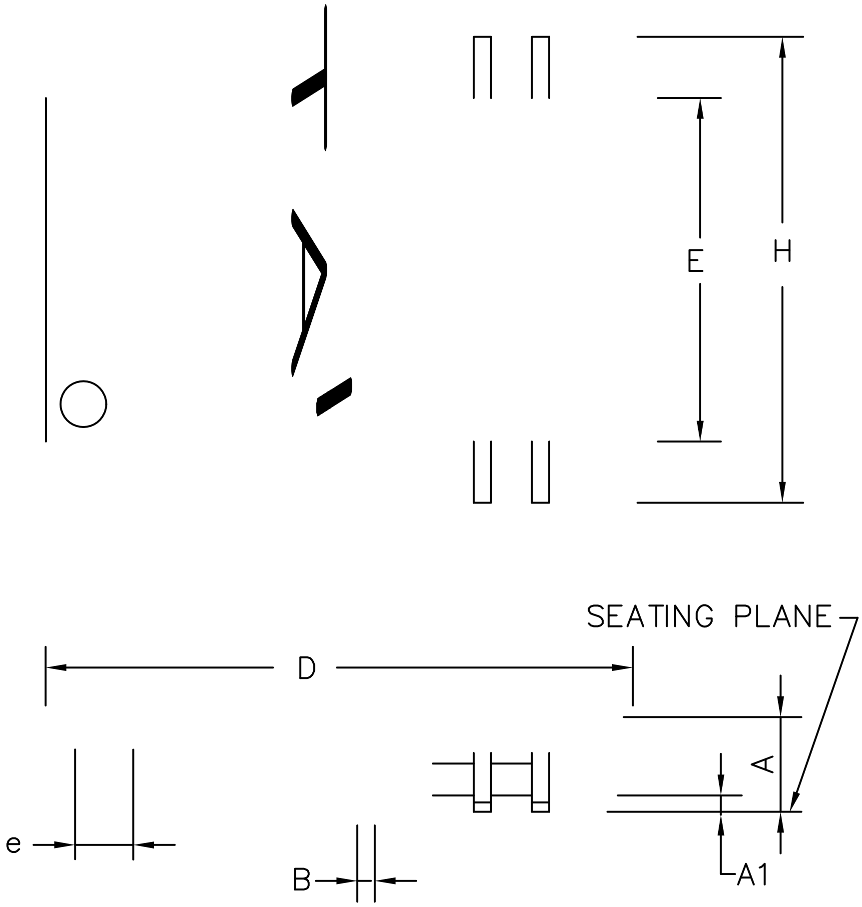
DATE 23 SEP 2015




| DOCUMENT NUMBER: | 98AON30885E | Electronic versions are uncontrolled except when |
|---|---|---|
| DESCRIPTION: | | |

SOIC 20 W




| DIMENSIONS IN INCHES | | | |
| --- | --- | --- | --- |
| SYMBOL | MIN. | NOM. | MAX. |
| A | 0.093 | 0.099 | 0.104 |
| A1 | 0.004 | 0.008 | 0.012 |
| A2 | 0.088 | 0.094 | 0.100 |
| B | 0.013 | 0.016 | 0.020 |
| C | 0.0090 | 0.0100 | 0.0125 |
| D | 0.496 | 0.503 | 0.510 |
| | | | |
| e | .050 B | | |
| H | 0. | | |
| h | 0.010 | 0.015 | 0.019 |
| L | 0.016 | 0.033 | 0.050 |
| α | 0° | 5° | 8° |