AMIS-30621 LIN Micro-stepping Motor Driver

Data Sheet

1.0 General Description

The AMIS-30621 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 AMIS-30621 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.

The chip is implemented in I^2T100 technology, enabling both high voltage analog circuitry and digital functionality on the same chip. The AMIS-30621 is fully compatible with the automotive voltage requirements.

2.0 Product Features

Motor Driver
- Micro-stepping technology
- Sensorless step-loss detection
- Peak current up to 800mA
- Fixed frequency PWM current-control
- Automatic selection of fast and slow decay mode
- No external fly-back diodes required
- 14V/24V compliant
- Motion qualification mode

Controller with RAM and OTP Memory
- Position controller
- Configurable speeds and acceleration
- Input to connect optional motion switch

LIN Interface
- Both physical and data-link layers (conform to LIN rev. 1.3)
- Field programmable node addresses
- Dynamically allocated identifiers
- Full diagnostics and status information

Protection
- Over-current protection
- Under-voltage management
- Open circuit detection
- High-temp warning and management
- Low-temp flag
- LIN bus short-circuit protection to supply and ground
- Lost LIN safe operation

Power Saving
- Power-down supply current < 100µA
- 5V regulator with wake-up on LIN activity

EMI Compatibility
- LIN bus integrated slope control
- HV outputs with slope control

3.0 Applications

The AMIS-30621 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.

4.0 Ordering Information

Table 1: Ordering Information

Part No.	Package	UV (*)	Peak Current	Temp. Range	Ordering Code Tubes	Ordering Code Tapes
AMIS-30621-AGA	SOIC-20	High	800 mA	-40°C…..125°C	0C621-003-XTD	0C621-003-XTP
AMIS-30621-UGA	SOIC-20	Low	800 mA	-40°C…..125°C	0C621-007-XTD	0C621-007-XTP
AMIS-30621-ANA	NQFP-32 (7 x 7mm)	High	800 mA	-40°C…..125°C	0C621-004-XTD	0C621-004-XTP
AMIS-30621-UNA	NQFP-32 (7 x 7mm)	Low	800 mA	-40°C…..125°C	0C621-006-XTD	0C621-006-XTP

(*) UV under-voltage lock out levels: see DC Parameters UV1 & UV2 (Stop Voltage thresholds)

5.0 Quick Reference Data

Table 2: Absolute Maximum Ratings

Parameter		Min.	Max.	Unit
Vbb	Supply voltage	-0.3	+40 (1)	V
Vlin	Bus input voltage	-80	+80	V
Tamb	Ambient temperature under bias (2)	-50	+150	°C
Tst	Storage temperature	-55	+160	°C
Vesd (3)	Electrostatic discharge voltage on LIN pin	-4	+4	kV
	Electrostatic discharge voltage on other pins	-2	+2	kV

Notes:
(1) For limited time <0.5s.
(2) The circuit functionality is not guaranteed.
(3) Human body model (100 pF via 1.5 kΩ, according to JEDEC std.).

Table 3: Operating Ranges

Parameter			Min.	Max.	Unit
Vbb	Supply voltage		+8	+29	V
Top	Operating temperature range	Vbb ≤ 18V	-40	+125	°C
		Vbb ≤ 29V	-40	+85	°C

6.0 Block Diagram

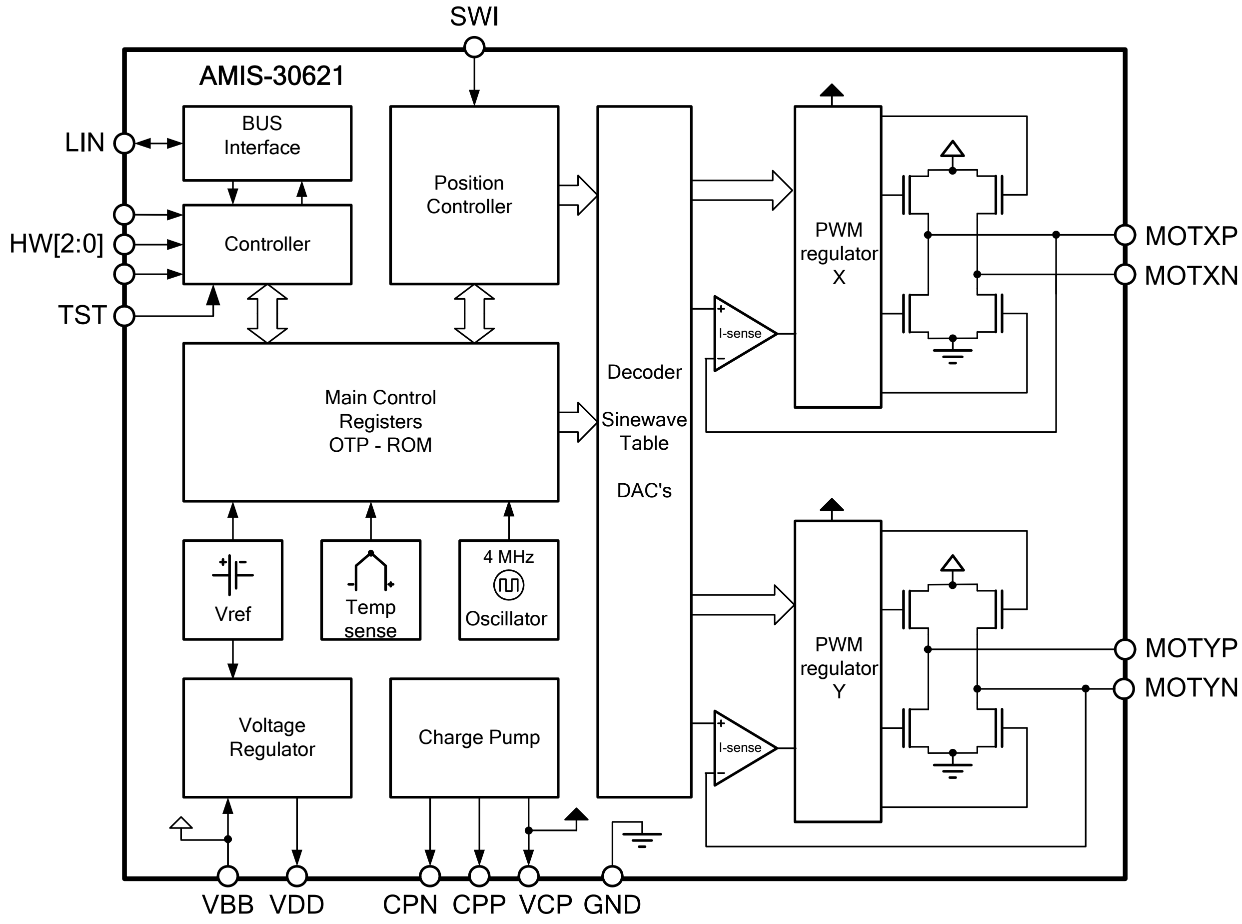


Figure 1: Block Diagram

7.0 Pin Out

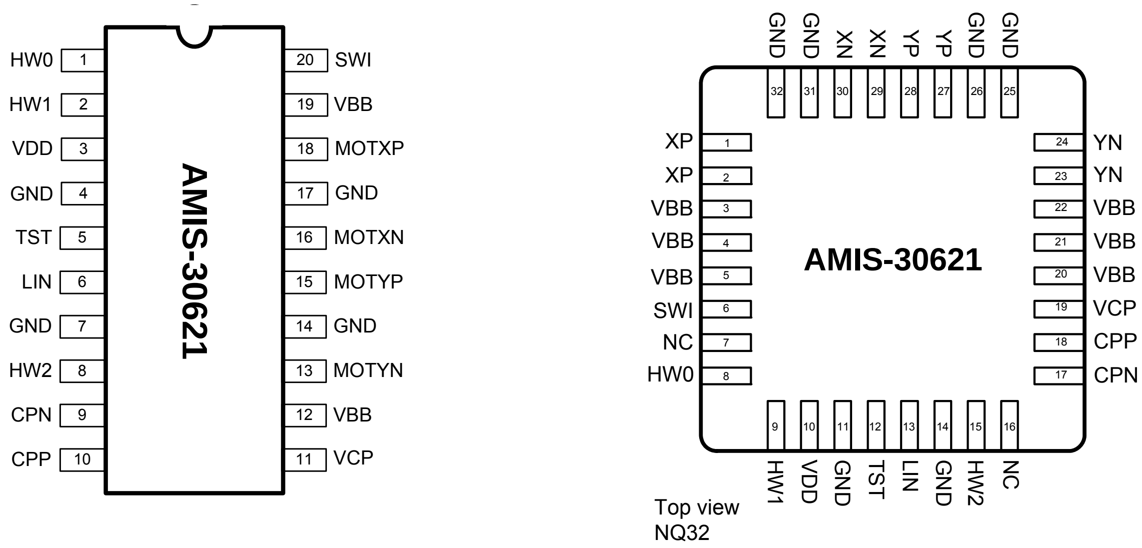


Figure 2: SOIC 20 and NQFP-32 Pin Out

Table 4: Pin Description

Pin Name	Pin Description		SOIC-20	NQFP-32
HWO	Bit 0 of LIN-ADD	To be tied to GND or VDD	1	8
HW1	Bit 1 of LIN-ADD		2	9
VDD	Internal supply (needs external decoupling capacitor)		3	10
GND	Ground, heat sink		4,7,14,17	11, 14, 25, 26, 31, 32
TST	Test pin (to be tied to ground in normal operation)		5	12
LIN	LIN-bus connection		6	13
HW2	Bit 2 LIN-ADD		8	15
CPN	Negative connection of pump capacitor (charge pump)		9	17
CPP	Positive connection of pump capacitor (charge pump)		10	18
VCP	Charge-pump filter-capacitor		11	19
VBB	Battery voltage supply		12,19	3, 4, 5, 20, 21, 22
MOTYN	Negative end of phase Y coil		13	23, 24
MOTYP	Positive end of phase Y coil		15	27, 28
MOTXN	Negative end of phase X coil		16	29, 30
MOTXP	Positive end of phase X coil		18	1, 2
SWI	Switch input		20	6
NC	Not connected (can be either soldered to a signal or left floating)			7, 16

8.0 Package Thermal Resistance

8.1 SOIC-20

To lower the junction-to-ambient thermal resistance, it is recommended to connect the ground leads to a PCB ground plane layout as illustrated in Figure 3. The junction-to-case thermal resistance is depending on the copper area, copper thickness, PCB thickness, and number of copper layers. Calculating with a total area of 460 mm^2, 35µm copper thickness, 1.6mm PCB thickness and 1layer, the thermal resistance is 28°C/W, leading to a junction-ambient thermal resistance of 63°C/W.

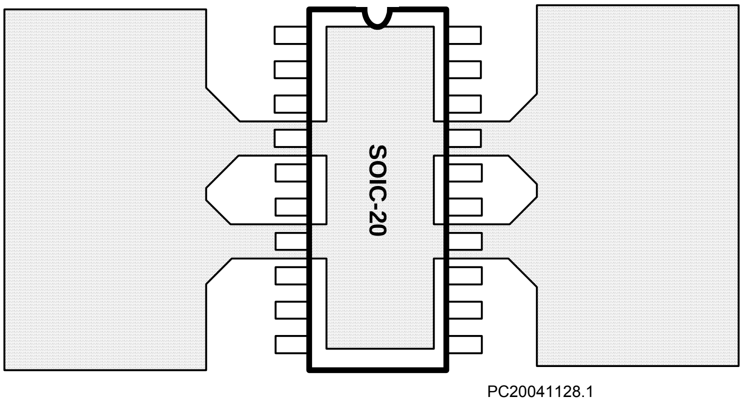


Figure 3: PCB Ground Plane Layout Condition

8.2 NQFP-32

The NQFP is designed to provide superior thermal performance. Using an exposed die pad on the bottom surface of the package is partly contributing to this. In order to take full advantage of this, the PCB must have features to conduct heat away from the package. A thermal grounded pad with thermal vias can achieve this. With a layout as shown in Figure 4 the thermal resistance junction – to – ambient can be brought down to a level of 25°C/W.

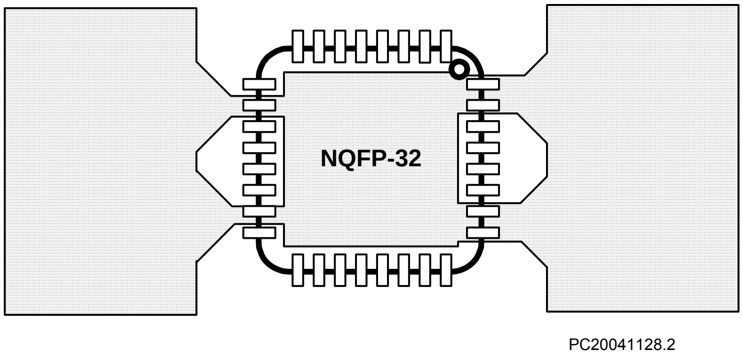


Figure 4: PCB Ground Plane Layout Condition

9.0 DC Parameters

The DC parameters are given for Vbb and temperature in their operating ranges. Convention: currents flowing in the circuit are defined as positive.

Table 5: DC Parameters

Symbol	Pin(s)	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Motor Driver							
$I_{MSmax,Peak}$	MOTXP MOTXN MOTYP MOTYN	Max current through motor coil in normal operation			800		mA
$I_{MSmax,RMS}$		Max RMS current through coil in normal operation			570		mA
I_{MSabs}		Absolute error on coil current		-10		10	%
I_{MSrel}		Error on current ratio I_{coilx} / I_{coily}		-7		7	%
R_{DSon}		On resistance for each motor pin (including bond wire) at I_{MSmax}	V_{bb} = 12V, T_j = 50°C		0.50	1	Ω
			V_{bb} = 8V, T_j = 50°C		0.55	1	Ω
			V_{bb} = 12V, T_j = 150°C		0.70	1	Ω
			V_{bb} = 8V, T_j = 150°C		0.85	1	Ω
I_{MSL}		Pull down current	HiZ mode		2		mA
LIN Transmitter							
I_{bus_on}	LIN	Dominant state, driver on	Vbus = 1.4V	40			mA
I_{bus_off}		Dominant state, driver off	Vbus = 0V	-1			mA
I_{bus_off}		Recessive state, driver off	Vbus = Vbat			20	µA
I_{bus_lim}		Current limitation		50		200	mA
R_{slave}		Pull-up resistance		20	30	47	kΩ
LIN Receiver							
V_{bus_dom}	LIN	Receiver dominant state		0		0.4 *Vbb	V
V_{bus_rec}		Receiver recessive state		0.6 * Vbb		Vbb	V
V_{bus_hys}		Receiver hysteresis		0.05 * Vbb		0.2 * Vbb	V
Thermal Warning and Shutdown							
T_{tw}		Thermal warning		138	145	152	°C
T_{tsd} (1) (2)		Thermal shutdown			Ttw + 10		°C
T_{low} (2)		Low temperature warning			Ttw - 155		°C
Supply and Voltage Regulator							
V_{bb}	VBB	Nominal operating supply range		6.5		18	V
V_{bb}		Nominal operating supply range	$T_{amb} \leq 85$°C	6.5		29	
UV_1		Stop voltage high threshold	Product versions with low UV; see Ordering Information	7.8	8.4	8.9	V
UV_2		Stop voltage low threshold		7.1	7.5	8.0	V
UV_1		Stop voltage high threshold	Product versions with high UV; see Ordering Information	8.8	9.4	9.9	V
UV_2		Stop voltage low threshold		8.0	8.5	9.0	V
V_{bbOTP}		Supply voltage for OTP zapping (3)		9.0		10.0	V
I_{bat}		Total current consumption	Unloaded outputs		3.50	10.0	mA
I_{bat_s}		Sleep mode current consumption			50	100	µA
V_{dd}	VDD	Internal regulated output (4)	8V < V_{bb} < 18V	4.75	5	5.50	V
I_{ddStop}		Digital current consumption	V_{bb} < UV_2		2		mA
$V_{ddReset}$		Digital supply reset level @ power down (5)				4.5	V
I_{ddLim}		Current limitation	Pin shorted to ground			42	mA
Switch Input and Hardwire Address Input							
Rt_{OFF}	SWI HW2	Switch OFF resistance (6)	Switch to Gnd or Vbat,	10			kΩ

Symbol	Pin(s)	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$Rt\_{}_{ON}$		Switch ON resistance (6)				2	kΩ
V_{bb_sw}		Vbb range for guaranteed operation of SWI and HW2		6		29	V
V_{max_sw}		Maximum voltage	T < 1s			40V	V
I_{lim_sw}		Current limitation	Short to Gnd or Vbat		30		mA
Hardwired Address Inputs and Test Pin							
V_{low}	HW0 HW1 TST	Input level high		0.7 * Vdd		.	V
V_{high}		Input level low				0.3 * Vdd	V
HW_{hyst}		Hysteresis		0.075 * Vdd			V
Charge Pump							
V_{cp}	VCP	Output voltage	Vbb > 15V	Vbb+10	Vbb+12.5	Vbb+15	V
			8V < Vbb < 15V	2 * Vbb – 5	2 * Vbb – 2.5	2 * Vbb	V
C_{buffer}		External buffer capacitor		220		470	nF
C_{pump}	CPP CPN	External pump capacitor		220		470	nF

Notes:

(1) No more than 100 cumulated hours in life time above T_{tsd}.
(2) Thermal shutdown and low temperature warning are derived from thermal warning.
(3) A 10µF buffer capacitor of between VBB and GND is minimum needed. Short connections to the power supply are recommended.
(4) Pin VDD must not be used for any external supply.
(5) The RAM content will not be altered above this voltage.
(6) External resistance value seen from pin SWI or HW2, including 1 kΩ series resistor.

10.0 AC Parameters

The AC parameters are given for Vbb and temperature in their operating ranges.

The LIN transmitter/receiver parameters conform to LIN Protocol Specification Revision 1.3. Unless otherwise specified 8V < V_{bb} < 18V, Load for propagation delay = 1kΩ, Load for slope definitions: [L1] = 1nF / 1kΩ; [L2] = 6.8nF / 660Ω; [L3] = 10nF / 510Ω.

Table 6: AC Parameters

Symbol	Pin(s)	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
Power-up							
T_{pu}		Power-up time	Guaranteed by design			10	ms
Internal Oscillator							
f_{osc}		Frequency of internal oscillator		3.6	4.0	4.4	MHz
LIN Transmitter							
T_slope_F/R	LIN	Slope time falling or rising edge	Extrapolated between 40% and 60% Vbus_dom	3.5		22.5	µs
T_slope_Sym		Slope time symmetry (1)	T_slope_F – T_slope_R	-4		4	µs
T_tr_F		Propagation delay TxD low to bus		0.1	1	4	µs
T_tr_R		Propagation delay TxD high to bus		0.1	1	4	µs
Tsym_tr		Transmitter delay symmetry	T_tr_F – T_tr_R	-2		2	µs
LIN Receiver							
T_rec_F	LIN	Propagation delay bus dominant to RxD low		0.1	4	6	µs
T_rec_R		Propagation delay bus recessive to RxD high		0.1	4	6	µs
Tsym_rec		Receiver delay symmetry	T_rec_F – T_rec_R	-2		2	µs
T_{wake}		Wake-up delay time		50	100	200	µs
Switch Input and Hardwire Address Input							
T_{sw}	SWI HW2	Scan pulse period (2)			1024		µs
T_{sw_on}		Scan pulse duration (2)			64		µs
Motor Driver							
F_{pwm}	MOTxx	PWM frequency (2)	PWMfreq = 0 (3)	20.6	22.8	25.0	kHz
			PWMfreq = 1 (3)	41.2	45.6	50.0	kHz
F_{jit_depth}		PWM jitter modulation depth	PWMJen = 1 (3)		10		%
T_{brise}		Turn-on transient time	Between 10% and 90%		170		ns
T_{bfall}		Turn-off transient time			140		ns
Charge Pump							
f_{CP}	CPN CPP	Charge pump frequency (2)			250		kHz

Notes:
(1) For loads [L1] and [L2].
(2) Derived from the internal oscillator.
(3) See SetMotorParam and PWM regulator .

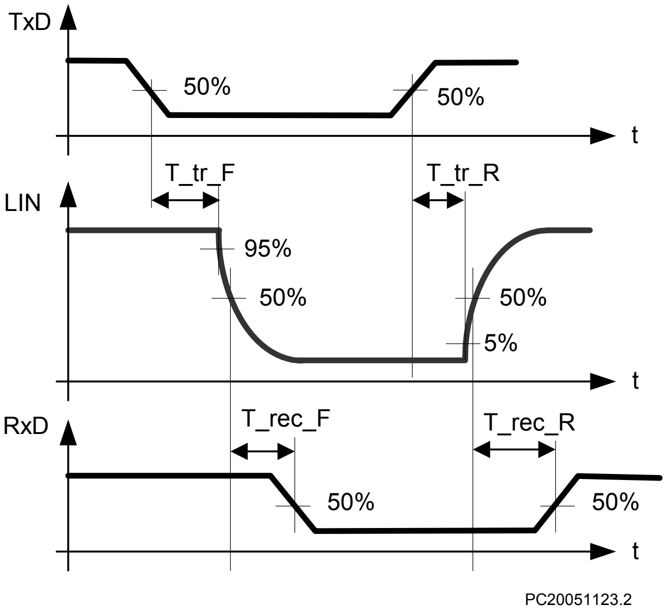


Figure 5: LIN Delay Measurement

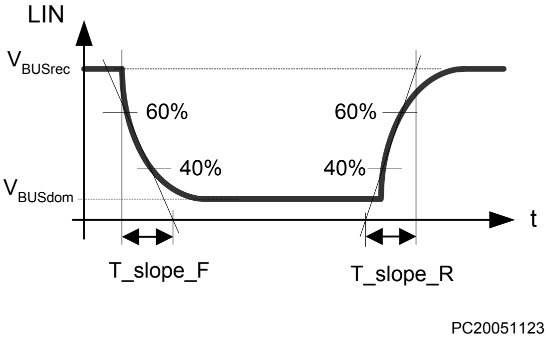


Figure 6: LIN Slope Measurement

11.0 Typical Application

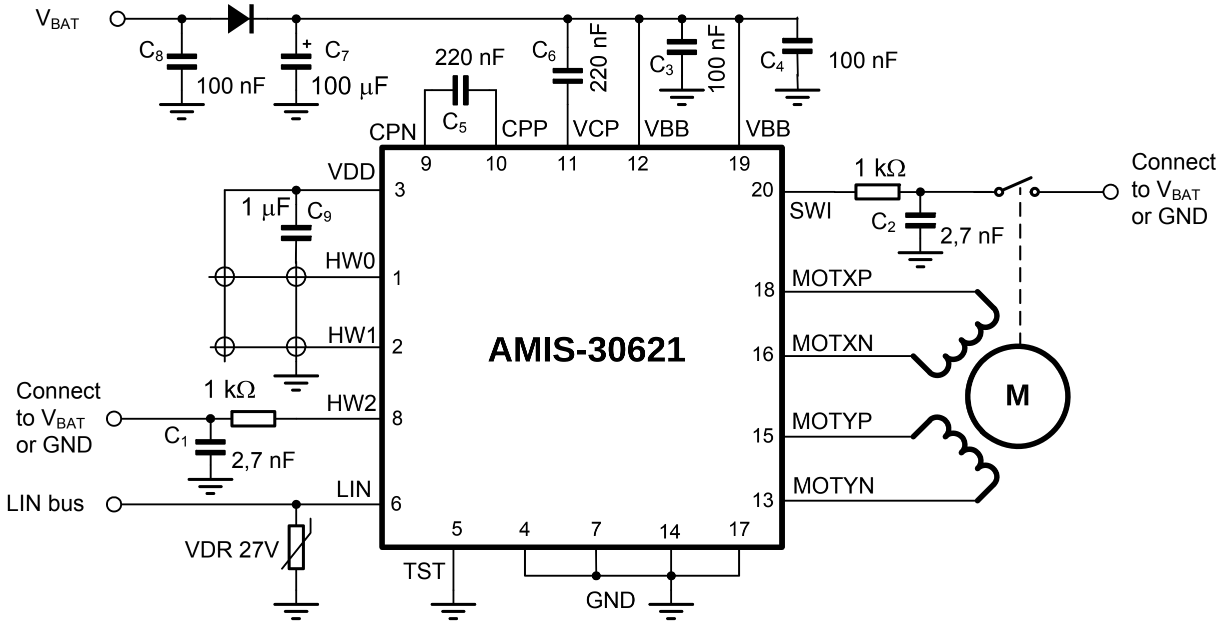


Figure 7: Typical Application Diagram

Notes:

(1) All resistors are ± 5%, ¼ W.
(2) C_1, C_2 minimum value is 2.7nF, maximum value is 10nF.
(3) Depending on the application, the ESR value and working voltage of C_7 must be carefully chosen.
(4) C_3 and C_4 must be close to pins VBB and GND.
(5) C_5 and C_6 must be as close as possible to pins CPN, CPP, VCP, and VBB to reduce EMC radiation.
(6) C_9 must be a ceramic capacitor to assure low ESR.

12.0 Positioning Parameters

12.1 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

12.2 Maximum Velocity

For each stepping mode, the maximum velocity Vmax can be programmed to 16 possible values given in Table 7.

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 7: 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

12.3 Minimum Velocity

Once the maximum velocity is chosen, 16 possible values can be programmed for the minimum velocity Vmin. Table 8 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

Notes:
(1) The Vmax factor is an approximation.
(2) 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**.

12.4 Acceleration and Deceleration

Sixteen possible values can be programmed for Acc (acceleration and deceleration between Vmin and Vmax). Table 9 provides the obtainable values in full-step/s². One observes restrictions for some combination of acceleration index and maximum speed (gray cells).

The accuracy of Acc is derived from the internal oscillator.

Table 9: Acceleration and Deceleration Selection Table

<table>
<tr><th colspan="2">Vmax (FS/s) →</th><th>99</th><th>136</th><th>167</th><th>197</th><th>213</th><th>228</th><th>243</th><th>273</th><th>303</th><th>334</th><th>364</th><th>395</th><th>456</th><th>546</th><th>729</th><th>973</th></tr>
<tr><th colspan="2">↓ Acc index</th><th colspan="16" rowspan="2">Acceleration (Full-step/s²)</th></tr>
<tr><th>Hex</th><th>Dec</th></tr>
<tr><td>0</td><td>0</td><td colspan="7">49</td><td colspan="7">106</td><td colspan="2">473</td></tr>
<tr><td>1</td><td>1</td><td colspan="14">218</td><td colspan="2">735</td></tr>
<tr><td>2</td><td>2</td><td colspan="16">1004</td></tr>
<tr><td>3</td><td>3</td><td colspan="16">3609</td></tr>
<tr><td>4</td><td>4</td><td colspan="16">6228</td></tr>
<tr><td>5</td><td>5</td><td colspan="16">8848</td></tr>
<tr><td>6</td><td>6</td><td colspan="16">11409</td></tr>
<tr><td>7</td><td>7</td><td colspan="16">13970</td></tr>
<tr><td>8</td><td>8</td><td colspan="16">16531</td></tr>
<tr><td>9</td><td>9</td><td rowspan="7">14785</td><td colspan="15">19092</td></tr>
<tr><td>A</td><td>10</td><td colspan="15">21886</td></tr>
<tr><td>B</td><td>11</td><td colspan="15">24447</td></tr>
<tr><td>C</td><td>12</td><td colspan="15">27008</td></tr>
<tr><td>D</td><td>13</td><td colspan="15">29570</td></tr>
<tr><td>E</td><td>14</td><td colspan="6" rowspan="2">29570</td><td colspan="9">34925</td></tr>
<tr><td>F</td><td>15</td><td colspan="9">40047</td></tr>
</table>

The formula to compute the number of equivalent full-step during acceleration phase is:

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

12.5 Positioning

The position programmed in commands `SetPosition` and `SetPositionShort` is given as a number of (micro) steps. According to the chosen stepping mode, the position words must be aligned as described in Table 10. When using command `SetPositionShort` or `GotoSecurePosition`, data is automatically aligned.

Table 10: Position Word Alignment

Stepping mode	Position word: `Pos[15:0]`																Shift
1/16th	**S**	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	**LSB**	No shift
1/8th	**S**	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	**LSB**	0	1-bit left ⇔ ×2
1/4th	**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
PositionShort	**S**	**S**	**S**	B9	B8	B7	B6	B5	B4	B3	B2	B1	**LSB**	0	0	0	No shift
SecurePosition	**S**	B9	B8	B7	B6	B5	B4	B3	B2	B1	**LSB**	0	0	0	0	0	No shift

Notes:
(1) LSB: Least Significant Bit
(2) S: Sign bit

12.5.1. 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 Table 11.

Table 11: Position Range

Command	Stepping Mode	Position Range	Full Range Excursion	Number of Bits
`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

When using the command `SetPosition`, although coded on 16 bits, the position word will have to be shifted to the left by a certain number of bits, according to the stepping mode.

12.5.2. Secure Position

A secure position can be programmed. It is coded in 11-bits, thus having a lower resolution than normal positions, as shown in Table 12. See also command `GotoSecurePosition` and LIN Lost Behavior.

Table 12: 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:

(1) The secure position is disabled in case the programmed value is the reserved code "10000000000" (0x400 or most negative position).
(2) After start-up the OTP register is copied in RAM as illustrated below.

12.5.3. Shaft

A shaft bit which can be programmed in OTP or with command `SetMotorParam`, defines whether a positive motion is a clockwise or counter-clockwise rotation (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.

13.0 Structural Description

See
Figure 1.

13.1 Stepper Motor Driver

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

- Two H-bridges designed to drive a stepper motor with 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.
- The charge pump to allow driving the H-bridges' high side transistors.
- Two pre-scale 4-bit DACs to set the maximum magnitude of the current through X and Y.
- Two DACs to set the correct current ratio through X and Y.

Battery voltage monitoring is also performed by this block, which provides needed 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.

13.2 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.

13.3 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, etc.) and pass it to the LIN master node.

13.4 Miscellaneous

The AMIS-30621 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 (POR) circuit.
- A 5V regulator (from the battery supply) to supply the internal logic circuitry.

14.0 Functions Description

This section 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 sections.

14.1 Position Controller

14.1.1. Positioning and Motion Control

A positioning command will produce a motion as illustrated in Figure 8. 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 configuration phase. The current in the coils is also programmable.

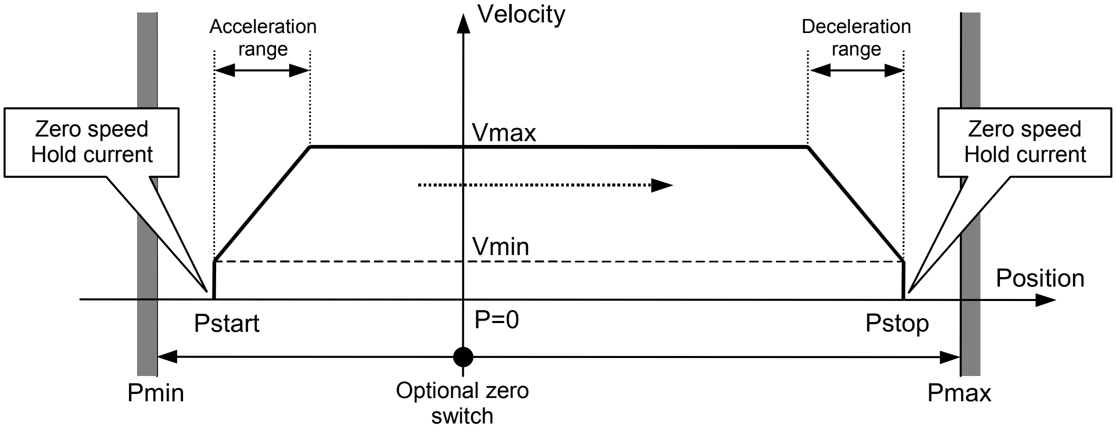


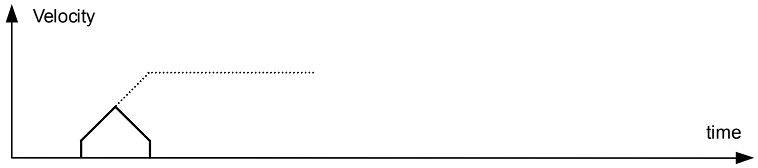
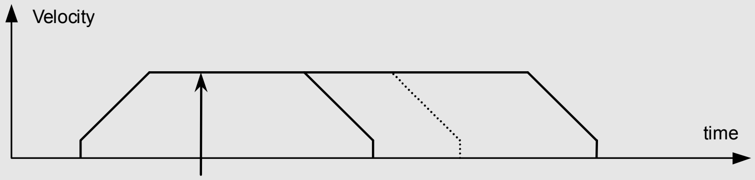
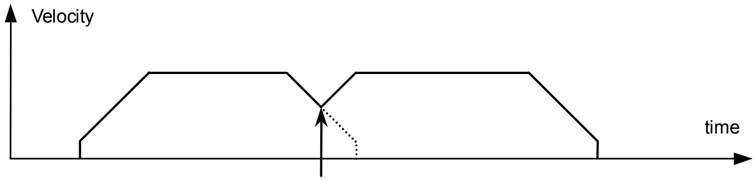
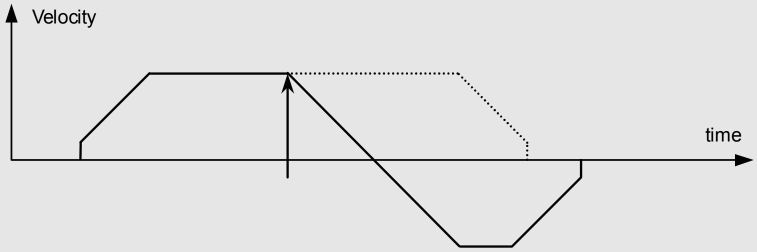
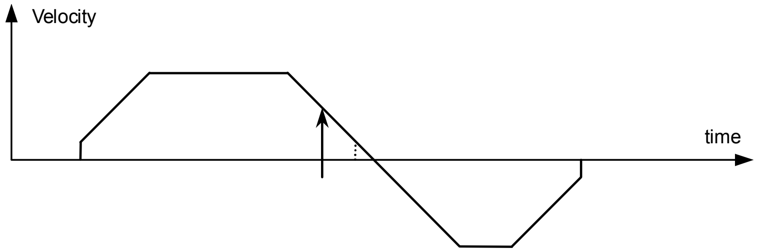
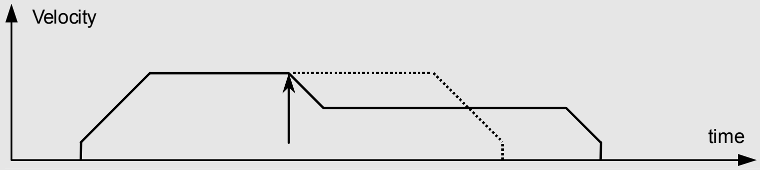
Figure 8: Positioning and Motion Control

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 14.

Table 14: Positioning Examples

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: 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	

14.1.2. 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 motion to a position `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.

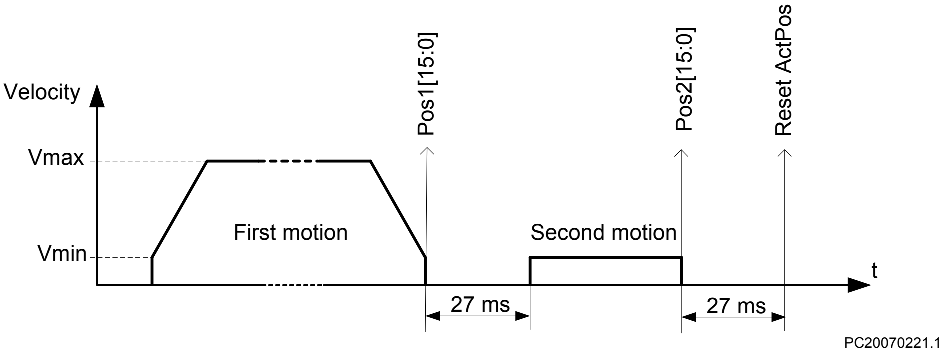


Figure 9: Dual Positioning

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.

Notes:
(0) The **priority encoder** is describing the management of states and commands. All notes below are to be considered illustrative.
(1) The last SetPosition(Short) command issued during an DualPosition sequence will be kept in memory and executed afterwards. This applies also for the commands sleep and SetMotorParam and GotoSecurePosition.
(2) 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.
(3) A DualPosition sequence starts by setting TagPos register to SecPos value, provided secure position is enabled otherwise TagPos is reset to zero.
(4) The acceleration/deceleration value applied during a DualPosition sequence is the one stored in RAM before the `SetDualPosition` command is sent. The same applies for shaft bit, but not for Irun, Ihold and StepMode, which can be changed during the dual positioning sequence.
(5) The Pos1, Pos2, Vmax and Vmin values programmed in a `SetDualPosition` command apply only for this sequence. All further positioning will use the parameters stored in RAM (programmed for instance by a former SetMotorParam command).
(6) Commands ResetPosition, SetDualPosition and SoftStop will be ignored while a DualPosition sequence is ongoing, and will not be executed afterwards.
(7) A SetMotorParam command should not be sent during a `SetDualPosition` sequence.
(8) If for some reason ActPos equals Pos1[15:0] at the moment the `SetDualPosition` command is issued, the circuit will enter in deadlock state. Therefore, the application should check the actual position by a GetPosition or a `GetFullStatus` command prior to send the `SetDualPosition` command.

14.1.3. 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 minimum.

Figure 10 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. One could also use for larger movements the command <`RunVelocity`>.

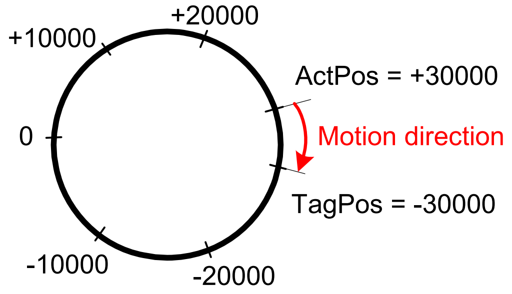


Figure 10: Motion Direction is Function on Difference Between ActPos and TagPos

14.1.4. Hardwired Address HW2

Figure 11 shows a simplified schematic diagram of the HW2 comparator circuit. The HW2 pin is sensed via two 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 VBAT. The corresponding I→ R converter output is low and via S_{PASS_B} fed to the comparator. The output HW2_Cmp will be high.

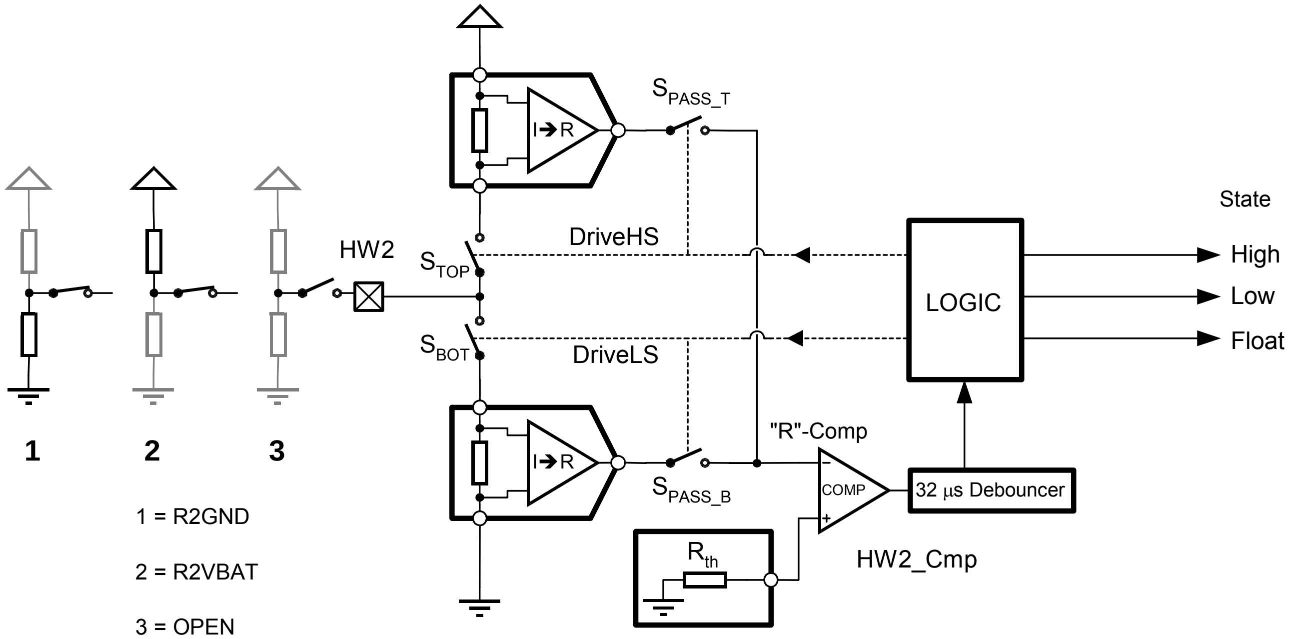


Figure 11: Simplified Schematic Diagram of the HW2 Comparator

Three cases can be distinguished (see Figure 11):

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

Table 15: 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 32µs 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 Table 17 the state is depending on the previous state, the condition of the two switch controls (DriveLS and DriveHS) and the output of HW2_Cmp. Figure 12 is showing an example of a practical case where a connection to VBAT is interrupted.

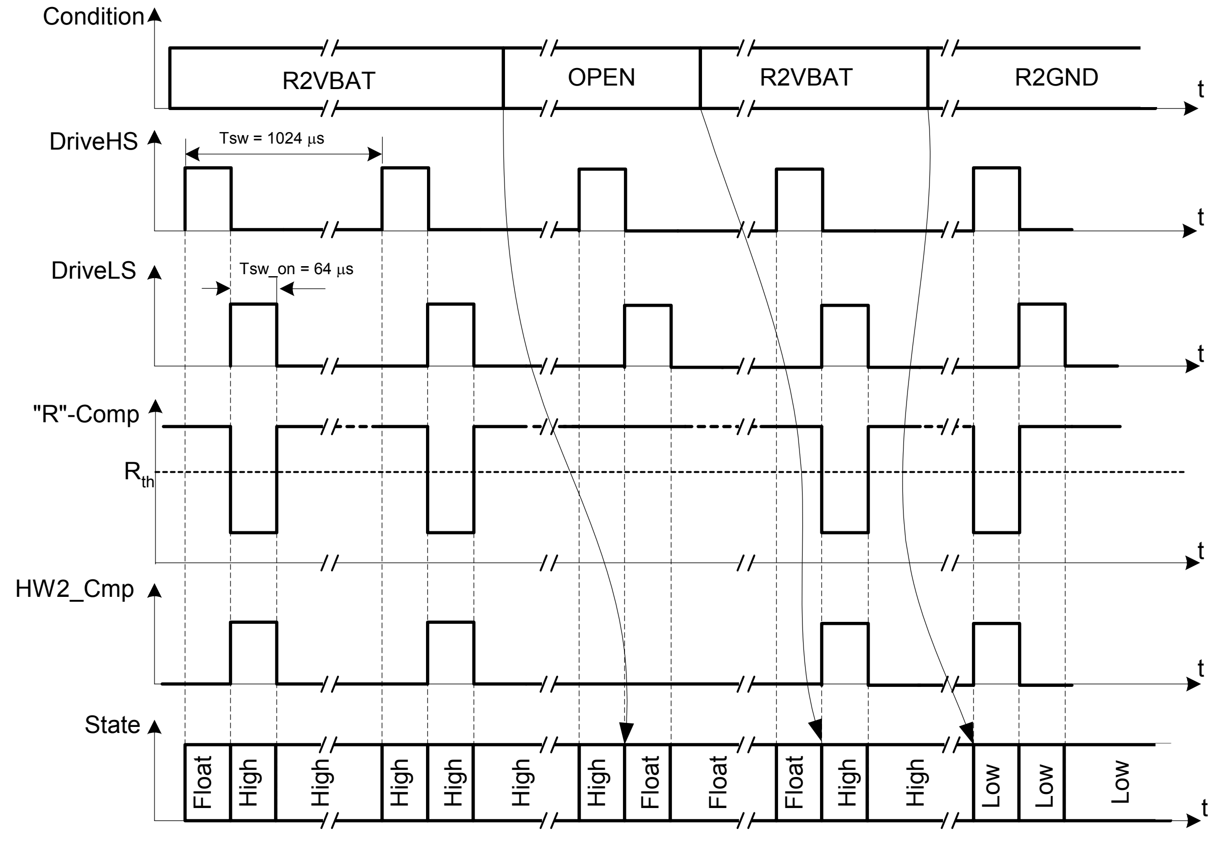


Figure 12: Timing Diagram Showing the Change in States for HW2 Comparator

R2VBAT
A resistor is connected between VBAT and HW2. Every 1024µs S_{BOT} is closed 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 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 on Table 17 one can see that the state changes to float. This will trigger a motion to secure position. See also Electrical Transient Conduction Along Supply Lines.

R2GND
If a resistor is connected between HW2 and the GND, a current is sensed every 1024µs whet 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 1 one can see that the state will change to low. This low state will be interpreted as HW2 address = 0.

14.1.5. External Switch SWI

Figure 13 illustrates that the SWI comparator is almost identical to HW2. The major difference is in the limited number of states. Only open or closed is recognised leading to respectively ESW = 0 and ESW = 1.

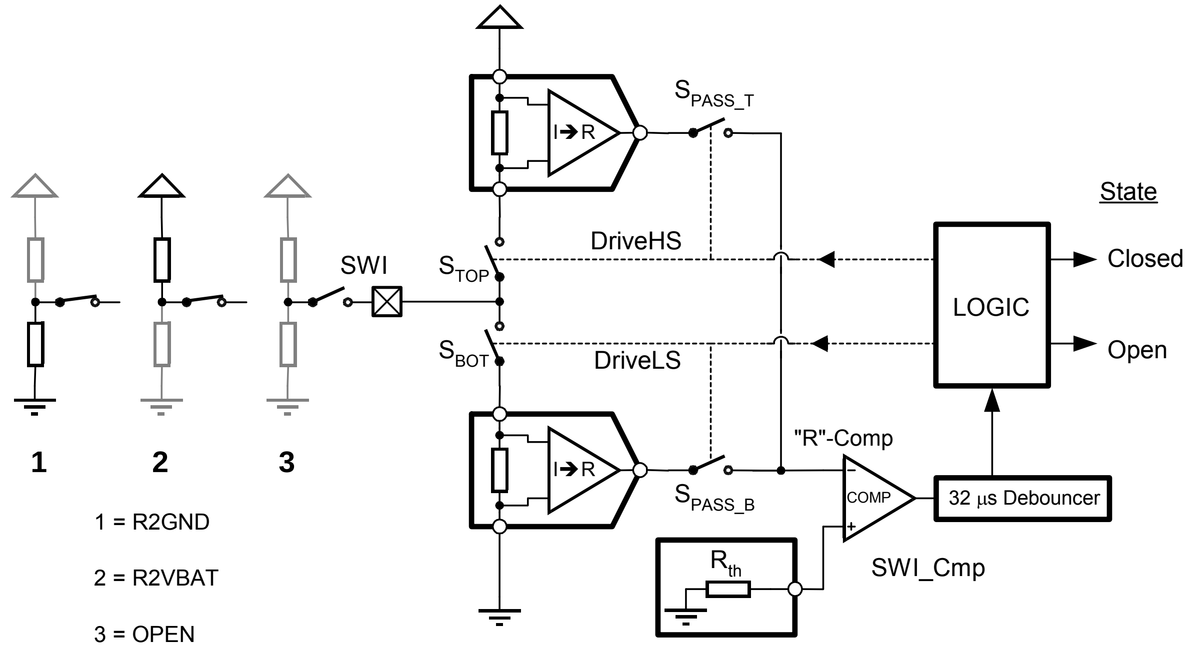


Figure 13: Simplified Schematic Diagram of the SWI Comparator

Figure 15 shows a change in state is always synchronised with DriveHS or DriveLS. The same synchronisation 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 position switch together with the corresponding position is memorized.

Using the GetActualPos commands 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 Figure 14.

Reading Frame									
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	UV2	TSD	TW	Tinfo[1:0]	

Figure 14: GetActualPos *LIN Commando*

Important remark. Every 512µs this information is refreshed.

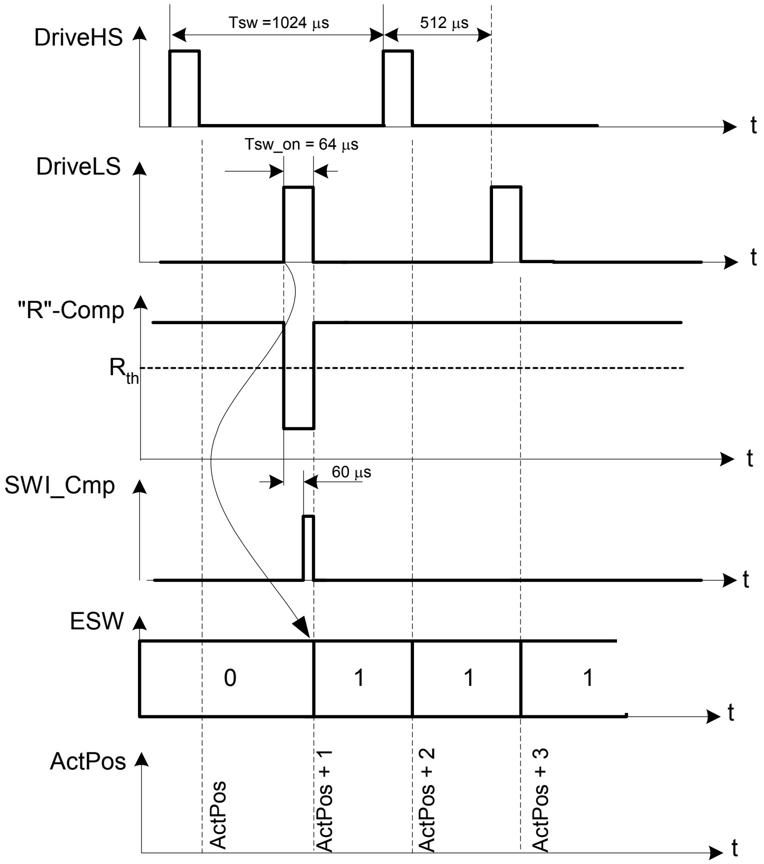


Figure 15: Timing Diagram Showing the Change in States for SWI Comparator

14.2 Main Control and Register, OTP Memory + ROM

14.2.1. Power-up Phase

Power up phase of the AMIS-30621 will not exceed 10ms. After this phase, the AMIS-30621 is in shutdown mode, ready to receive LIN messages and execute the associated commands. After power-up, the registers and flags are in the reset state; some of them being loaded with the OTP memory content (see Table 18).

14.2.2. Reset State

After power-up or after a reset occurrence (e.g. a micro cut on pin VBB has made Vdd 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 18 and Table 19.

14.2.3. 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` to avoid an attempt of the circuit to achieve the motion (see Table 18). The circuit is then ready to execute a new positioning command, provided thermal and electrical conditions allow for it.

14.2.4. 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 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. See `Sleep` command.
- The LIN bus remains inactive (or is lost) during more than 25000 time slots (1.30s at 19.2kbit/s) after which a time-out signal switches the circuit of sleep mode. See LIN Lost Behavior.

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

14.2.5. Thermal Shutdown Mode

When thermal shutdown occurs, the circuit performs a `SoftStop` and goes to motor shutdown mode (see below).

14.2.6. Temperature Management

The AMIS-30621 monitors temperature by means of two thresholds and one shutdown level, as illustrated in Figure 16. 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.

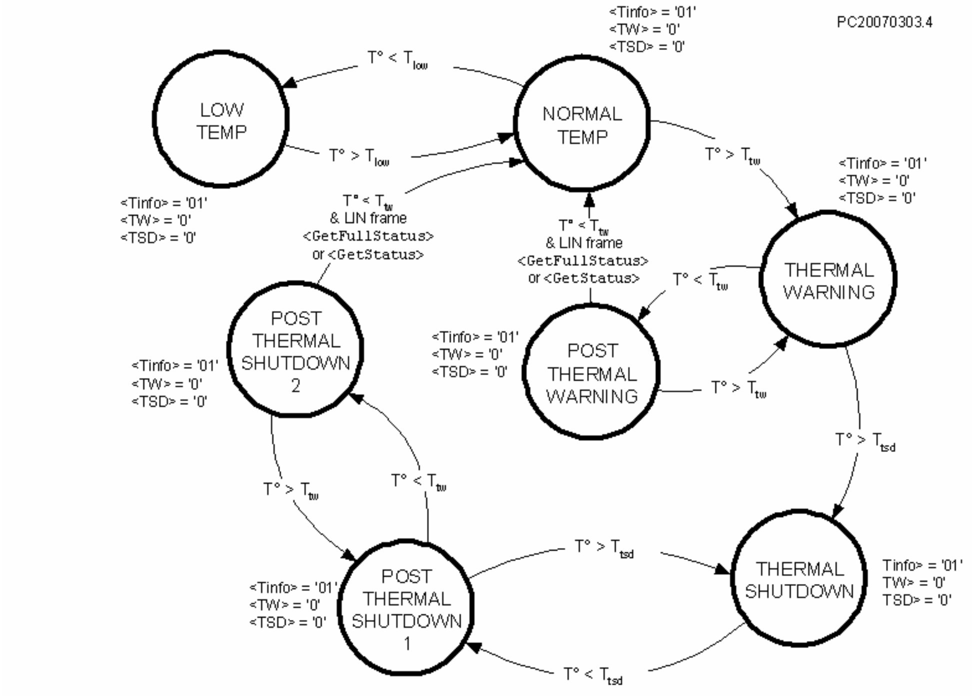


Figure 16: State Diagram Temperature Management

14.2.7. Battery Under-Voltage Management

The AMIS-30621 monitors the battery voltage by means of one threshold and one shutdown level, as illustrated in Figure 17. The only condition to reset **flags** <`UV2`> and <`StepLoss`> is to recover a battery voltage higher than UV1 and to receive a `GetStatus` or a `GetFullStatus` command.

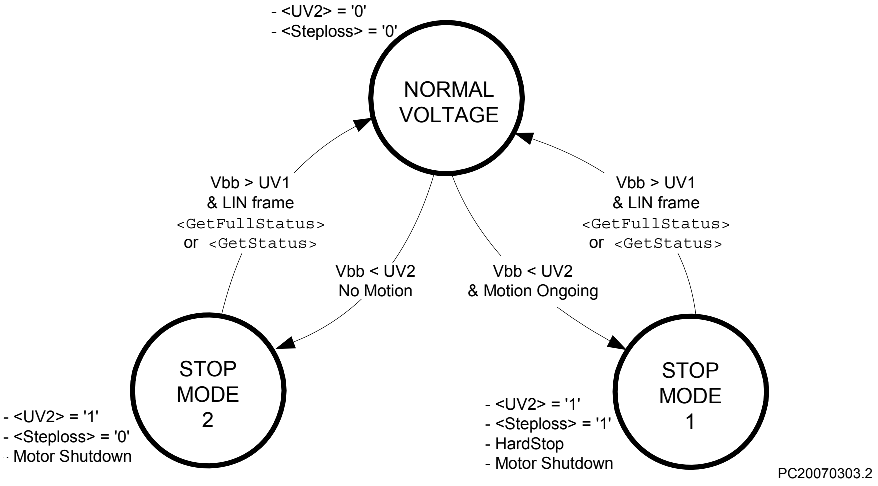


Figure 17: State Diagram Battery Voltage Management

In **Stop mode 1** the motor is put in shutdown state. The <`UV2`> **flag** is set. In case Vbb > UV1, AMIS-30621 accepts updates of the target position by means of the reception of `SetPosition`, `SetPositionShort` and `GotoSecurePosition` commands, only AFTER the <`UV2`> flag is cleared by receiving a `GetStatus` or `GetFullStatus` command.

In **Stop mode 2** the motor is stopped immediately and put in shutdown state. The <`UV2`> and <`Steploss`> **flags** are set. In case Vbb > UV1, AMIS-30621 accepts updates of the target position by means of the reception of `SetPosition`, `SetPositionShort` and `GotoSecurePosition` commands, only AFTER the <`UV2`> and <`Steploss`> flags are cleared by receiving a `GetStatus` or `GetFullStatus` command.

Important Notes:

- In the case of Stop mode 2 care needs to be taken because the accumulated steploss can cause a significant deviation between physical and stored actual position.
- The `SetDualPosition` command will only be executed after clearing the <`UV2`> and <`Steploss`> flags.
- RAM reset occurs when Vdd < VddReset (digital POR level).

14.2.8. OTP Register

OTP Memory Structure

Table 16 shows how the parameters to be stored in the OTP memory are located.

Table 16: 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	EnableLIN	TSD2	TSD1	TSD0	BG3	BG2	BG1	BG0
0x02	ADM				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	SecPos1	SecPos0
0x07					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 OPT 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 17: 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 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 really be “active” after a GetOTPparam or a ResetToDefault command or after a power-up.

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 overridden in RAM by a LIN 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.

`ADM` Addressing mode bit enabling to swap the combination of OTP memory bits PA[3:0] with hardwired address bits HW[2:0] to form the physical address AD[6:0] of the stepper motor.

`StepMode` Indicator of stepping mode to be used.

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` Indicator of reference position. If `Shaft` = ‘0’, the reference position is the maximum inner position, whereas if `Shaft` = ‘1’, the reference position is the maximum outer position.

`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)
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

`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)
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

`SecPos[10:0]` 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[10:0]` = “100 0000 0000”, this means that secure position is disabled, e.g. the stepper motor will be kept in the position occupied at the moment these events occur.

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’.

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/s²)
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

14.2.9. RAM Registers

Table 18: RAM Registers

Register	Mnemonic	Length (bit)	Related Commands	Comment	Reset State

Actual position	`ActPos`	16	`GetActualPos`
			`GetFullStatus`
			`GotoSecurePos`
			`ResetPosition`
				16-bit signed	Note 1

Last programmed position	`Pos/ TagPos`	16/11	`GetFullStatus`
			`GotoSecurePos`
			`ResetPosition`
			`SetPosition`
			`SetPositionShort`
				16-bit signed or
				11-bit signed for half stepping
				(see Positioning)	

Acceleration shape	`AccShape`	1	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				'0' ⇒ normal acceleration from Vmin to Vmax
				'1' ⇒ motion at Vmin without acceleration
					'0'

Coil peak current	`Irun`	4	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				Operating current
				See look-up table Irun
					From OTP memory

Coil hold current	`Ihold`	4	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				Standstill current
				See look-up table Ihold	

Minimum velocity	`Vmin`	4	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				See Section 13.3 Minimum Velocity
				See look-up table Vmin	

Maximum velocity	`Vmax`	4	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				See Section 13.2 Maximum Velocity
				See look-up table Vmax	

Shaft	`Shaft`	1	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				Direction of movement for positive velocity	

Acceleration/ deceleration	`Acc`	4	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				See Section 13.4 Acceleration
				See look-up table Acc	

Secure position	`SecPos`	11	`GetFullStatus`
			`ResetToDefault`
			`SetMotorParam`
				Target position when LIN connection fails; 11 MSBs of 16-bit position (LSBs fixed to '0')	

Stepping mode	`StepMode`	2	`GetFullStatus`	See Section 13.1 Stepping Modes See look-up table StepMode	

Note 1: A `ResetToDefault` command will act as a reset of the RAM content, except for `ActPos` and `TagPos` registers that are not modified. Therefore, the application should not send a `ResetToDefault` during a motion, to avoid any unwanted change of parameter.

14.2.10. Flags Table

Table 19: Flags Table

Flag	Mnemonic	Length (bit)	Related Commands	Comment	Reset State
Charge pump failure	CPFail	1	GetFullStatus	'0' = charge pump OK '1' = charge pump failure Reset only after GetFullStatus	'0'
Electrical defect	ElDef	1	GetActualPos GetStatus GetFullStatus	<OVC1> or <OVC2> or <open circuit 1> or <open circuit 2> or <CPFail> Resets only after Get(Full)Status	'0'
External switch status	ESW	1	GetActualPos GetStatus GetFullStatus	'0' = open '1' = close	'0'
Electrical flag	HS	1	Internal use	<CPFail> or <UV2> or <ElDef> or <VDDreset>	'0'
Motion status	Motion	3	GetFullStatus	"x00" = Stop "001" = inner motion acceleration "010" = inner motion deceleration "011" = inner motion max. speed "101" = outer motion acceleration "110" = outer motion deceleration "111" = outer 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" '1' otherwise	n.a.
Circuit going to sleep mode	Sleep	1	Internal use	'1' = sleep mode Reset by LIN command	'0'
Step loss	StepLoss	1	GetActualPos GetStatus GetFullStatus	'1' = step loss due to under voltage, over current or open circuit	'1'
Motor stop	Stop	1	Internal use		'0'
Temperature info	Tinfo	2	GetActualPos GetStatus GetFullStatus	"00" = normal temperature range "01" = low temperature warning "10" = high temperature warning "11" = motor shutdown	"00"
Thermal shutdown	TSD	1	GetActualPos GetStatus GetFullStatus	'1' = shutdown (> 155°C typ.) reset only after Get(Full)Status and if <Tinfo> = "00"	'0'
Thermal warning	TW	1	GetActualPos GetStatus GetFullStatus	'1' = over temp. (> 145°C) reset only after Get(Full)Status and if <Tinfo> = "00"	'0'
Battery stop voltage	UV2	1	GetActualPos GetStatus GetFullStatus	'0' = Vbb > UV2 '1' = Vbb ≤ UV2 Reset only after Get(Full)Status	'0'
Digital supply reset	VddReset	1	GetActualPos GetStatus 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 GetStatus or a GetFullStatus command.	'1'

14.2.11. Priority Encoder

Table 20 describes the state management performed by the main control block.

Table 20: Priority Encoder

State →	Stopped	GotoPos	DualPosition	SoftStop	HardStop	ShutDown	Sleep
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 1)
`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	🟦
`GetOTPparam`	OTP refresh; LIN in-frame response	OTP refresh; LIN in-frame response	OTP refresh; LIN in-frame response	OTP refresh; LIN in-frame response	OTP refresh; LIN in-frame response	OTP refresh; LIN in-frame response	🟦
`GetFullStatus` or `GetStatus` [attempt to clear `<TSD>` and `<HS>` flags]	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	🟦
`ResetToDefault` [`ActPos` and `TagPos` are not altered]	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset (note 3)	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset	🟦
`SetMotorParam` [Master takes care about proper update]	RAM update	RAM update	RAM update	RAM update	RAM update	RAM update	🟦
`ResetPosition`	`TagPos` and `ActPos` reset	🟦	🟦	🟦	🟦	`TagPos` and `ActPos` reset	🟦
`SetPosition`	`TagPos` updated; → **GotoPos**	`TagPos` updated	`TagPos` updated	🟦	🟦	🟦	🟦
`SetPositionShort` [half-step mode only)]	`TagPos` updated; → **GotoPos**	`TagPos` updated	`TagPos` updated	🟦	🟦	🟦	🟦
`GotoSecPosition`	If `<SecEn>` = '1' then `TagPos` = `SecPos`; → **GotoPos**	If `<SecEn>` = '1' then `TagPos` = `SecPos`	If `<SecEn>` = '1' then `TagPos` = `SecPos`	🟦	🟦	🟦	🟦
`DualPosition`	→ **DualPosition**	🟦	🟦	🟦	🟦	🟦	🟦
`HardStop`	🟦	→ **HardStop**; `<StepLoss>` = '1'	→ **HardStop**; `<StepLoss>` = '1'	→ **HardStop**; `<StepLoss>` = '1'	🟦	🟦	🟦
`SoftStop`	🟦	→ **SoftStop**	🟦	🟦	🟦	🟦	🟦
`Sleep` or LIN timeout [⇒ `<Sleep>` = '1', reset by any LIN command received later]	See note 9	If `<SecEn>` = '1' then `TagPos` = `SecPos` else → **SoftStop**	If `<SecEn>` = '1' then `TagPos` = `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	→ **Sleep**	🟦
`HardStop` [⇔ (`<CPFail>` or `<UV2>` or `<ElDef>`) = '1' ⇒ `<HS>` = '1']	→ **Shutdown**	→ **HardStop**	→ **HardStop**	→ **HardStop**	🟦	🟦	🟦
Thermal shutdown [`<TSD>` = '1']	→ **Shutdown**	→ **SoftStop**	→ **SoftStop**	🟦	🟦	🟦	🟦
Motion finished	n.a.	→ **Stopped**	→ **Stopped**	→ **Stopped**; `TagPos =ActPos`	→ **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 7)

Notes:

1) Leaving sleep state is equivalent to POR.
2) After POR, the shutdown state is entered. The shutdown state can only be left after `GetFullStatus` command (so that the master could read the `<VddReset>` flag).
3) 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 even if a `ResetToDefault` command is issued during a DualPosition sequence, in which case `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.
4) 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).
5) Shutdown state can be left only when `<TSD>` and `<HS>` flags are reset.
6) 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 or charge pump defect).
7) 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.
8) Some transitions like GotoPos → sleep are actually done via several states: GotoPos → SoftStop → Stopped → Sleep (see diagram below).
9) 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`
10) `<SecEn>` = ‘1’ when register `SecPos` is loaded with a value different from the most negative value (i.e. different from 0x400 = “100 0000 0000”)
11) `<Stop>` flag allows to distinguish 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.
12) Command for dynamic assignment of Ids is decoded in all states except sleep and has not effect on the current state.
13) 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>`, etc. are first evaluated for possible transitions.
14) If `<StepLoss>` is active, then `SetPosition, SetPositionShort` and `GotoSecurePosition` commands are ignored (they will not modify `TagPos` register whatever the state), and motion to secure position is forbidden after a `Sleep` command or a LIN timeout (the circuit will go into sleep state immediately, without positioning to secure position). Other command like `DualPosition` or `ResetPosition` will be executed if allowed by current state. `<StepLoss>` can only be cleared by a `GetStatus` or `GetFullStatus` command.

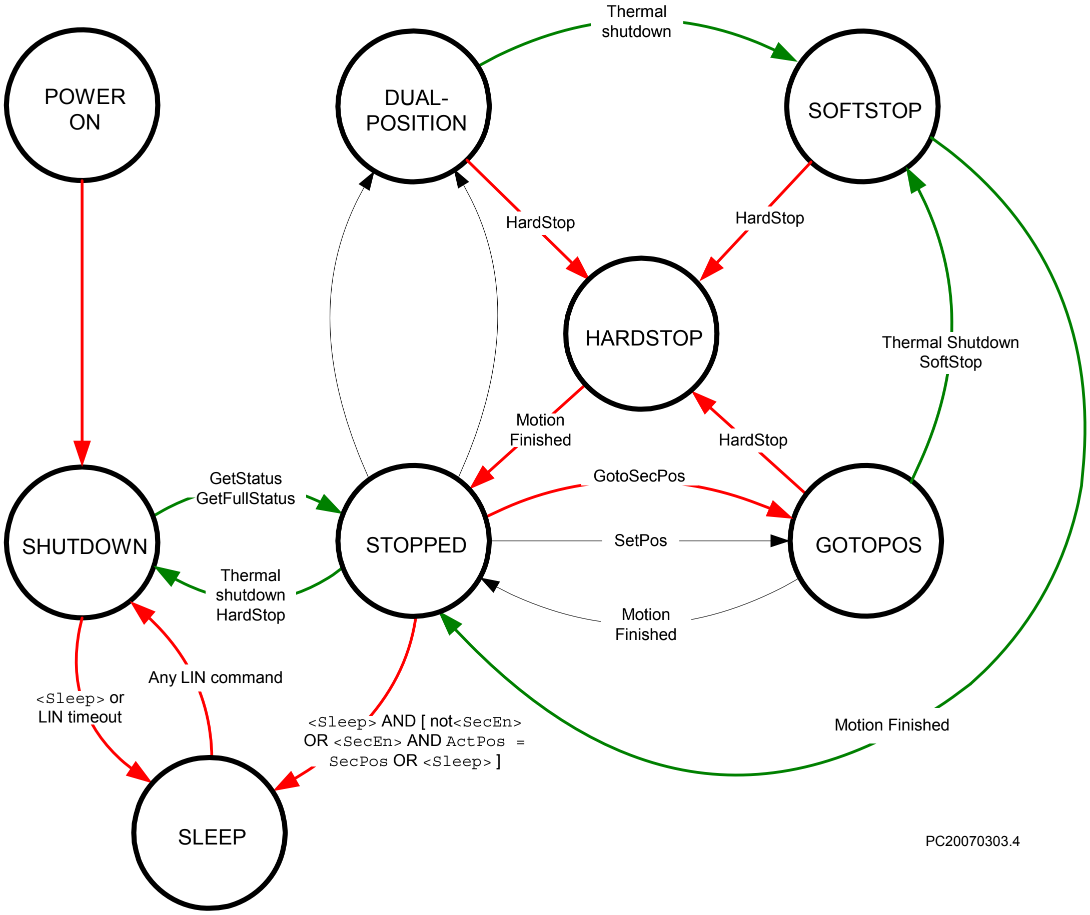


Figure 18: State Diagram

14.3 Motor Driver

14.3.1. Current Waveforms in the Coils

Figure 19 illustrates the current fed to the motor coils by the motor driver in half-step mode.

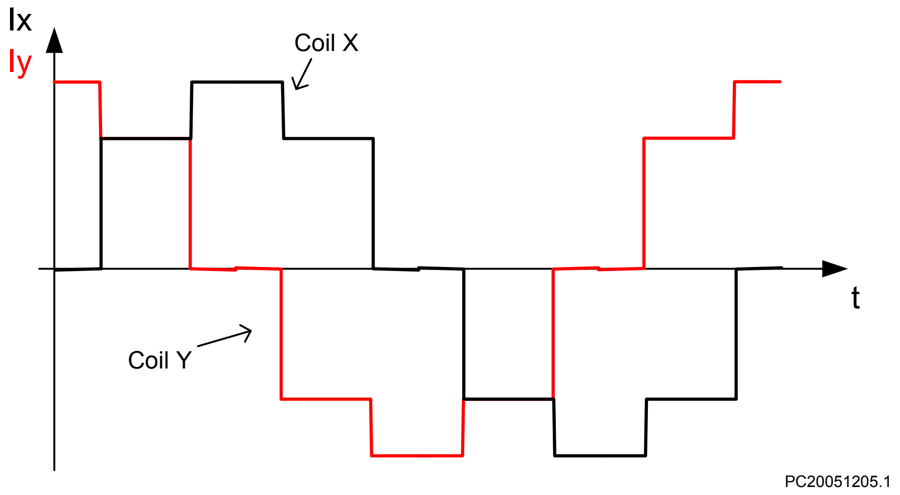


Figure 19: Current Waveforms in Motor Coils X and Y in Half-step Mode

Where as Figure 20 shows the current fed to one coil in 1/16th micro-stepping (1 electrical period).

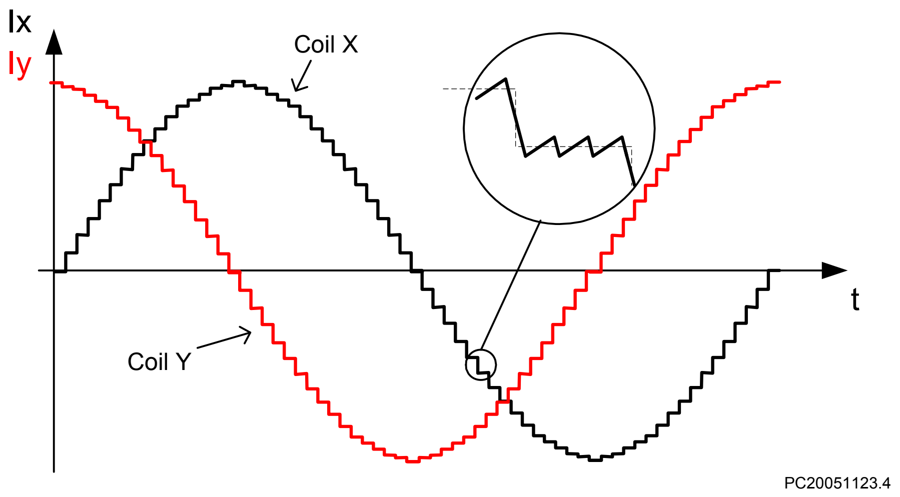


Figure 20: Current Waveforms in Motor Coils X and Y in 1/16th Micro-step Mode

14.3.2. 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 21 shows how the PWM circuit performs this regulation.

14.3.3. 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.

14.3.4. 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} equal to 1/4th of an electrical period at the `Vmin` velocity. 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.

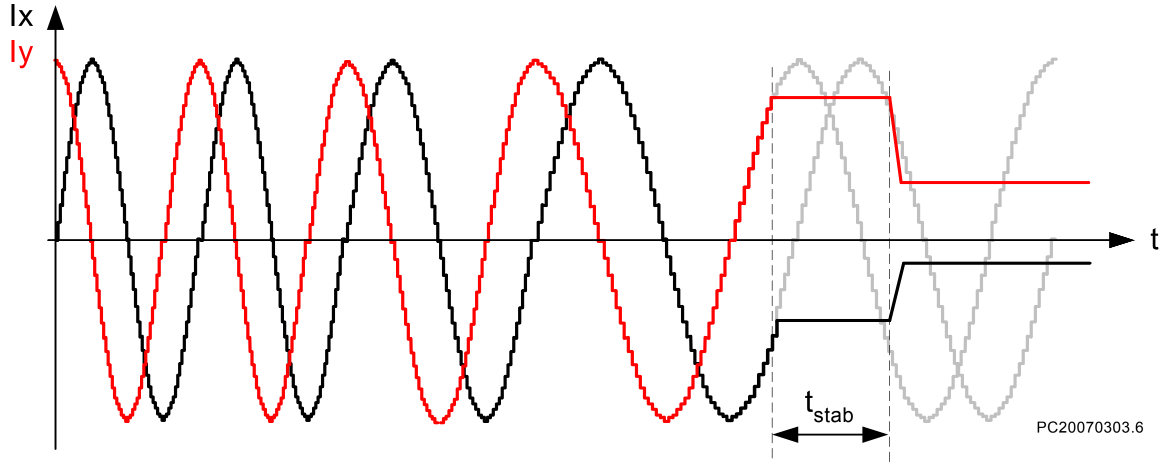


Figure 21: Motor Stopping Phase

14.3.5. Charge Pump Monitoring

If the charge pump voltage is not sufficient for driving the high side transistors (due to a failure), an internal `HardStop` command is issued. This is acknowledged to the master by raising flag <`CPFail`> (available with command `GetFullStatus`). In case this failure occurs while a motion is ongoing, the flag <`StepLoss`> is also raised.

14.3.6. 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 detecting the following short circuits:

- External coil short circuit
- Short between one terminal of the coil and Vbat or Gnd
- One cannot detect internal short in the motor

Open circuits are detected by 100 percent PWM duty cycle value during a long time.

Table 21: 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

14.3.7. 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 (see Battery Under-Voltage Management).
- Flag <ElDef> = ‘1’, meaning an electrical problem is detected on one or both coils, e.g. a short circuit.
- Flag <CPFail> = ‘1’, meaning there is a charge pump failure.

A motor shutdown leads to the following:

- H-bridges in high impedance mode.
- The TagPos register is loaded with the ActPos (to avoid any motion after leaving the motor shutdown mode).

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 more detected

Which leads to H-bridges 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 tip. The master can check whether there is a problem or not and decide which application strategy to adopt.

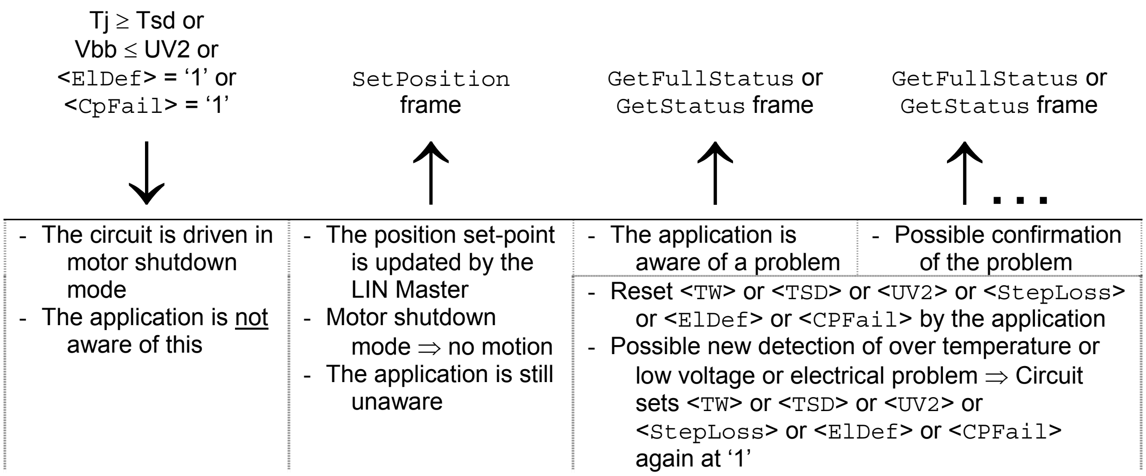


Figure 22: Example of Possible Sequence Used to Detect and Determine Cause of Motor Shutdown

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 AMIS-30621.

Warning: The application should limit the number of consecutive `GetStatus` or `GetFullStatus` commands to try to get the AMIS-30621 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.

Notes:
(0) The Priority Encoder is describing the management of states and commands. The note below is to be considered illustrative.
(1) If the LIN communication is lost while in shutdown mode, the circuit enters the sleep mode immediately.

15.0 Lin Controller

15.1 General Description

The local interconnect network (LIN) is a serial communications protocol that efficiently supports the control of mechatronic nodes in distributed automotive applications. The interface implemented in the AMIS-30621 is compliant with the LIN rev. 1.3 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-wire communication
- Transmission speed of 19.2kbit/s
- Selectable length of message frame: 2, 4 and 8 bytes
- Configuration flexibility
- Data checksum security and error detection
- Detection of defective nodes in the network

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

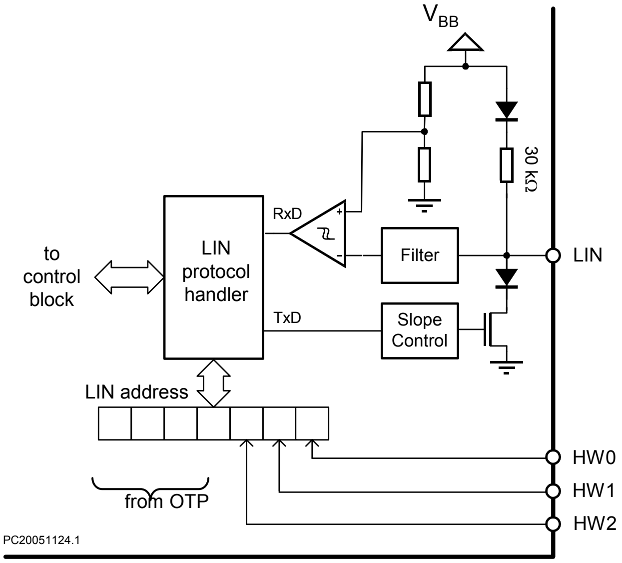


Figure 23: LIN Interface

15.2 Slave Operational Range for Proper Self Synchronization

The LIN interface will synchronize properly in the following conditions:
- Vbb ≥ 7.3V
- Ground shift between master node and slave node < ±10% Vbb

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

15.3 Functional Description

15.3.1. Analog Part

The transmitter is a low-side driver with a pull-up resistor and slope control. Figure 5 shows the characteristics of the transmitted signal, including the delay between internal TxD – and LIN signal. See AC Parameters for timing values.

The receiver mainly consists of a comparator with a threshold equal to Vbb/2. Figure 5 also shows the delay between the received signal and the internal RXD signal. See also AC Parameters for timing values.

15.3.2. Protocol Handler

This block implements:
- Bit synchronization
- Bit timing
- The MAC layer
- The LLC layer
- The supervisor

15.3.3. Electromagnetic Compatibility (EMC)

EMC behavior fulfills requirements defined by LIN specification, rev. 1.3.

15.4 Error Status Register

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

Table 22: 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:

Time out error:	The message frame is not fully completed within the maximum length T_{FRAME_MAX}
Data error flag:	Checksum error ⊕ StopBit error ⊕ Length error
Header error flag:	Parity ⊕ SynchField error
Bit error flag:	Difference in bit send and bit monitored on the LIN bus

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

15.5 Physical Address of the Circuit

The circuit must be provided with a physical address in order to discriminate it 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.

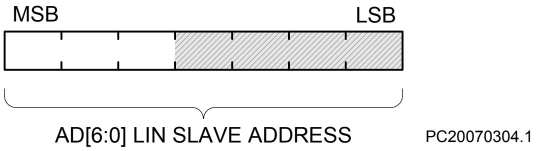


Figure 24: 7-bit LIN Address

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 three percent.

All LIN commands are using 7-bit addressing except `SetPositionShort` where only the four least significant address bits are used. These bits are shaded in Figure 24.

The physical address AD[6:0] is a combination of four OTP memory bits PA[3:0] from the OTP Memory Structure and three hardwired address bits HW[2:0]. Depending on the addressing diode (ADM –bit in OTP Memory Structure) the combination is as illustrated in Figure 25.

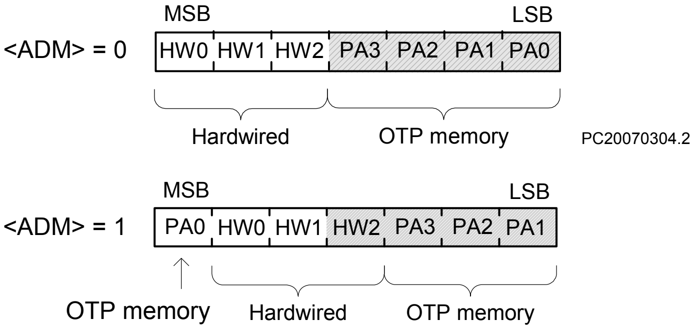


Figure 25: Combination of OTP and Hardwired Address Bits in Function of ADM

Note:
Pins HW0 and HW1 are 5V digital inputs, whereas pin HW2 is compliant with a 12V level, e.g. it can be connected to Vbat or Gnd via a terminal of the PCB. To provide cleaning current for this terminal, the system used for pin SWI is also implemented for pin HW2 (see Hardwired Address HW2).

15.6 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. 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

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

15.6.1. Writing Frames

A writing frame is sent by the LIN master to send commands and/or information to the slave nodes. 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)											

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		Data1		Data2	Data3	Data4	Data5	Data6	Data7	Data8
0x3C	00		1							
		`AppCmd`		command	physical address	parameters				

Note:
Bit 7 of byte Data1 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 AMIS-30621 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…

- **Type #2:** 2, 4 or 8 data bytes writing frame with an identifier dynamically assigned to an application command, regardless of the physical address of the circuit.
- **Type #3:** 2 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 AMIS-30621 circuits using this command connected to the LIN bus.
- **Type #4:** 8 data bytes writing frame with 0x3C identifier.

15.6.2. 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.

Notes:
(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 AMIS-30621 are the following:

- **Type #5:** 2, 4 or 8 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**: 8 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.

15.6.3. Preparing Frames

A preparing frame is a writing frame that warns a particular slave node that it will have to answer in the next frame (hence 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 AMIS-30621 can be of Type #7 or Type #8 described below.

- **Type #7:** two data bytes writing frame with dynamically assigned identifier.

Preparing 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	1	CMD[6:0]						
2	Data 2	1	AD[6:0]						

Where:

(*)	According to parity computation

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

SetDualPositioning 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	AppCMD = 0x80							
2	Data 2	1	CMD[6:0]						
3	Data 3	1	AD[6:0]						
4	Data 4	Data4[7:0]							
5	Data 5	Data5[7:0]							
6	Data 6	Data6[7:0]							
7	Data 7	Data7[7:0]							
8	Data 8	Data8[7:0]							

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.
Datan[7:0]:	Data transmitted.

15.6.4. 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 1: 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.

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	AppCmnd	0x80							
2	CMD	1	0x11						
3	Address	Broad	AD6	AD5	AD4	AD3	AD2	AD1	AD0
4	Data 1	DynID_1[3:0]				ROMp_1[3:0]			
5	Data 2	DynID_2[1:0]		ROMp_2[3:0]				DynID_1[5:4]	
6	Data 3	ROMp_3[3:0]				DynID_2[5:2]			
7	Data 4	ROMp_4[2:0]		DynID_3[5:0]					
8	Data 5	DynID_4[5:0]						ROMp_4[3:2]	

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.

DynID_x[5:0]: Dynamically assigned LIN identifier to the application command which ROM pointer is ROMp_x[3:0].

One frame allows only assigning four identifiers. Therefore, additional frames could be needed in order to assign more identifiers (maximum three for the AMIS-30621).

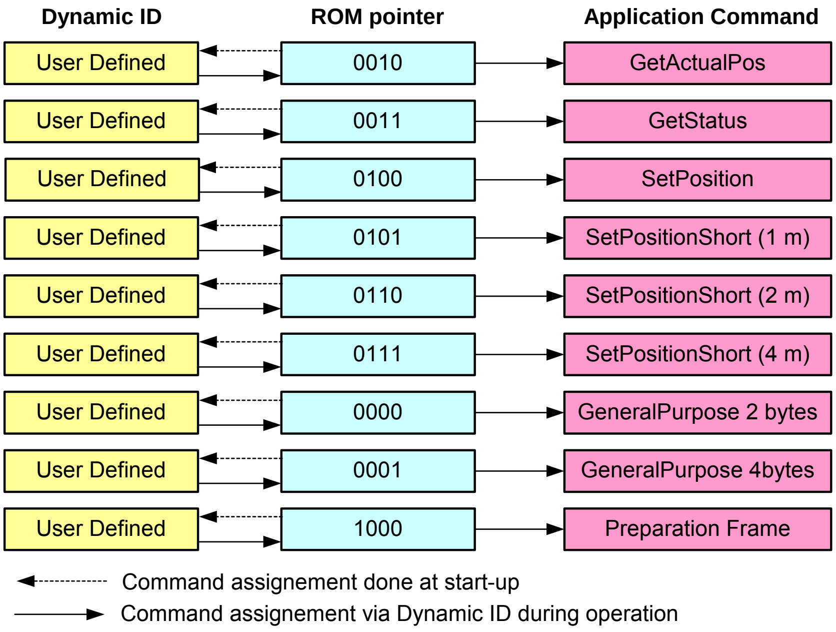


Figure 26: Principle of Dynamic Command Assignment

15.7 Commands Table

Table 23: 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.	

Command Mnemonic	Command Byte (CMD)		Dynamic ID (example)	ROM Pointer
ResetPosition	000110	0x06	n.a.	
ResetToDefault	000111	0x07	n.a.	
SetDualPosition	001000	0x08	n.a.	
SetMotorParam	001001	0x09	n.a.	
SetOTPparam	010000	0x10	n.a.	
SetPosition (16-bit)	001011	0x0B	010xxx	0100
SetPositionShort (1 motor)	001100	0x0C	001001	0101
SetPositionShort (2 motors)	001101	0x0D	101001	0110
SetPositionShort (4 motors)	001110	0x0E	111001	0111
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
Preparation frame			011010	1000

xxx allows addressing physically a slave node. Therefore, these dynamic Ids cannot be used for more than eight stepper motors. Only ten ROM pointers are needed for the AMIS-30621.

15.8 LIN Lost Behavior

15.8.1. Introduction

When the LIN communication is broken for a duration of 25000 consecutive frames (= 1.30s @ 19200kbit/s) AMIS-30621 sets an internal flag called “LIN lost”. Dependant on the content of RAM register SecPos[10:0] a motion to the secure position will start followed by entering the sleep state.

15.8.2. Motion to Secure Position

AMIS-30621 is able to perform an autonomous motion to the predefined secure position SecPos[10:0]. This positioning starts after the detection of lost LIN communication and in case RAM register SecPos[10:0] ≠ 0x400. The functional behavior depends if LIN communication is lost during normal operation (see Figure 27 case A) or at (or before) start-up (See Figure 27 state SHUTDOWN):

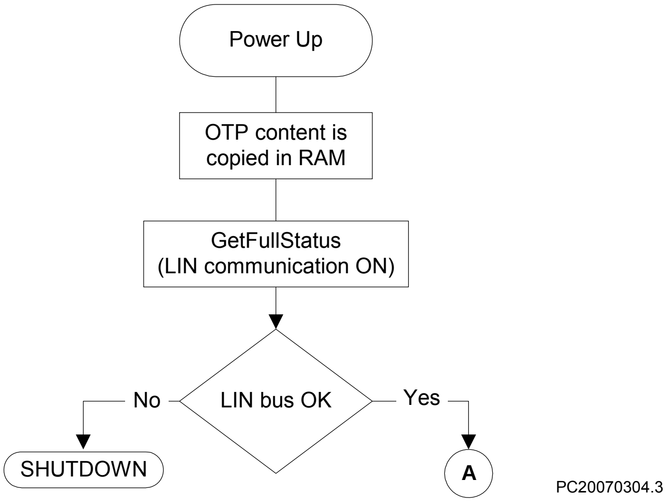


Figure 27: Flow Chart Power-up of AMIS-30621. Two Cases are Illustrated; Case A: LIN Lost During Operation and LIN Lost at Start-up Resulting in Shutdown

LIN Lost During Normal Operation

If the LIN communication is lost during normal operation, it is assumed that AMIS-30621 is referenced. 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.

Following sequence will be followed. See Figure 28.

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`.
2. If the LIN communication is lost there are two possibilities:
   I. If SecPos[10:0] = 0x400:
   No secure positioning will be performed
   AMIS-30621 will enter the SLEEP state
   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).
   After the positioning is finished AMIS-30621 will enter the SLEEP state.

Important Remarks:
(1) The secure position has a resolution of 11 bit.
(2) Same behavior in case of HW2 float (= lost LIN address). See also Hardwired Address HW2

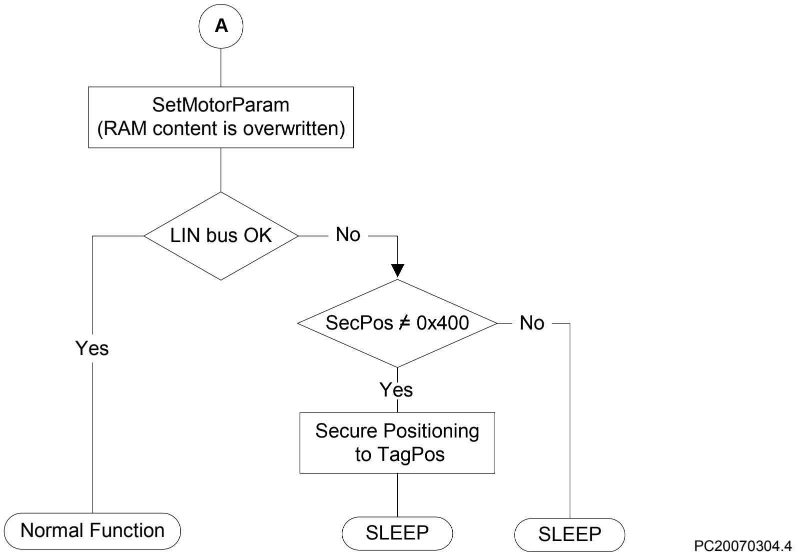


Figure 28: Case A: LIN Lost During Normal Operation

LIN Lost Before or at Power On

If the LIN communication is lost before or at power on, no correct GetFullStatus command is received. For that reason the ShutDown state is not left and the stepper motor is kept un-powered.

16.0 LIN Application Commands

16.1 Introduction

The LIN master will have to use commands to manage the different application tasks the AMIS-30621 can feature. The commands summary is given in Table 24.

Table 24: 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
`ResetToDefault`	0x07				RAM content reset
`SetDualPosition`	0x08			4	Drives the motor to two different positions with different speeds
`SetMotorParam`	0x09			4	Programs the motion parameters and values for the current in the motor's coils
`SetOTPparam`	0x10				Programs (and zaps) a selected byte of the OTP memory
`SetPosition`	0x0B			1, 3, 4	Drives the motor to a given position
`SetPositionShort` (1 motor)	0x0C			2	Drives the motor to a given position (half-step mode only)
`SetPositionShort` (2 motors)	0x0D			2	Drives two motors to two given positions (half-step only)
`SetPositionShort` (4 motors)	0x0E			2	Drives four motors to four given positions (half-step only)
Service Commands					
`Sleep`				1	Drives circuit into sleep mode
`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 in Figure 29.

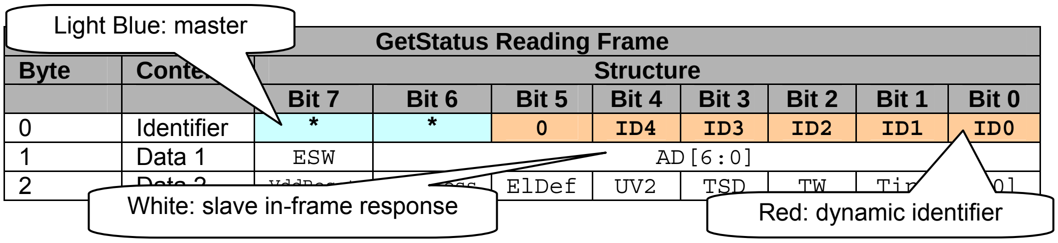


Figure 29: Color Code Used in the Definition of LIN Frames

Usually, the AMIS-30621 makes use of dynamic identifiers for general-purpose 2, 4 or 8 bytes writing frames. If dynamic identifiers are used for other purpose, this is acknowledged. Some frames implement a `Broad` bit that allows addressing a command to all the AMIS-30621 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.

16.2 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 LSBs of `ActPos[15:0]` may have no meaning and should be assumed to be '0', as described 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 GetActualPosition 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)

Reading Frame									
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	UV2	TSD	TW	Tinfo[1:0]	

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.

2) One preparing frame prior four data bytes in-frame response with **0x3D** indirect ID

Preparing 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	1	CMD[6:0] = 0x00						
2	Data 2	1	AD[6:0]						

Reading Frame									
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	UV2	TSD	TW	Tinfo[1:0]	
5	Data 5	0xFF							
6	Data 6	0xFF							
7	Data 7	0xFF							
8	Data 8	0xFF							

Where:

(*) According to parity computation.

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`>, <`ElDef`>, <`StepLoss`>, <`CPFail`>, <`OVC1`>, <`OVC2`> and <`VddReset`>.

`GetFullStatus` corresponds to two successive LIN in-frame responses with **0x3D** indirect ID.

Preparing 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	`1`	`CMD[6:0] = 0x01`						
2	Data 1	`1`	`AD[6:0]`						

Reading Frame 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`	`UV2`	`TSD`	`TW`	`Tinfo[1:0]`	
6	Data 6	`Motion[2:0]`			`ESW`	`OVC1`	`OVC2`	`1`	`CPFail`
7	Data 7	0	0	0	0	`TimeE`	`DataE`	`HeadE`	`BitE`
8	Data 8	`0xFF`							

Reading Frame 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	1	1	1	1	`1`	`SecPos[10:8]`		
8	Data 8	`0xFF`							

Where:
(*) According to parity computation.

Important: It is not mandatory for the LIN master to initiate the second in-frame response if `ActPos`, `TagPos` and `SecPos` are not needed by the application.

GetOTPparam

This command is provided to the circuit by the LIN master after a preparation frame (see Preparing frames) was issued, 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.

Preparing 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	1	CMD[6:0] = 0x02						
2	Data 2	1	AD[6:0]						

Reading Frame									
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	OTP byte @0x05							
7	Data 7	OTP byte @0x06							
8	Data 8	OTP byte @0x07							

Where:

(*****) According to parity computation.

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`>, <`UV2`>, <`ElDef`>, <`StepLoss`>, and <`VddReset`>. `GetStatus` corresponds to a two data bytes LIN in-frame response with a direct ID (type #5).

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	UV2	TSD	TW	Tinfo[1:0]	

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 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.

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

GotoSecurePosition Writing 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	CMD[6:0] = 0x04						
2	Data	Broad	AD[6:0]						

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.

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).

HardStop Writing 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	CMD[6:0] = 0x05						
2	Data	Broad	AD[6:0]						

Where:
(*) According to parity computation.
Broad: If Broad = ‘0’ 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.

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

ResetPosition Writing 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	CMD[6:0] = 0x06						
2	Data	Broad	AD[6:0]						

Where:
(*) According to parity computation.
Broad: If Broad = ‘0’ all the circuits connected to the LIN bus will reset their `ActPos` and `TagPos` registers.

ResetToDefault

This command is provided to the circuit by the LIN master in order to reset the whole slave node into the initial state. `ResetToDefault` will, for instance, overwrite the RAM with the reset state of the registers parameters (See RAM Registers). This is another way for the LIN master to initialize a slave node in case of emergency, or simply to refresh the RAM content.

Note: `ActPos` and `TagPos` are not modified by a `ResetToDefault` command.

Important: Care should be taken not to send a `ResetToDefault` command while a motion is ongoing, since this could modify the motion parameters in a way forbidden by the position controller.

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

ResetPosition Writing 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	CMD[6:0] = 0x07						
2	Data	Broad	AD[6:0]						

Where:
(*) According to parity computation.
Broad: If Broad = ‘0’ all the circuits connected to the LIN bus will reset to default.

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 Section Dual Positioning.

Note 1: This sequence cannot be interrupted by another positioning command.

Important: If for some reason `ActPos` equals `Pos1[15:0]` at the moment the `SetDualPosition` command is issued, the circuit will enter in deadlock state. Therefore, the application should check the actual position by a `GetPosition` or a `GetFullStatus` command prior to start a dual positioning. Another solution may consist of programming a value out of the stepper motor range for `Pos1[15:0]`. For the same reason `Pos2[15:0]` should not be equal to `Pos1[15:0]`.

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

SetDualPositioning 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	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]							

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]: Position to be reached during the first motion.
Pos2[15:0]: Position to be reached during the second motion.

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).

SetMotorParam 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	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	1	1	1	AccShape	StepMode[1:0]		1	1

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[2:0]`, and to zap it.

Important: This command must be sent under a specific Vbb 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).

SetOTPParam 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	AppCMD = 0x80							
2	Data 2	1	CMD[6:0] = 0x10						
3	Data 3	**Broad**	AD[6:0]						
4	Data 4	1	1	1	1	1	OTPA[2:0]		
5	Data 5	D[7:0]							
6	Data 6	0xFF							
7	Data 7	0xFF							
8	Data 8	0xFF							

Where:

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

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.

The priority encoder table (See Priority Encoder) acknowledges the cases where a `SetPosition` command will be ignored. `SetPosition` corresponds to the following LIN write frames.

1) Two data bytes frame with a direct ID (type #3)

SetPosition Writing 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	Pos[15 :8]							
2	Data 2	Pos[7 :0]							

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 data bytes frame with a general purpose identifier (type #1)

SetPosition Writing Frame									
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]							

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 motors positioning frame with **0x3C** identifier (type #4)

SetPosition 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	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]							

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. It applies only for half stepping mode (`StepMode[1:0]` = “00”) and is ignored when in other stepping modes. See Positioning for more details. The physical address is coded on four bits, hence `SetPositionShort` can only be used with a network implementing a maximum of 16 slave nodes. These four address bits are corresponding to bits `AD[3:0]`. See Physical Address of the Circuit.

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

`SetPositionShort` corresponds to the following LIN writing frames

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

SetPositionShort Writing 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	Pos[10:8]			Broad	AD [3:0]			
2	Data 2	Pos [7:0]							

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

SetPositionShort Writing Frame									
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	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]							

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 data bytes frame for four motors, with specific identifier (type #2)

SetPositionShort Writing Frame									
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]							

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).

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.

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							

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: 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 LIN master requests a `SoftStop`. Hence `SoftStop` will correspond to the following two data bytes LIN writing frame (type #1).

SoftStop Writing 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	1	CMD[6:0] = 0x0F						
2	Data 2	Broad	AD[6:0]						

Where:

(*****)	According to parity computation.
Broad:	If Broad = '0' all the stepper motors connected to the LIN bus will stop with deceleration.

17.0 Resistance to Electrical and Electromagnetic Disturbances

17.1 Electrostatic Discharges

Table 25: Absolute Maximum Ratings

Parameter		Min.	Max.	Unit
Vesd [(1)]	Electrostatic discharge voltage on LIN pin	-4	+4	kV
	Electrostatic discharge voltage on other pins	-2	+2	kV

Note:
(1) Human body model (100 pF via 1.5 kΩ, according to JEDEC std.).

17.2 Electrical Transient Conduction Along Supply Lines

Test pulses are applied to the power supply wires of the equipment implementing the AMIS-30621 (see application schematic), according to ISO 7637-1 document. Operating Classes are defined in ISO 7637-2.

Table 26: Test Pulses and Test Levels According to ISO 7637-1

Pulse	Amplitude	Rise Time	Pulse Duration	Rs	Operating Class
#1	-100V	≤ 1µs	2ms	10Ω	C
#2a	+100V	≤ 1µs	50µs	2Ω	B
#3a	-150V (from +13.5V)	5ns	100ns (burst)	50Ω	A
#3b	+100V (from +13.5V)	5ns	100ns (burst)	50Ω	A
#5b (load dump)	+21.5V (from +13.5V)	≤ 10ms	400ms	≤ 1Ω	C

17.3 EMC

Bulk current injection (BCI), according to ISO 11452-4. Operating classes are defined in ISO 7637-2.

Table 27: Bulk Current Injection Operating Classes

Current	Operating Class
60mA envelope	A
100mA envelope	B
200mA envelope	C

17.4 Power Supply Micro-interruptions

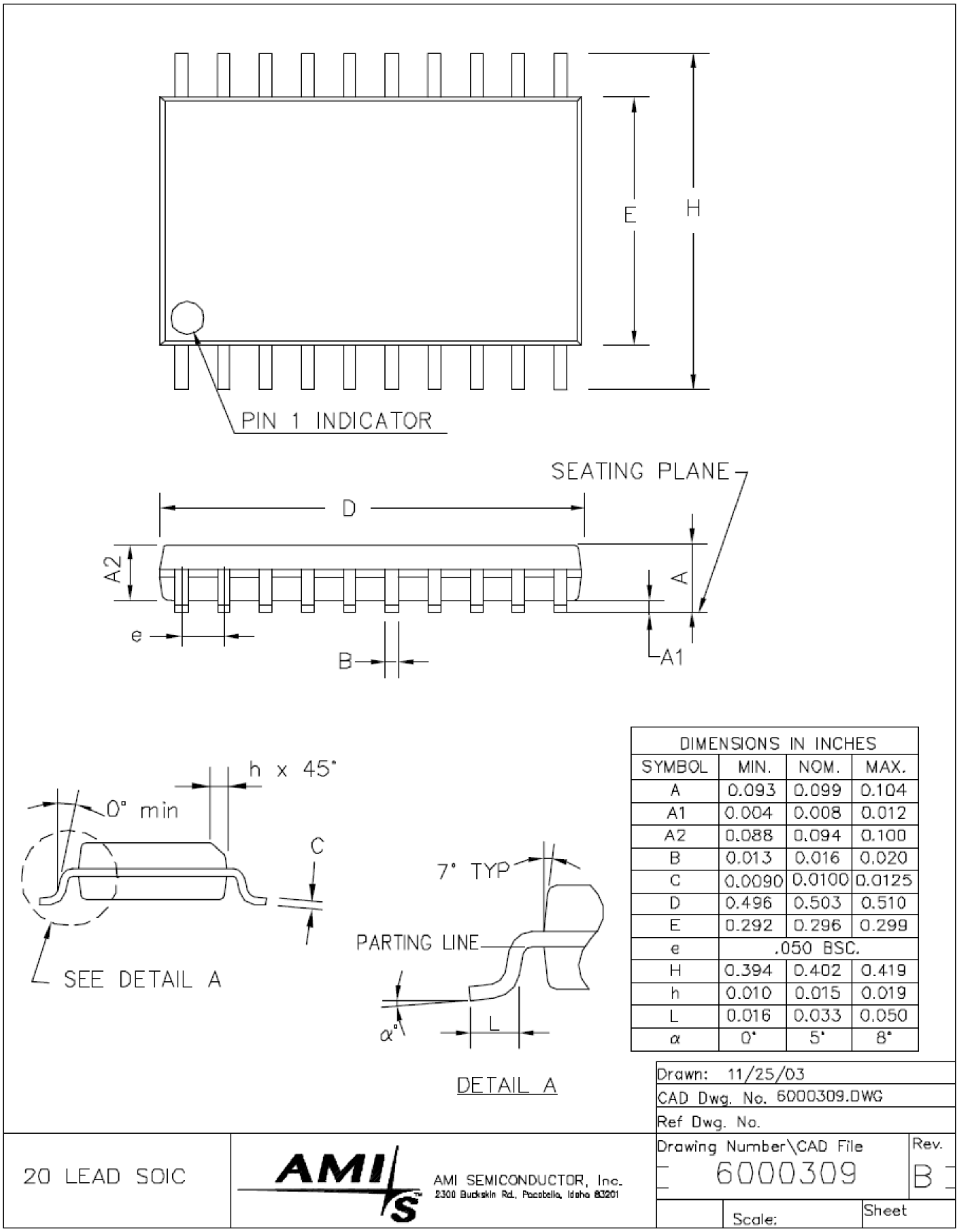
According to ISO 16750-2.

Table 28: Immunity to Power Supply Micro-interruptions

Test	Operating Class
10µs micro-interruptions	A
100µs micro-interruptions	B
5ms micro-interruptions	B
50ms micro-interruptions	C
300ms micro-interruptions	C

18.0 Package Outline

18.1 SOIC-20: Plastic Small Outline; 20 Leads; Body Width 300 mil AMIS reference: SOIC300 20 300G

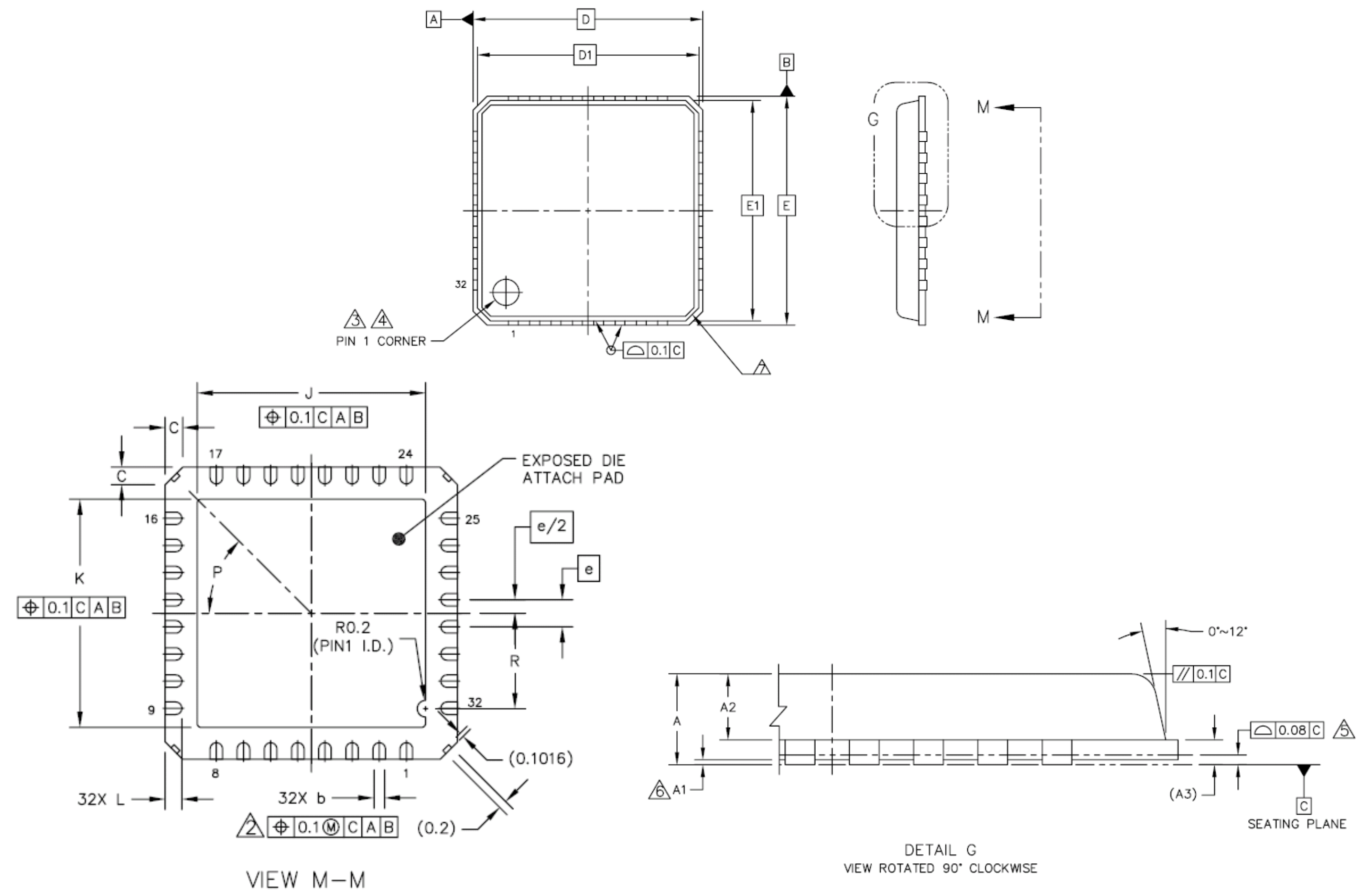


DIMENSIONS IN INCHES

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	0.292	0.296	0.299
e	.050 BSC.		
H	0.394	0.402	0.419
h	0.010	0.015	0.019
L	0.016	0.033	0.050
α	0°	5°	8°

18.2 NQFP-32: No Lead Quad Flat Pack; 32 Pins; Body Size 7 x 7 mm

AMIS reference: NQFP-32



Dimensions:

Dim	Min	Nom	Max	Unit
A	0.8		0.9	mm
A1	0	0.02	0.05	mm
A2	0.576	0.615	0.654	mm
A3		0.203		mm
b	0.25	0.3	0.35	mm
C	0.24	0.42	0.6	mm
D		7		mm
D1		6.75		mm
E		7		mm
E1		6.75		mm
e		0.65		mm
J	5.37	5.47	5.57	mm
K	5.37	5.47	5.57	mm
L	0.35	0.4	0.45	mm
P		45		Degree
R	2.185		2.385	mm

Notes

2) Dimensions apply to plated terminal and are measured between 0.2 and 0.25 mm from terminal tip.

3) The pin #1 indication must be placed on the top surface of the package by using indentation mark or other feature of package body.

4) Exact shape and size of this feature is optional

5) Applied for exposed pad and terminals. Exclude embedding part of exposed pad from measuring.

6) Applied only to terminals

7) Exact shape of each corner is optional

7x7 NQFP

19.0 Soldering

19.1 Introduction to Soldering Surface Mount Packages

This text gives a very brief insight to a complex technology. A more in-depth account of soldering ICs can be found in the AMIS "Data Handbook IC26; Integrated Circuit Packages" (document order number 9398 652 90011). There is no soldering method that is ideal for all surface mount IC packages. Wave soldering is not always suitable for surface mount ICs, or for printed-circuit boards (PCBs) with high population densities. In these situations re-flow soldering is often used.

19.2 Re-flow Soldering

Re-flow soldering requires solder paste (a suspension of fine solder particles, flux and binding agent) to be applied to the PCB by screen printing, stencilling or pressure-syringe dispensing before package placement. Several methods exist for re-flowing; for example, infrared/convection heating in a conveyor type oven.

Throughput times (preheating, soldering and cooling) vary between 100 and 200 seconds depending on heating method. Typical re-flow peak temperatures range from 215 to 260°C. The top-surface temperature of the packages should preferably be kept below 230°C.

19.3 Wave Soldering

Conventional single wave soldering is not recommended for surface mount devices (SMDs) or PCBs with a high component density, as solder bridging and non-wetting can present major problems. To overcome these problems the double-wave soldering method was specifically developed.

If wave soldering is used the following conditions must be observed for optimal results:

- Use a double-wave soldering method comprising a turbulent wave with high upward pressure followed by a smooth laminar wave.
- For packages with leads on two sides and a pitch (e):
    - Larger than or equal to 1.27mm, the footprint longitudinal axis is preferred to be parallel to the transport direction of the PCB;
    - Smaller than 1.27mm, the footprint longitudinal axis must be parallel to the transport direction of the PCB. The footprint must incorporate solder thieves at the downstream end.
    - For packages with leads on four sides, the footprint must be placed at a 45° angle to the transport direction of the PCB. The footprint must incorporate solder thieves downstream and at the side corners.

During placement and before soldering, the package must be fixed with a droplet of adhesive. The adhesive can be applied by screen printing, pin transfer or syringe dispensing. The package can be soldered after the adhesive is cured. Typical dwell time is four seconds at 250°C. A mildly-activated flux will eliminate the need for removal of corrosive residues in most applications.

19.4 Manual Soldering

Fix the component by first soldering two diagonally-opposite end leads. Use a low voltage (24V or less) soldering iron applied to the flat part of the lead. Contact time must be limited to 10 seconds at up to 300°C.

When using a dedicated tool, all other leads can be soldered in one operation within two to five seconds between 270 and 320°C.

Table 29: Soldering Process

Package	**Soldering Method**	
	Wave	**Re-flow** [(1)]
BGA, SQFP	Not suitable	Suitable
HLQFP, HSQFP, HSOP, HTSSOP, SMS	Not suitable (2)	Suitable
PLCC (3) , SO, SOJ	Suitable	Suitable
LQFP, QFP, TQFP	Not recommended (3)(4)	Suitable
SSOP, TSSOP, VSO	Not recommended (5)	Suitable

Notes:

(1) All surface mount (SMD) packages are moisture sensitive. Depending upon the moisture content, the maximum temperature (with respect to time) and body size of the package, there is a risk that internal or external package cracks may occur due to vaporization of the moisture in them (the so called popcorn effect). For details, refer to the drypack information in the "Data Handbook IC26; Integrated Circuit Packages; Section: Packing Methods."

(2) These packages are not suitable for wave soldering as a solder joint between the printed-circuit board and heatsink (at bottom version) can not be achieved, and as solder may stick to the heatsink (on top version).

(3) If wave soldering is considered, then the package must be placed at a 45° angle to the solder wave direction. The package footprint must incorporate solder thieves downstream and at the side corners.

(4) Wave soldering is only suitable for LQFP, TQFP and QFP packages with a pitch (e) equal to or larger than 0.8mm; it is definitely not suitable for packages with a pitch (e) equal to or smaller than 0.65mm.

(5) Wave soldering is only suitable for SSOP and TSSOP packages with a pitch (e) equal to or larger than 0.65mm; it is definitely not suitable for packages with a pitch (e) equal to or smaller than 0.5mm.

20.0 Company or Product Inquiries

For more information about AMI Semiconductor's products or services visit our Web site at: http://www.amis.com.

21.0 Document History

Table 30: Document history

Version	Date of Version	Modifications/Additions
1.4	March 3, 2003	First non-preliminary version for public release
1.5	September 6, 2007	Various cosmetic changes, addition of detailed feature descriptions and U-product options.