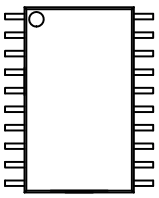
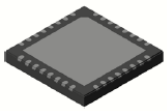
Micro-Stepping Motor Driver

AMIS-30621



SOIC−20
3 & 7 SUFFIX
CASE 751AQ



QFNW32 7x7, 0.65P
CASE 484BB

INTRODUCTION

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 I2T100 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.

PRODUCT FEATURES

Motordriver

- Micro−Stepping Technology
- Peak Current Up to 800 mA
- Fixed Frequency PWM Current−Control
- Automatic Selection of Fast and Slow Decay Mode
- No External Fly−Back Diodes Required
- Compliant with 14 V Automotive Systems and Industrial Systems Up to 24 V

Controller with RAM and OTP Memory

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

LIN Interface

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

Protection

- Overcurrent Protection
- Undervoltage Management
- Open−Circuit Detection
- High Temperature Warning and Management
- Low Temperature Flag
- LIN Bus Short−Circuit Protection to Supply and Ground
- Lost LIN Safe Operation

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

ORDERING INFORMATION

See detailed ordering and shipping information on page 2 of this data sheet.

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

Power Saving

- Powerdown Supply Current < 50 μA
- 5 V Regulator with Wake−up on LIN Activity

EMI Compatibility

- LIN Bus Integrated Slope Control
- HV Outputs with Slope Control

- These are Pb−Free Devices

© Semiconductor Components Industries, LLC, 2010

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.

Table 1. ORDERING INFORMATION

Part No.	Peak Current	UV*	Package	Shipping†
AMIS30621C6213G	800 mA	High	SOIC−20 (Pb−Free)	Tube / Tray
AMIS30621C6213RG	800 mA	High		Tape & Reel
AMIS30621C6216G	800 mA	Low	QFNW32 7x7 (Pb−Free)	Tube / Tray
AMIS30621C6216RG	800 mA	Low		Tape & Reel
AMIS30621C6217G**	800 mA	Low	SOIC−20 (Pb−Free)	Tube / Tray
AMIS30621C6217RG**	800 mA	Low		Tape & Reel

†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specification Brochure, BRD8011/D.
*UV undervoltage lock out levels: see DC Parameters UV1 & UV2 (Stop Voltage thresholds).
** For prodcut versions AMIS30621C6217G and AMIS30621C6217RG the Ihold0 bit in OTP is programmed to ‘1’.

QUICK REFERENCE DATA

Table 2. ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Min	Max	Unit
V_{BB}, VHW2, VSWI	Supply voltage, Hardwired Address and SWI Pins	−0.3	+40 (Note 1)	V
V_{lin}	Bus input voltage	−40	+40	V
T_J	Junction temperature range (Note 2)	−50	+175	°C
T_{st}	Storage temperature	−55	+160	°C
V_{esd}	Human Body Model Electrostatic discharge voltage on LIN pin (Note 3)	−4	+4	kV
	Human Body Model Electrostatic discharge voltage on other pins (Note 3)	−2	+2	kV
	CDM Electrostatic discharge voltage on other pins (Note 4)	−500	+500	V

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.
1. For limited time: V_{BB} < 0.5 s, SWI and HW2 pins < 1.0 s.
2. The circuit functionality is not guaranteed.
3. Human Body Model according to MIL−STD−883 Method 3015.7, measured on SOIC devices, and according to AEC−Q100: EIA−JESD22−A114−B (100 pF via 1.5 kΩ) measured on QFNW device.
4. CDM according to EOS_ESD−DS5.3−1993 (draft)−socketed mode, measured on SOIC devices, and according to AEC−Q100: EIA−JESD22−A115−A measured on QFNW devices.

Table 3. OPERATING RANGES

Symbol	Parameter	Min	Max	Unit
V_{BB}	Supply voltage	+6.5	+29	V
T_J	Operating temperature range (Note 5)	−40	+165	°C

5. Note that the thermal warning and shutdown will get active at the level specified in the “DC Parameters”. No more than 100 cumulated hours in life time above T_{tw}.

Table of Contents

General Description 1
Product Features 1
Applications 2
Ordering Information 2
Quick Reference Data 2
Maximum Ratings 2
Block Diagram 3
Pin Description 4
Package Thermal Resistance 5
DC Parameters 6
AC Parameters 8
Typical Application 9
Positioning Parameters 10
Structural Description 13
Functions Description 14
Lin Controller 33
LIN Application Commands 42

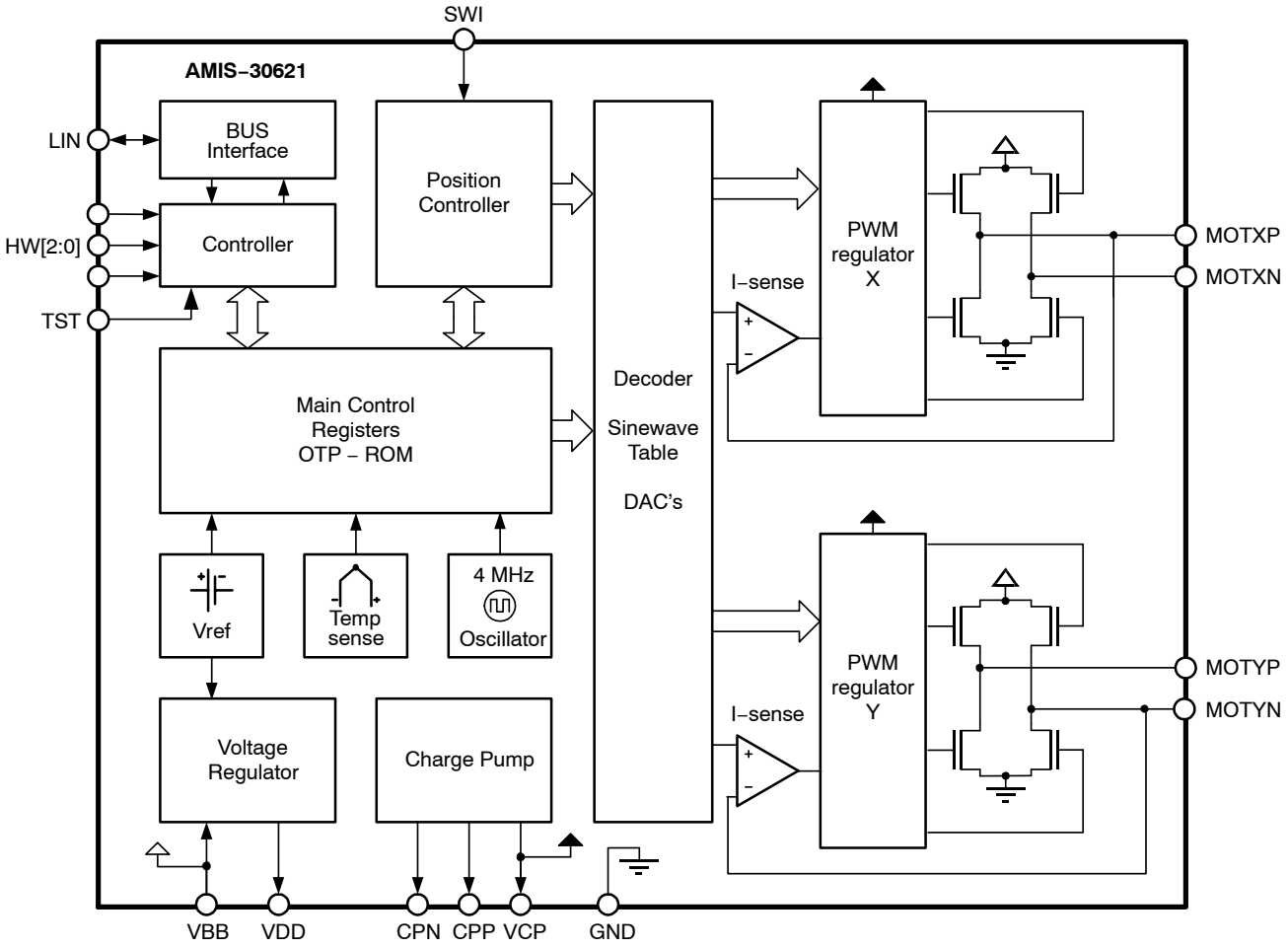


Figure 1. Block Diagram

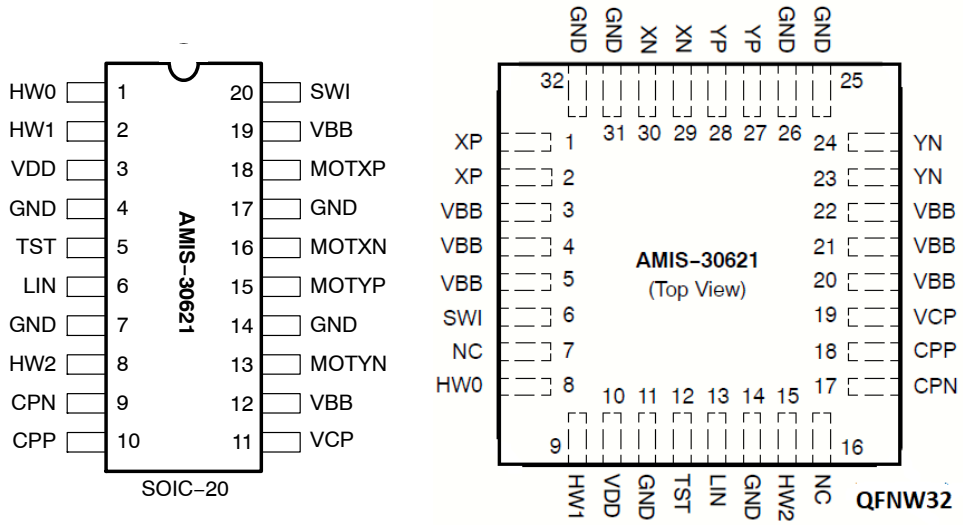


Figure 2. SOIC−20 and QFNW32 Pin−out

Table 4. PIN DESCRIPTION

Pin Name	Pin Description		SOIC−20	QFNW32
HW0	Bit 0 of LIN−ADD	To be Tied to GND or V_{DD}	1	8
HW1	Bit 1 of LIN−ADD		2	9
V_{DD}	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
V_{BB}	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 (to be tied to ground)			7, 16

PACKAGE THERMAL RESISTANCE

The AMIS−30621 is available in SOIC−20 and optimized QFNW32 packages. For cooling optimizations, the QFNW32 has an exposed thermal pad which has to be soldered to the PCB ground plane. The ground plane needs thermal vias to conduct the head to the bottom layer. Figures 3 and 4 give examples for good power distribution solutions.

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

- Static environmental air (via the case)
- PCB board copper area (via the device pins and exposed pad)

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

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

The QFNW32 device is designed to provide superior thermal performance. Using an exposed die pad on the bottom surface of the package, is mainly contributing to this performance. In order to take full advantage of the exposed pad, it is most important that the PCB has features to conduct heat away from the package. A thermal grounded pad with thermal vias can achieve this.

In below table, one can find the values for the Rthja and Rthjp, simulated according to the JESD−51standard:

Package	Rth Junction−to−Leads and Exposed Pad (Rthjp)	Rth Junction−to−Leads (Rthjp)	Rth Junction−to−Ambient Rthja 1S0P	Rth Junction−to−Ambient Rthja 2S2P
SOIC−20		19	62	39
QFNW32	0.95		60	30

The Rthja for 2S2P is simulated conform to JESD−51 as follows:

- A 4−layer printed circuit board with inner power planes and outer (top and bottom) signal layers is used
- Board thickness is 1.46 mm (FR4 PCB material)
- The 2 signal layers: 70 μm thick copper with an area of 5500 mm^2 copper and 20% conductivity
- The 2 power internal planes: 36 μm thick copper with an area of 5500 mm^2 copper and 90% conductivity

The Rthja for 1S0P is simulated conform to JESD−51 as follows:

- A 1−layer printed circuit board with only 1 layer
- Board thickness is 1.46 mm (FR4 PCB material)
- The layer has a thickness of 70 μm copper with an area of 5500 mm^2 copper and 20% conductivity

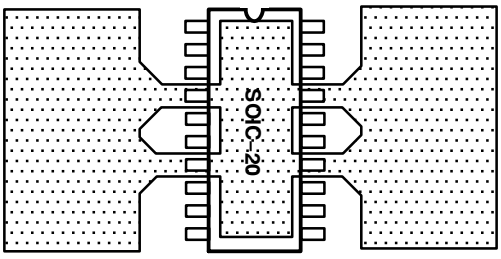


Figure 3. Example of SOIC−20 PCB Ground Plane Layout (Preferred Layout at Top and Bottom)

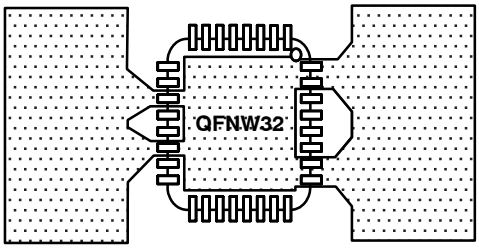


Figure 4. Example of QFNW32 PCB Ground Plane Layout (Preferred Layout at Top and Bottom)

DC PARAMETERS

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

Table 5. DC PARAMETERS

Symbol	Pins	Parameter	Test Conditions	Min	Typ	Max	Unit
MOTORDRIVER							
$I_{MSmax,Peak}$	MOTXP MOTXN MOTYP MOTYN	Max current through motor coil in normal operation	V_{BB} = 14 V		800		mA
$I_{MSmax,RMS}$		Max RMS Current Through Coil in Normal Operation	V_{BB} = 14 V		570		mA
I_{MSabs}		Absolute Error on Coil Current (Note 6)	V_{BB} = 14 V	−10		10	%
I_{MSrel}		Matching of X and Y Coil Currents	V_{BB} = 14 V	−7	0	7	%
$R_{DS(on)}$		On Resistance for Each Motor Pin at I_{MSmax} (Note 7)	V_{BB} = 12 V, T_J = 50°C		0.50	1	Ω
			V_{BB} = 8 V, T_J = 50°C		0.55	1	Ω
			V_{BB} = 12 V, T_J = 150°C		0.70	1	Ω
			V_{BB} = 8 V, T_J = 150°C		0.85	1	Ω
I_{MSL}		Pull down current	HiZ Mode, V_{BB} = 7.7 V	0.4		2.2	mA
LIN TRANSMITTER							
I_{bus_off}	LIN	Dominant State, Driver Off	V_{bus} = 0 V, V_{BB} = 8 V and 18 V	−1			mA
I_{bus_off}		Recessive State, Driver Off	V_{bus} = V_{bat}, V_{BB} = 8 V and 18 V			20	μA
I_{bus_lim}		Current Limitation	V_{BB} = 8 V and 18 V	50	75	130	mA
R_{slave}		Pullup Resistance	V_{BB} = 8 V and 18 V	20	30	47	kΩ
LIN RECEIVER							
V_{bus_dom}	LIN	Receiver Dominant State	V_{BB} = 8 V and 18 V	0		0.4 * V_{BB}	V
V_{bus_rec}		Receiver Recessive State	V_{BB} = 8 V and 18 V	0.6 * V_{BB}		V_{BB}	V
V_{bus_hys}		Receiver Hysteresis	V_{BB} = 8 V and 18 V	0.05 * V_{BB}		0.175 * V_{BB}	V
THERMAL WARNING AND SHUTDOWN							
T_{tw}		Thermal warning		138	145	152	°C
T_{tsd}		Thermal shutdown (Notes 8 and 9)			T_{tw} + 10		°C
T_{low}		Low temperature warning (Note 9)			T_{tw} − 152		°C
SUPPLY AND VOLTAGE REGULATOR							
V_{BBOTP}	V_{BB}	Supply voltage for OTP zapping (Note 10)		9.0		10.0	V
UV_1		Stop voltage high threshold	Product versions with low UV; See Ordering Information	7.7	8.3	8.9	V
UV_2		Stop voltage low threshold		7.0	7.5	8.0	V
UV_1		Stop voltage high threshold	Product versions with high UV; See Ordering Information	8.8	9.3	9.8	V
UV_2		Stop voltage low threshold		8.1	8.5	8.9	V
I_{bat}		Total current consumption	Unloaded outputs V_{BB} = 29 V	1	3.50	10.0	mA
I_{bat_s}		Sleep mode current consumption	V_{BB} = 8 V and 18 V		40	100	μA
V_{DD}	V_{DD}	Regulated internal supply (Note 11)	8 V < V_{BB} < 29 V	4.75	5	5.25	V
$V_{DDReset}$		Digital supply reset level @ powerdown (Note 12)				4.5	V
I_{DDLim}		Current limitation	Pin shorted to ground V_{BB} = 14 V			40	mA

Table 5. DC PARAMETERS

Symbol	Pins	Parameter	Test Conditions	Min	Typ	Max	Unit
SWITCH INPUT AND HARDWIRE ADDRESS INPUT							
$Rt_{_OFF}$	SWI HW2	Switch OPEN Resistance (Note 13)		10			kΩ
$Rt_{_ON}$		Switch ON Resistance (Note 13)	Switch to GND or V_{BB}			2	kΩ
V_{BB_sw}		V_{BB} range for guaranteed operation of SWI and HW2		6		29	V
I_{lim_sw}		Current limitation	Short to GND or V_{bat} V_{BB} = 29 V			45	mA
HARDWIRED ADDRESS INPUTS AND TEST PIN							
V_{high}	HW0 HW1 TST	Input level high	V_{BB} = 14 V	0.7 * V_{DD}			V
V_{low}		Input level low	V_{BB} = 14 V			0.3 * V_{DD}	V
HW_{hyst}		Hysteresis	V_{BB} = 14 V	0.075 * V_{DD}			V
CHARGE PUMP							
V_{CP}	VCP	Output voltage	7 V < V_{BB} ≤ 14 V		2 * V_{BB} − 2.5		V
			14 V < V_{BB}	V_{BB} + 10		V_{BB} + 15	V
C_{buffer}		External buffer capacitor		220		470	nF
C_{pump}	CPP CPN	External pump capacitor		220		470	nF
PACKAGE THERMAL RESISTANCE VALUES							
Rth_{ja}	SO	Thermal resistance junction−to−ambient (2S2P)	Simulated conform JEDEC JES.D51		39		K/W
Rth_{jp}	SO	Thermal resistance junction−to−leads			19		K/W
Rth_{ja}	NQ	Thermal resistance junction−to−ambient (2S2P)			30		K/W
Rth_{jp}	NQ	Thermal resistance junction−to−leads and exposed pad			0.95		K/W

6. Tested in production for 800 mA, 400 mA, 200 mA and 100 mA current settings for both X and Y coil.
7. Based on characterization data.
8. No more than 100 cumulated hours in life time above T_{tw}.
9. Thermal shutdown and low temperature warning are derived from thermal warning. Guaranteed by design.
10. A buffer capacitor of minimum 100 μF is needed between V_{BB} and GND. Short connections to the power supply are recommended.
11. Pin V_{DD} must not be used for any external supply
12. The RAM content will not be altered above this voltage.
13. External resistance value seen from pin SWI or HW2, including 1 kΩ series resistor. For the switch OPEN, the maximum allowed leakage current is represented by a minimum resistance seen from the pin.

AC PARAMETERS

The AC parameters are guaranteed for temperature and V_{BB} in the operating range unless otherwise specified. The LIN transmitter and receiver physical layer parameters are compliant to LIN rev. 2.0 & 2.1.

Table 6. AC PARAMETERS

Symbol	Pins	Parameter	Test Conditions	Min	Typ	Max	Unit
POWERUP							
T_{pu}		Powerup Time	Guaranteed by Design			10	ms
INTERNAL OSCILLATOR							
f_{osc}		Frequency of Internal Oscillator	V_{BB} = 14 V	3.6	4.0	4.4	MHz
LIN TRANSMITTER CHARACTERISTICS ACCORDING TO LIN V2.0 & V2.1							
D1	LIN	Duty Cycle 1 = $t_{Bus_rec(min)}$/ (2 x t_{Bit}); See Figure 5	THRec(max)= 0.744 x V_{BB} THDom(max)= 0.581 x V_{BB}; V_{BB} = 7.0 V...18 V; t_{Bit} = 50 μs	0.396			
D2		Duty Cycle 2 = $t_{Bus_rec(max)}$/ (2 x t_{Bit}); See Figure 5	THRec(min)= 0.284 x V_{BB} THDom(min)= 0.422 x V_{BB}; V_{BB} = 7.6 V...18 V; t_{Bit} = 50 μs			0.581	
LIN RECEIVER CHARACTERISTICS ACCORDING TO LIN V2.0 & V2.1							
trx_pdr	LIN	Propagation delay bus dominant to RxD = Low	V_{BB} = 7.0 V & 18 V; See Figure 5			6	μs
trx_pdf		Propagation delay bus recessive to RxD = High	V_{BB} = 7.0 V & 18 V; See Figure 5			6	μs
trx_sym		Symmetry of receiver propagation delay	trx_pdr − trx_pdf	−2		+2	μs
SWITCH INPUT AND HARDWIRE ADDRESS INPUT							
T_{sw}	SWI HW2	Scan pulse period (Note 14)	V_{BB} = 14 V		1024		μs
T_{sw_on}		Scan pulse duration (Note 14)	V_{BB} = 14 V		64		μs
MOTORDRIVER							
F_{pwm}	MOTxx	PWM frequency (Note 14)		18	20	22.0	kHz
T_{brise}		Turn−on transient time	Between 10% and 90% V_{BB} = 14 V		150		ns
T_{bfall}		Turn−off transient time			140		ns
T_{stab}		Run current stabilization time (Note 14)			1/Vmin		s
CHARGE PUMP							
f_{CP}	CPN CPP	Charge pump frequency (Note 14)	V_{BB} = 14 V		250		kHz

14. Derived from the internal oscillator

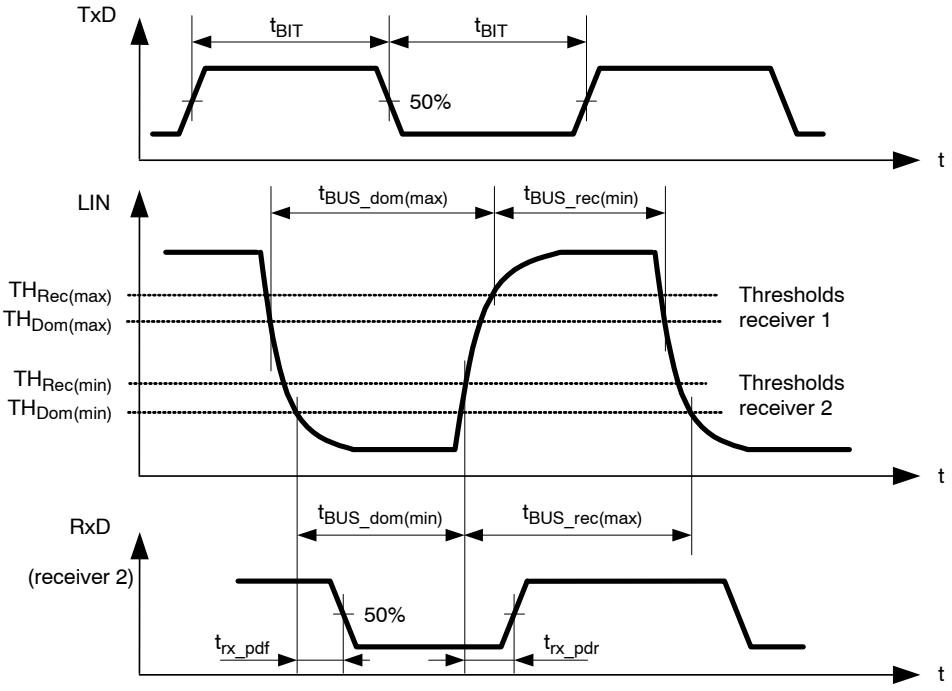


Figure 5. Timing Diagram for AC Characteristics According to LIN 2.0 & 2.1

TYPICAL APPLICATION

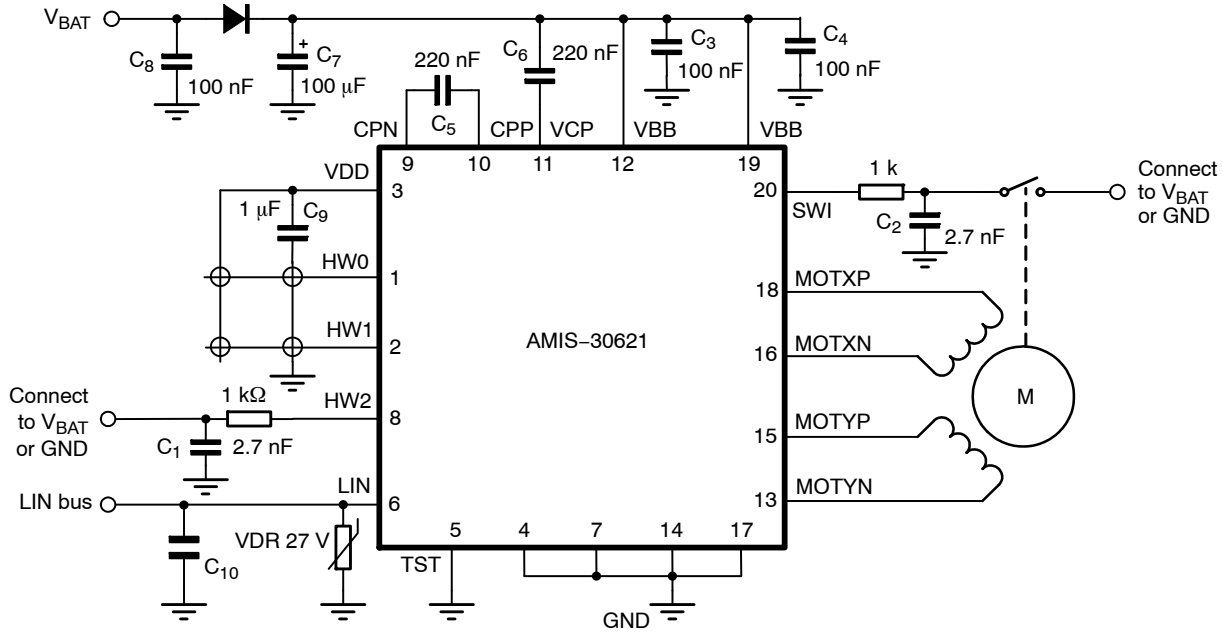


Figure 6. Typical Application Diagram for SO device.

1. All resistors are ±5%, 1/4 W
2. C_1, C_2 minimum value is 2.7 nF, maximum value is 10 nF
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 V_{BB} and GND
5. C_5 and C_6 must be as close as possible to pins CPN, CPP, VCP, and V_{BB} to reduce EMC radiation
6. C_9 must be a ceramic capacitor to assure low ESR
7. C_{10} is placed for EMC reasons; value depends on EMC requirements of the application

POSITIONING PARAMETERS

Stepping Modes

One of four possible stepping modes can be programmed:

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

Maximum Velocity

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

The accuracy of Vmax is derived from the internal oscillator. Under special circumstances it is possible to change the V_{max} 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

Minimum Velocity

Once the maximum velocity is chosen, 16 possible values can be programmed for the minimum velocity Vmin. The table below provides the obtainable values in full−step/s. The accuracy of Vmin is derived from the internal oscillator.

Table 8. OBTAINABLE VALUES IN FULL−STEP/S FOR THE MINIMUM VELOCITY

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

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

Acceleration and Deceleration

Sixteen possible values can be programmed for Acc (acceleration and deceleration between Vmin and Vmax). The table below provides the obtainable values in full−step/s^2. One observes restrictions for some combinations of acceleration index and maximum speed (gray cells).

The accuracy of Acc is derived from the internal oscillator.

Table 9. ACCELERATION AND DECELERATION SELECTION TABLE

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

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

$$Nstep = \frac{V\,max^2 - V\,min^2}{2 \times Acc}$$

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 the table below. 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: LSB: Least Significant Bit
S: Sign bit, two's complement

Position Ranges

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

Table 11. POSITION RANGE

Command	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.

Secure Position

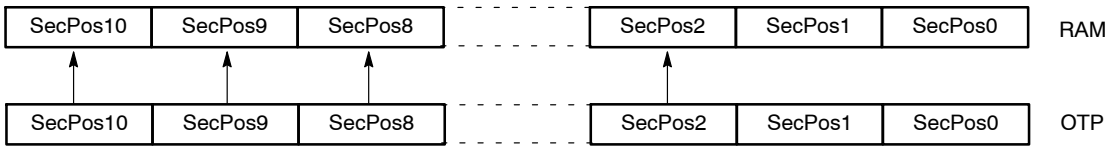
A secure position can be programmed. It is coded in 11−bits, thus having a lower resolution than normal positions, as shown in the following table. See also command `GotoSecurePosition` and LIN lost behavior.

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



Shaft

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

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

STRUCTURAL DESCRIPTION

See also the Block Diagram in Figure 1.

Stepper Motordriver

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 two separated coils. Each coil (X and Y) is driven by one H−bridge, and the driver controls the currents flowing through the coils. The rotational position of the rotor, in unloaded condition, is defined by the ratio of current flowing in X and Y. The torque of the stepper motor when unloaded is controlled by the magnitude of the currents in X and Y.
- The control block for the H−bridges, including the PWM control, the synchronous rectification and the internal current sensing circuitry.

- The charge pump to allow driving of the H−bridges' high side transistors.
- Two pre−scale 4−bit DAC's to set the maximum magnitude of the current through X and Y.
- Two DAC's to set the correct current ratio through X and Y.

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

Control Logic (Position Controller and Main Control)

The control logic block stores the information provided by the 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.

LIN Interface

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

Miscellaneous

The 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 5 V regulator (from the battery supply) to supply the internal logic circuitry.

FUNCTIONS DESCRIPTION

This chapter describes the following functional blocks in more detail:

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

The LIN controller is discussed in a separate chapter.

Position Controller

Positioning and Motion Control

A positioning command will produce a motion as illustrated in Figure 7. A motion starts with an acceleration phase from minimum velocity (Vmin) to maximum velocity (Vmax) and ends with a symmetrical deceleration. This is defined by the control logic according to the position required by the application and the parameters programmed by the application during the configuration phase. The current in the coils is also programmable.

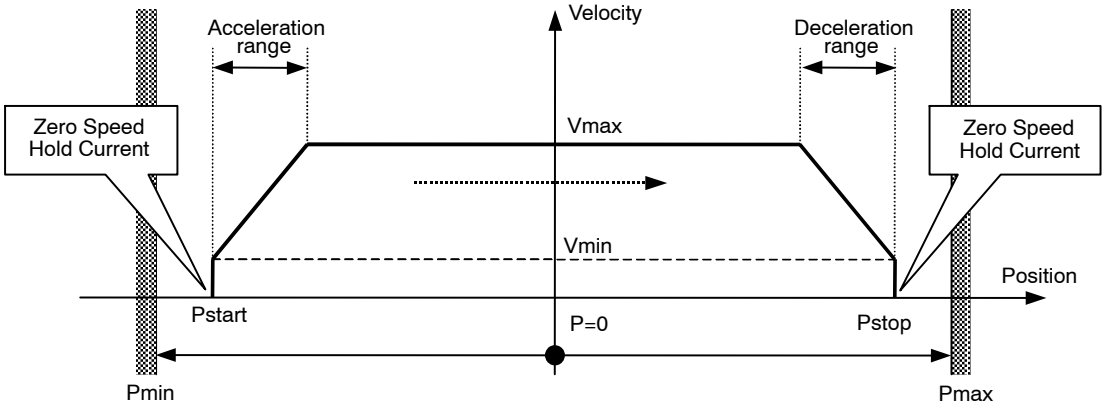


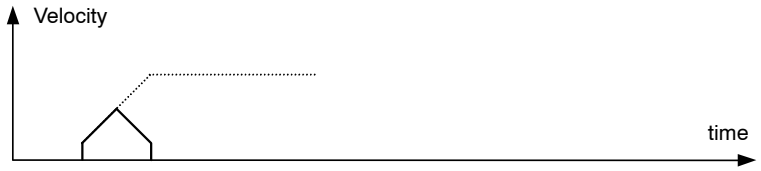
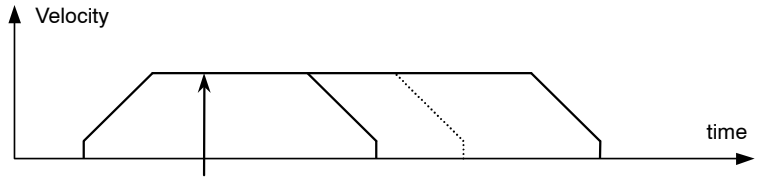
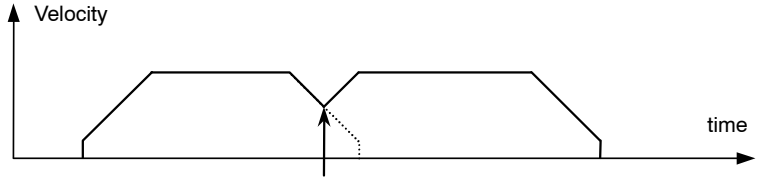
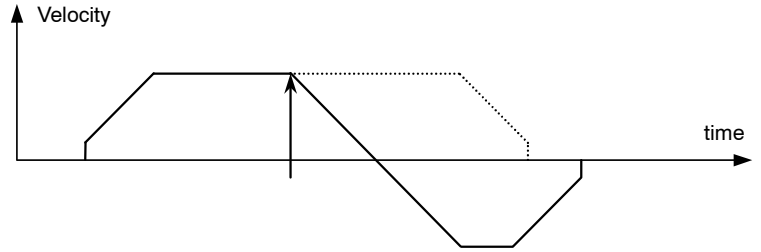
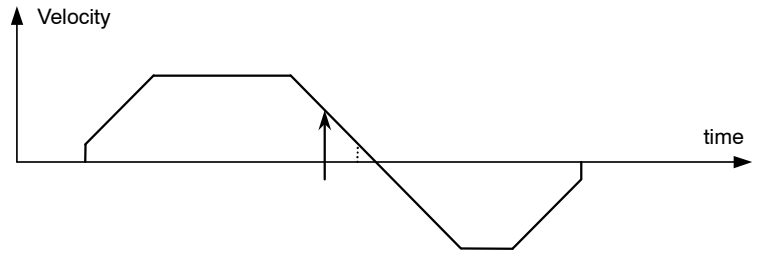
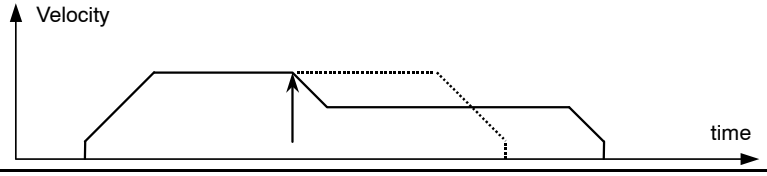
Figure 7. Positioning and Motion Control

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 below.

Table 14. POSITIONING EXAMPLES

Short motion.	
New positioning command in same direction, shorter or longer, while a motion is running at maximum velocity.	
New positioning command in same direction while in deceleration phase (Note 15) 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.	

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

Dual Positioning

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

Then a second 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.

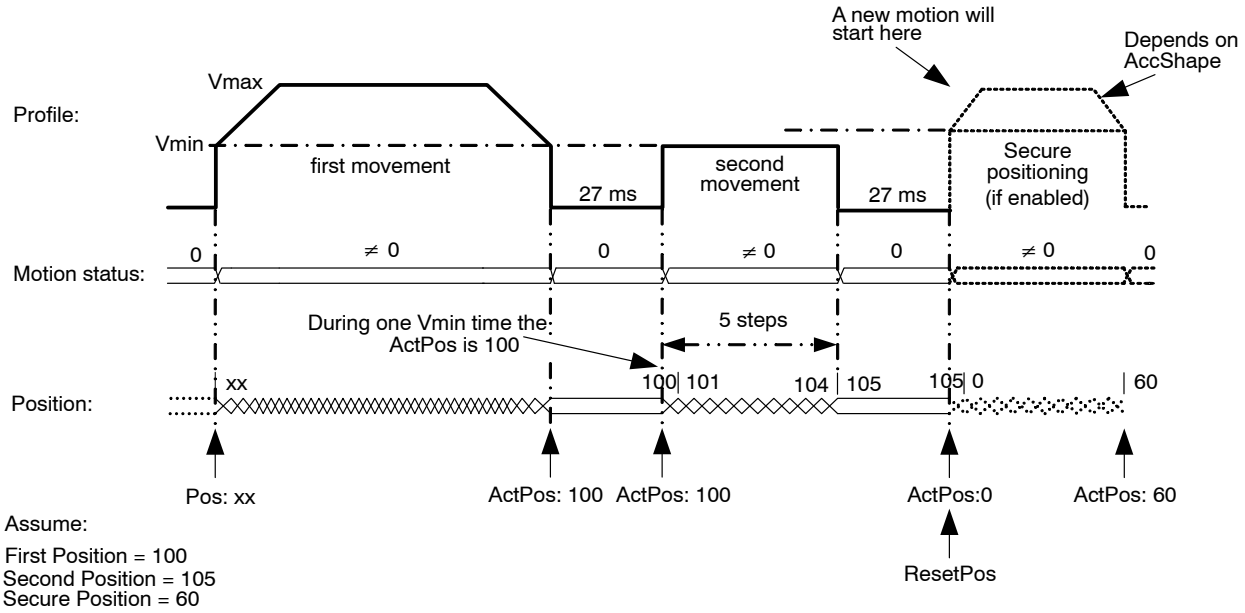


Figure 8. Dual Positioning

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. After dual positioning is executed the internal flag “Reference done” is set.

1. The **priority encoder** is describing the management of states and commands.
2. If a SetPosition(Short) command issued during a DualPosition sequence, it will be kept in position buffer memory and executed afterwards. This applies also for the commands sleep, SetMotorParam and GotoSecurePosition.
3. 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.
4. A DualPosition sequence starts by setting TagPos buffer register to SecPos value, provided secure position is enabled otherwise TagPos is reset to zero.
5. 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.
6. 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).
7. Commands ResetPosition, SetDualPosition and SoftStop will be ignored while a DualPosition sequence is ongoing, and will not be executed afterwards.
8. A SetMotorParam command should not be sent during a SetDualPosition sequence.
9. 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.

Position Periodicity

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

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

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

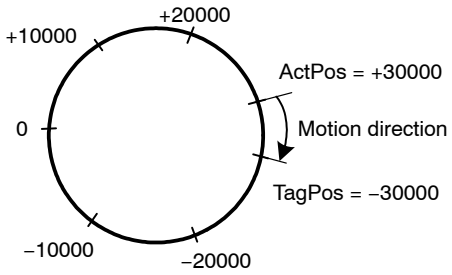


Figure 9. Motion Direction is Function of Difference between ActPos and TagPos

Hardwired Address HW2

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

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

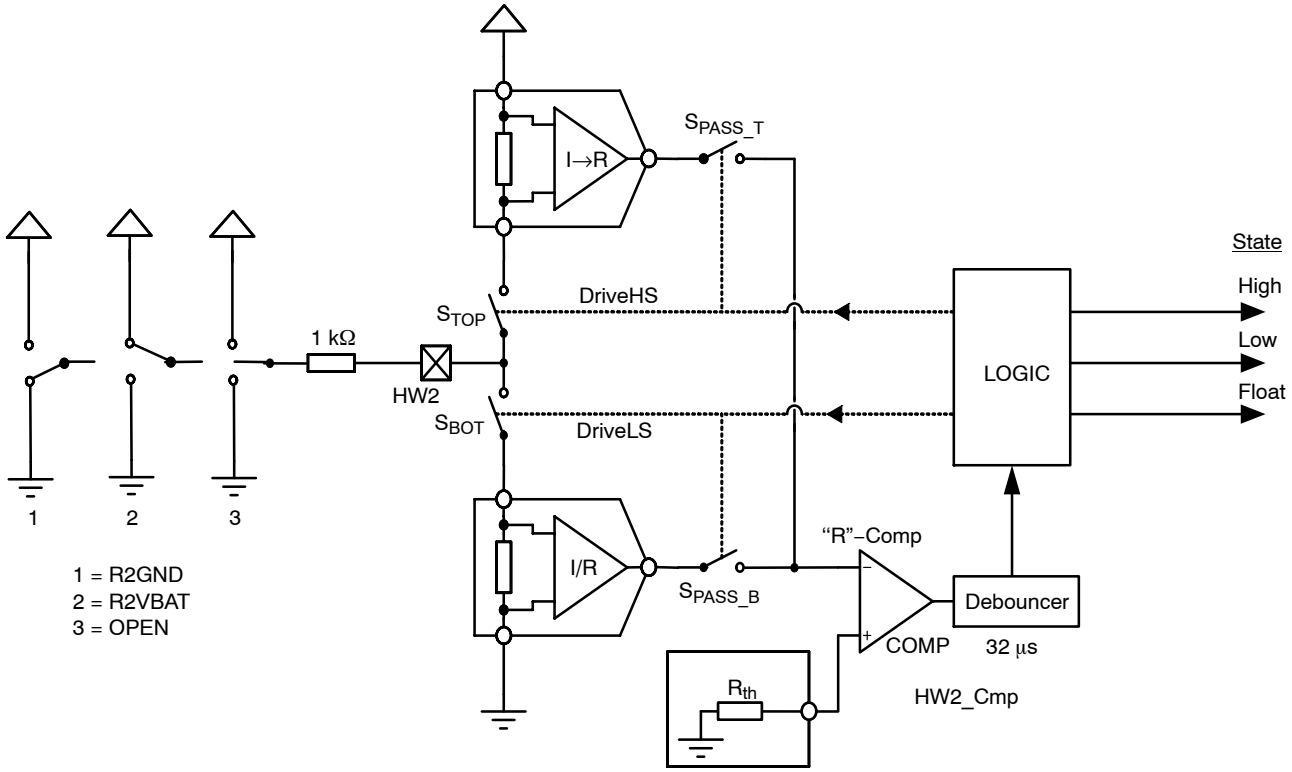


Figure 10. Simplified Schematic Diagram of the HW2 Comparator

3 cases can be distinguished (see also Figure 10 above):

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

Table 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 32 μs debounced HW2_Cmp output accordingly. The output of this small state−machine is corresponding to:

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

As illustrated in the table above (Table 15), the state is depending on the previous state, the condition of the 2 switch controls (DriveLS and DriveHS) and the output of HW2_Cmp. The figure below is showing an example of a practical case where a connection to VBAT is interrupted.

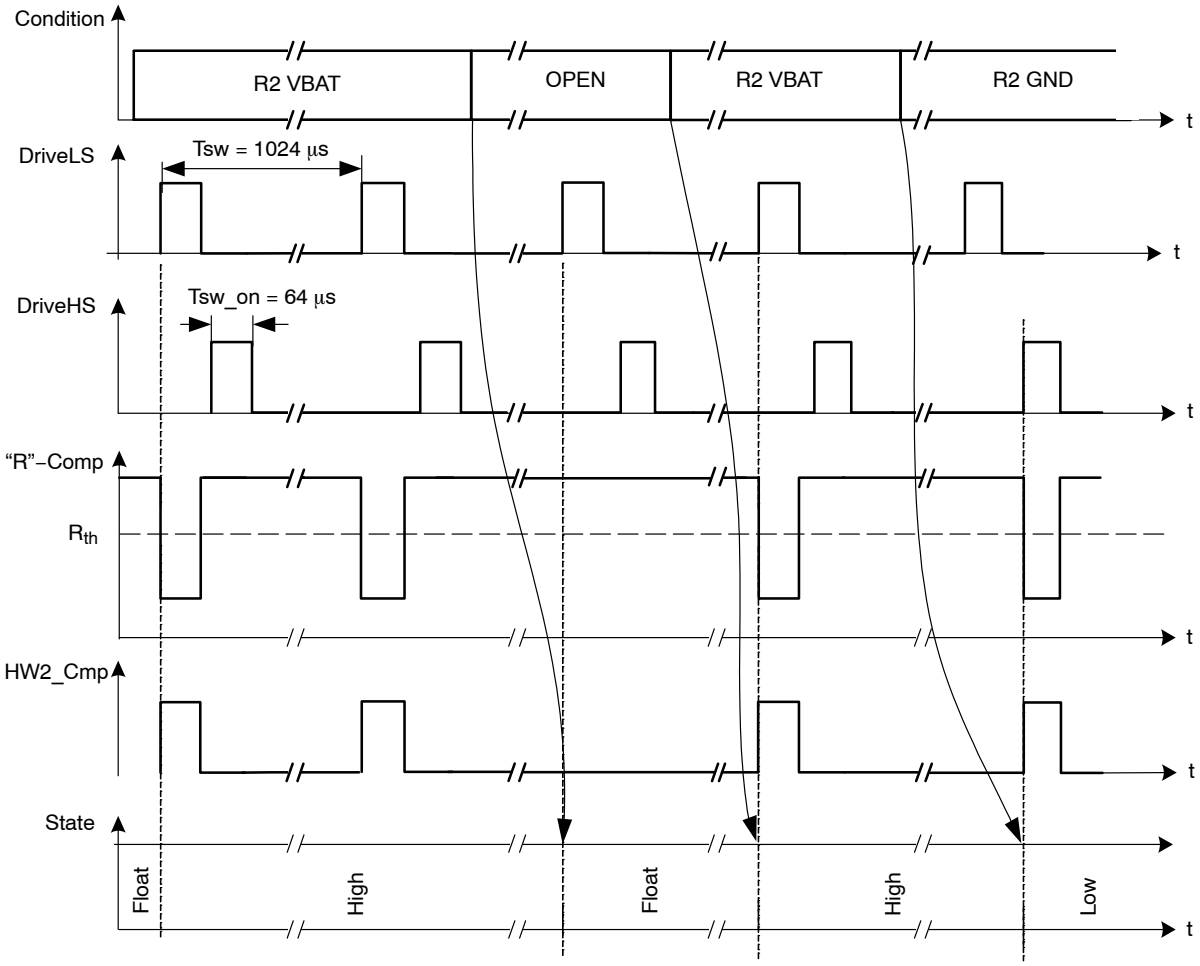


Figure 11. Timing Diagram Showing the Change in States for HW2 Comparator

R2VBAT

A resistor is connected between VBAT and HW2. Every 1024 μs S_{BOT} is closed and a current is sensed. The output of the I → R converter is low and the HW2_Cmp output is high. Assuming the previous state was floating, the internal logic will interpret this as a change of state and the new state will be high (see also Table 15). The next time S_{BOT} is closed the same conditions are observed. The previous state was high, so based on Table 15 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 15 one can see that the state changes to float. This will trigger a motion to secure position.

R2GND

If a resistor is connected between HW2 and the GND, a current is sensed every 1024 μs when S_{TOP} is closed. The output of the top I → R converter is low and as a result the HW2_Cmp output switches to high. Again based on the stated diagram in Table 15 one can see that the state will change to Low. This low state will be interpreted as HW2 address = 0.

External Switch SWI

As illustrated in Figure 12 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.

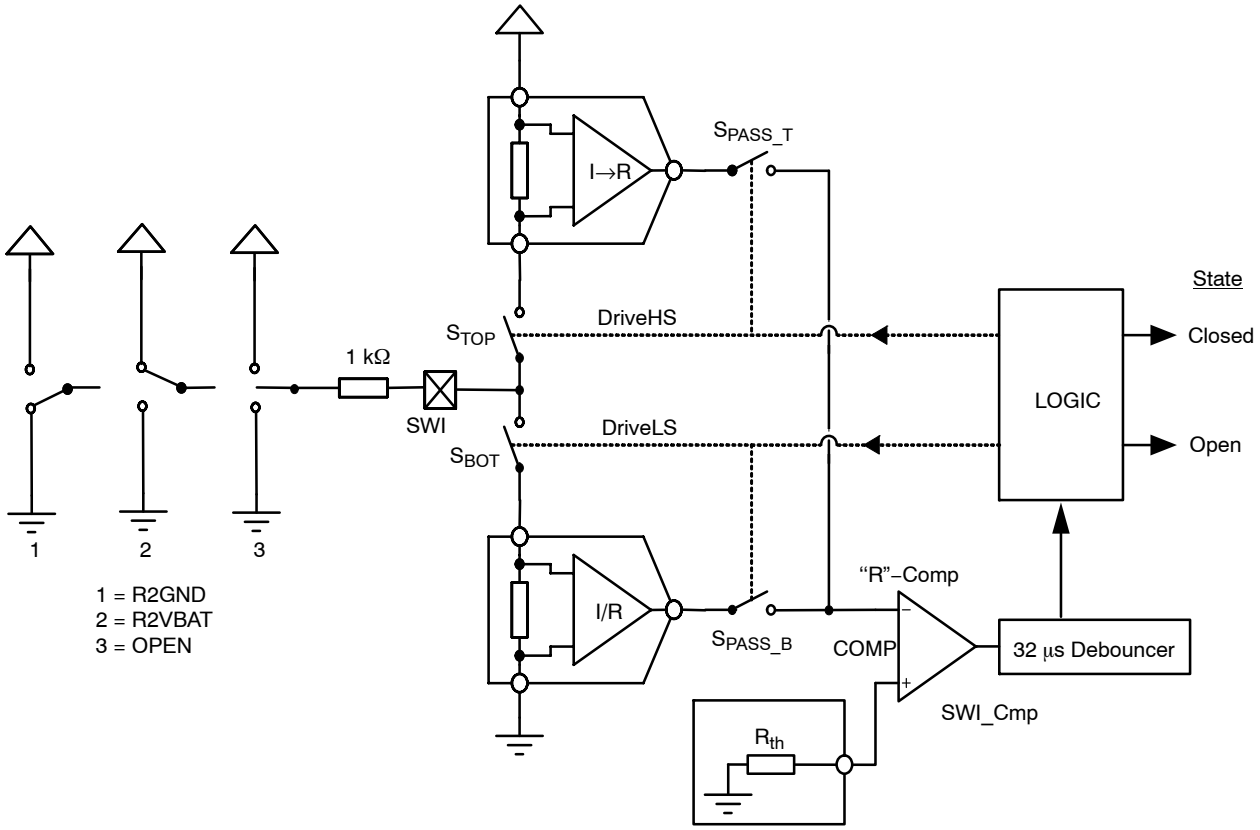


Figure 12. Simplified Schematic Diagram of the SWI Comparator

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

The GetActualPos command reads back the <ActPos> register and the status of ESW. In this way the master node may get synchronous information about the state of the switch together with the position of the motor. See Table 16 below.

Table 16. GetActualPos LIN COMMAND

Reading Frame									
		Structure							
Byte	Content	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	*	*	1	0	ID3	ID2	ID1	ID0
1	Data 1	ESW	AD[6:0]						
2	Data 2	ActPos[15:8]							
3	Data 3	ActPos[7:0]							
4	Data 4	VddReset	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	
5	Checksum	Checksum over data							

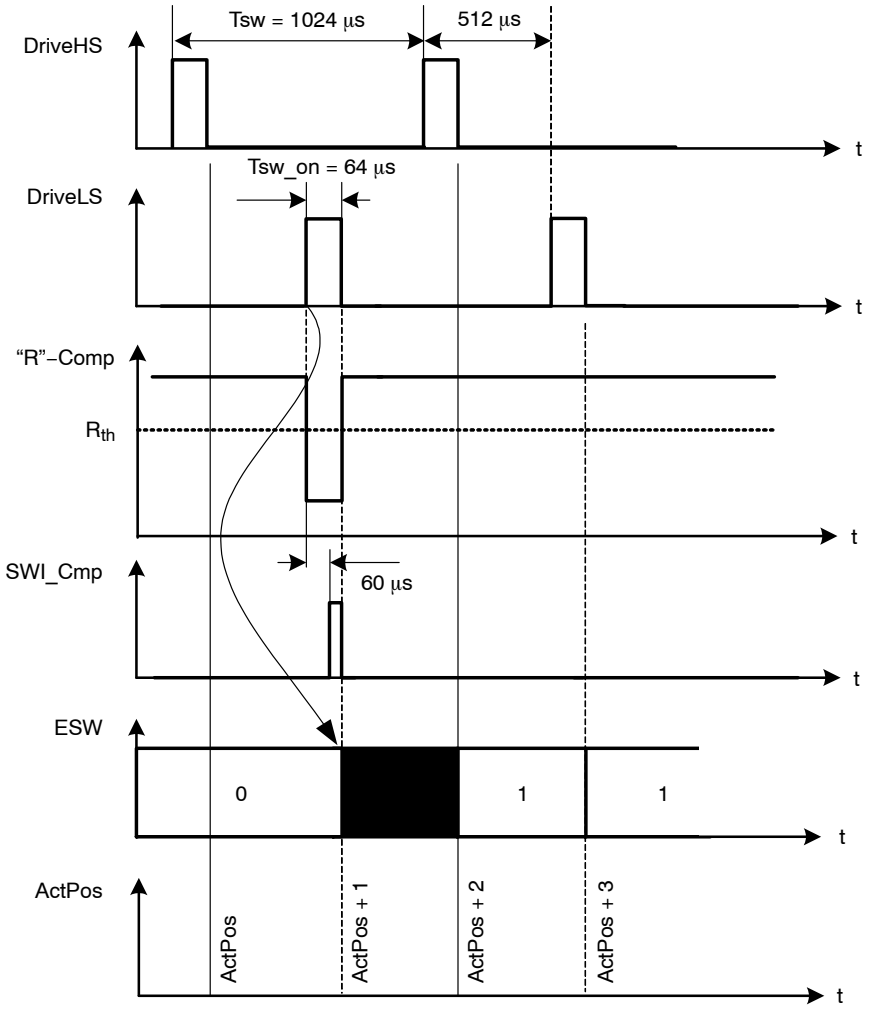


Figure 13. Simplified Timing Diagram Showing the Change in States for SWI Comparator

Main Control and Register, OTP memory + ROM

Power−up Phase

Power up phase of the AMIS−30621 will not exceed 10ms. After this phase, the AMIS−30621 is in standby mode, ready to receive LIN messages and execute the associated commands. After power−up, the registers and flags are in the reset state, while some of them are being loaded with the OTP memory content (see Table 19).

Reset

After power−up, or after a reset occurrence (e.g. a micro−cut on pin V_{BB} has made V_{DD} to go below $V_{DDReset}$ level), the H−bridges will be in high−impedance mode, and the registers and flags will have a predetermined value. This is documented in Tables 19 and 20.

Soft Stop

A soft stop is an immediate interruption of a motion, but with a deceleration phase. At the end of this action, the register `<TagPos>` is loaded with the value contained in register `<ActPos>`, see Table 19). The circuit is then ready to execute a new positioning command, provided thermal and electrical conditions allow for it.

Sleep Mode

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

Sleep mode can be entered in the following cases:

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

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

Thermal Shutdown Mode

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

Temperature Management

The AMIS−30621 monitors temperature by means of two thresholds and one shutdown level, as illustrated in the state diagram and illustration of Figure 14 below. The only condition to reset flags <TW> and <TSD> (respectively thermal warning and thermal shutdown) is to be at a temperature lower than T_{tw} and to get the occurrence of a GetStatus or a GetFullStatus LIN frame.

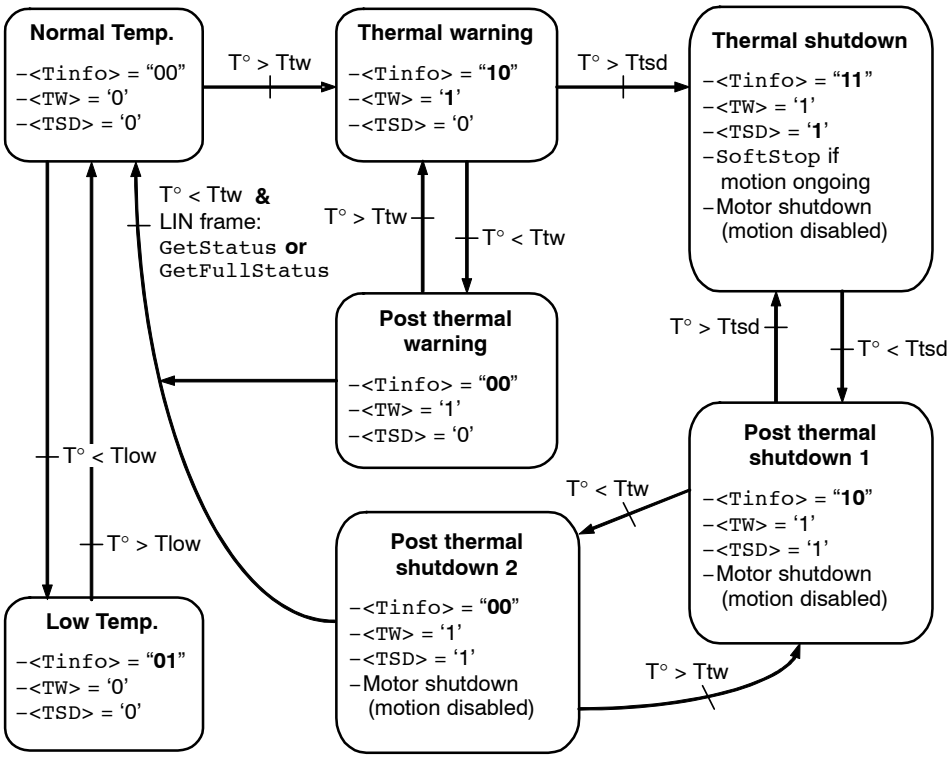


Figure 14. State Diagram Temperature Management

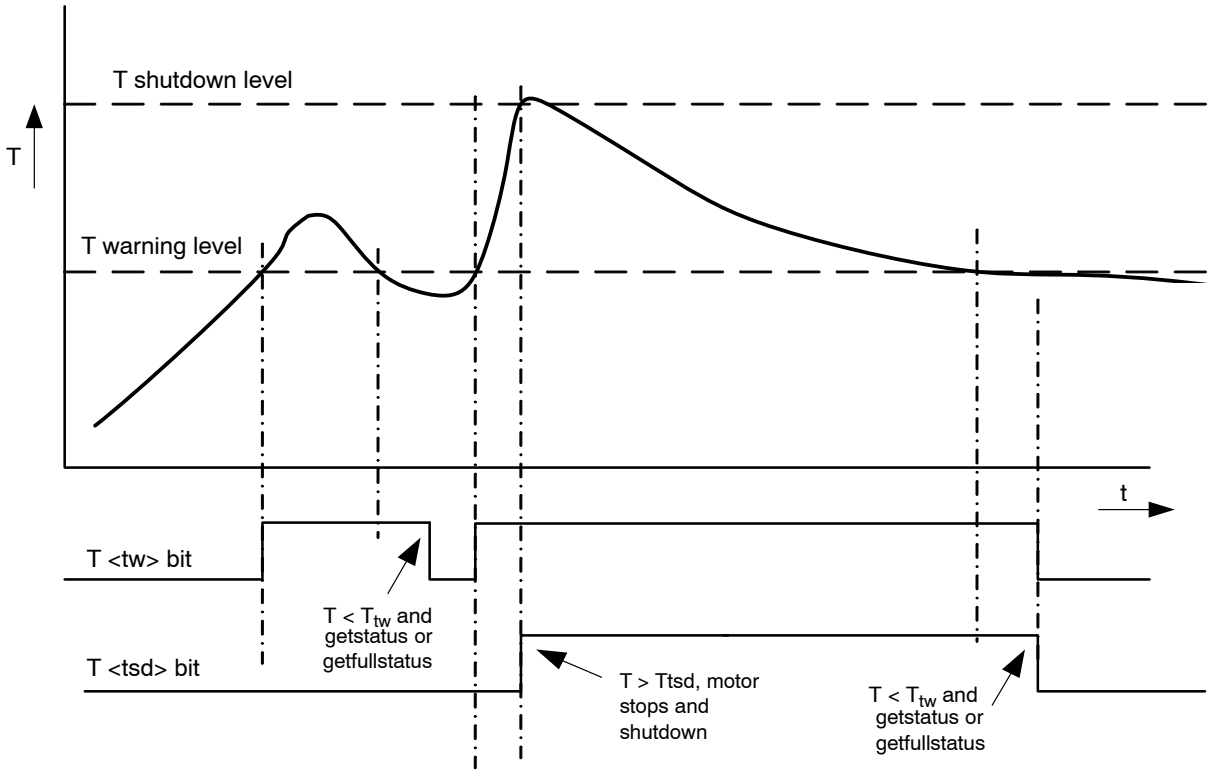


Figure 15. Illustration of Thermal Management Situation

Battery Voltage Management

The AMIS−30621 monitors the battery voltage by means of one threshold and one shutdown level. The only condition to reset flags <UV2> and <StepLoss> is to recover by a battery voltage higher than UV1 and to receive a GetStatus or a GetFullStatus command.

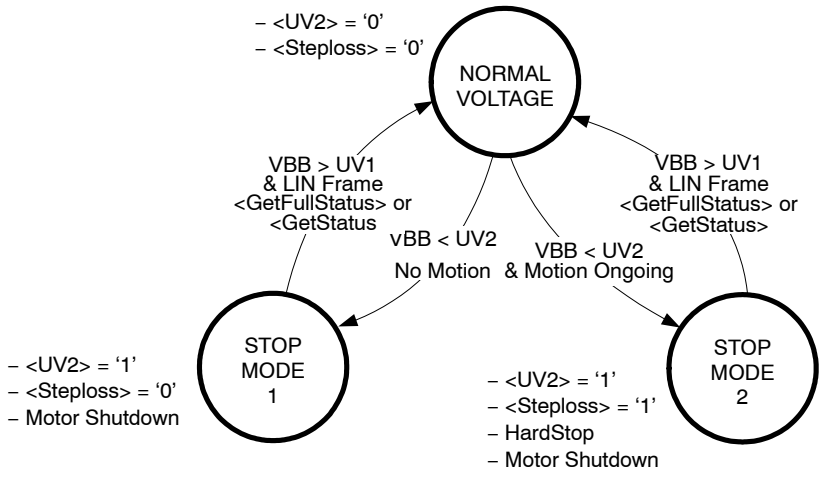


Figure 16. State Diagram Battery Voltage Management

In **Stop mode 1** the motor is put in shutdown state. The <UV2> **flag** is set. In case V_{BB} > 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 V_{BB} > 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 V_{DD} < V_{DD}Reset (digital POR level).

OTP Register

OTP Memory Structure

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

Table 17. OTP MEMORY STRUCTURE

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	OSC3	OSC2	OSC1	OSC0	IREF3	IREF2	IREF1	IREF0
0x01	1	TSD2	TSD1	TSD0	BG3	BG2	BG1	BG0
0x02	ADM	(HW2) (Note 16)	(HW1) (Note 16)	(HW0) (Note 16)	PA3	PA2	PA1	PA0
0x03	Irun3	Irun2	Irun1	Irun0	Ihold3	Ihold2	Ihold1	Ihold0 (Note 17)
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

16. Although not stored in the OTP memory the physical status of the hardware address input pins are returned by a read of the OTP contents (GetOTPparam).
17. Note for product version AMIS30621C6217G and AMIS30621C6217RG the Ihold0 bit is programmed to '1'.

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

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

Table 18. OTP OVERWRITE PROTECTION

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

The command used to load the application parameters via the 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] and ADM bit, 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.

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	800

Note: When the motor is stopped, the current is reduced from <IRUN> to <IHOLD>.

StepMode Setting of step modes.

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

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

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−Steps^2)
0	0	0	0	0	49*
1	0	0	0	1	218*
2	0	0	1	0	1004
3	0	0	1	1	3609
4	0	1	0	0	6228
5	0	1	0	1	8848
6	0	1	1	0	11409
7	0	1	1	1	13970
8	1	0	0	0	16531
9	1	0	0	1	19092*
A	1	0	1	0	21886*
B	1	0	1	1	24447*
C	1	1	0	0	27008*
D	1	1	0	1	29570*
E	1	1	1	0	34925*
F	1	1	1	1	40047*

*restriction on speed

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”, secure positioning is disabled; 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’.

Table 19. RAM REGISTERS

Register	Mnemonic	Length (bit)	Related commands	Comment	Reset State
Actual position	ActPos	16	GetActualPos GetFullStatus GotoSecurePos ResetPosition	16−bit signed	Note 18
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 Minimum Velocity See look−up table Vmin	
Maximum Velocity	Vmax	4	GetFullStatus ResetToDefault SetMotorParam	See Section Maximum Velocity See look−up table Vmax	
Shaft	Shaft	1	GetFullStatus ResetToDefault SetMotorParam	Direction of movement	
Acceleration/ deceleration	Acc	4	GetFullStatus ResetToDefault SetMotorParam	See Section Acceleration See look−up table Acc	
Secure Position	SecPos	11	GetFullStatus ResetToDefault SetMotorParam	Target position when LIN connection fails; 11 MSB's of 16−bit position (LSB's fixed to '0')	
Stepping mode	StepMode	2	GetFullStatus ResetToDefault SetMotorParam SetPositionShort	See Section Stepping Modes See look−up table StepMode	

18. 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.

Table 20. 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 Resets only after GetFullStatus	'0'
Electrical defect	ElDef	1	GetActualPos GetStatus GetFullStatus	<OVC1> or <OVC2> or 'open−load on coil X' or 'open−load on coil Y' 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 (CW) "010" = inner motion deceleration (CW) "011" = inner motion max. speed (CW) "101" = outer motion acceleration (CCW) "110" = outer motion deceleration (CCW) "111" = outer motion max. speed (CCW)	"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 ($T_j > T_{tsd}$) Resets only after Get(Full)Status and if <Tinfo> = "00"	'0'
Thermal warning	TW	1	GetActualPos GetStatus GetFullStatus	'1' = over temperature ($T_j > T_{tw}$) Resets only after Get(Full)Status and if <Tinfo> = "00"	'0'
Battery stop voltage	UV2	1	GetActualPos GetStatus GetFullStatus	'0' = V_{BB} > UV2 '1' = $V_{BB} \leq$ UV2 Resets only after Get(Full)Status	'0'
Digital supply reset	V_{DD}Reset	1	GetActualPos GetStatus GetFullStatus	Set at '1' after power 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 Get(Full)Status command.	'1'

Priority Encoder

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

Table 21. PRIORITY ENCODER

State →	Stopped	GotoPos	DualPosition	SoftStop	HardStop	ShutDown	Sleep
Command ↓	**Motor Stopped, Ihold in Coils**	**Motor Motion On-going**	**No Influence on RAM and TagPos**	**Motor Decelerating**	**Motor Forced to Stop**	**Motor Stopped, H−bridges in Hi−Z**	**No Power (Note 19)**
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 21)	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 27	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 25)

NOTE: See table notes on the following page.

19. Leaving sleep state is equivalent to POR.
20. After POR, the shutdown state is entered. The shutdown state can only be left after GetFullStatus command (so that the master could read the <V_{DD}Reset> flag).
21. 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.
22. 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).
23. Shutdown state can be left only when <TSD> and <HS> flags are reset.
24. 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).
25. 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.
26. Some transitions like GotoPos → sleep are actually done via several states: GotoPos → SoftStop → Stopped → Sleep (see diagram below).
27. 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
28. <SecEn> = '1' when register SecPos is loaded with a value different from the most negative value (i.e. different from 0x400 = "100 0000 0000")
29. <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.
30. Command for dynamic assignment of Ids is decoded in all states except sleep and has not effect on the current state.
31. 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.
32. 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

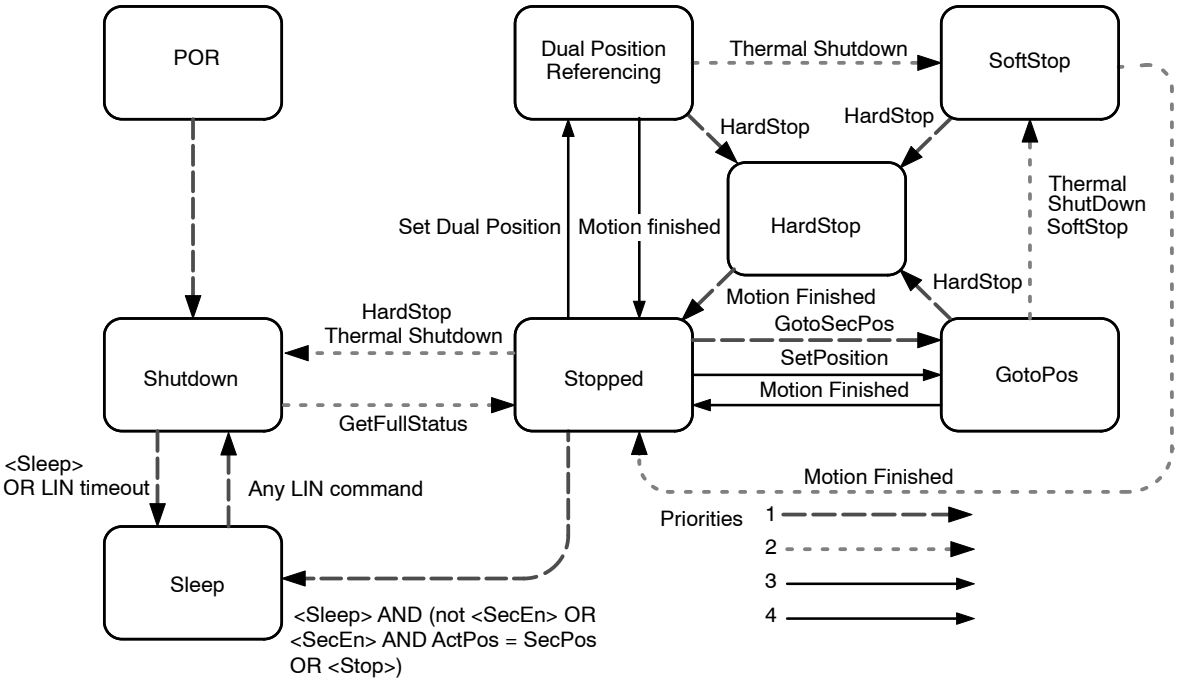


Figure 17. Simplified State Diagram

Motordriver

Current Waveforms in the Coils

Figure 18 below illustrates the current fed to the motor coils by the motor driver in half−step mode.

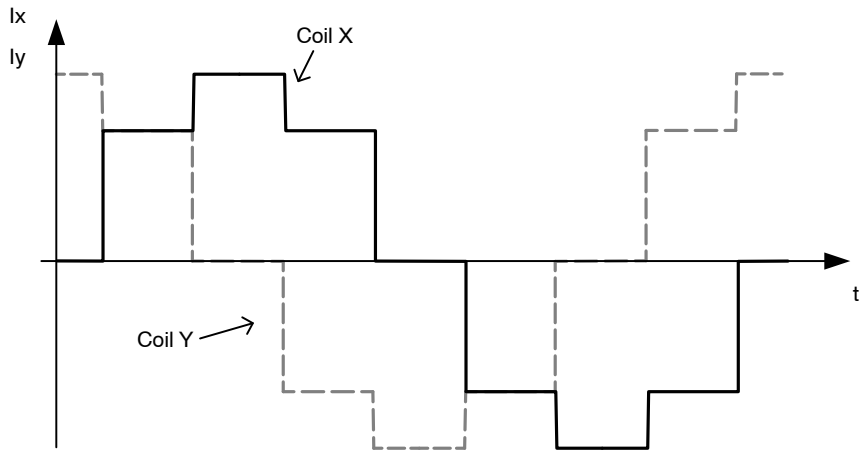


Figure 18. Current Waveforms in Motor Coils X and Y in Halfstep Mode

Whereas Figure 19 below shows the current fed to the coils in 1/16th micro stepping (1 electrical period).

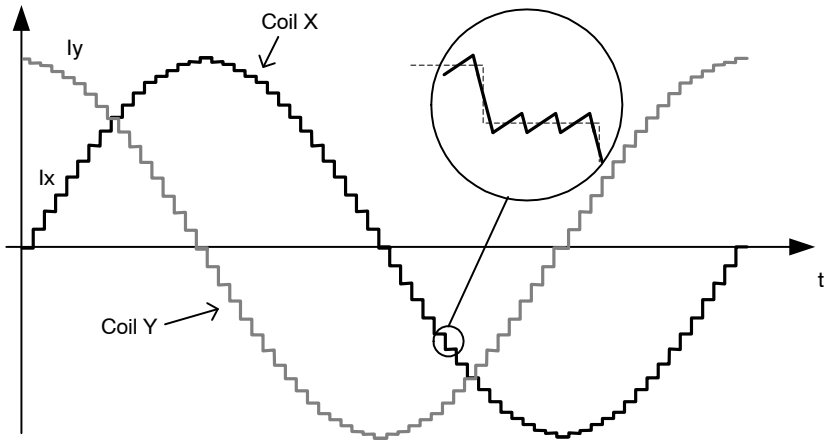


Figure 19. Current Waveforms in Motor Coils X and Y in 1/16th Micro−Step Mode

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 19 above shows how the PWM circuit performs this regulation.

Motor Starting Phase

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

Motor Stopping Phase

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

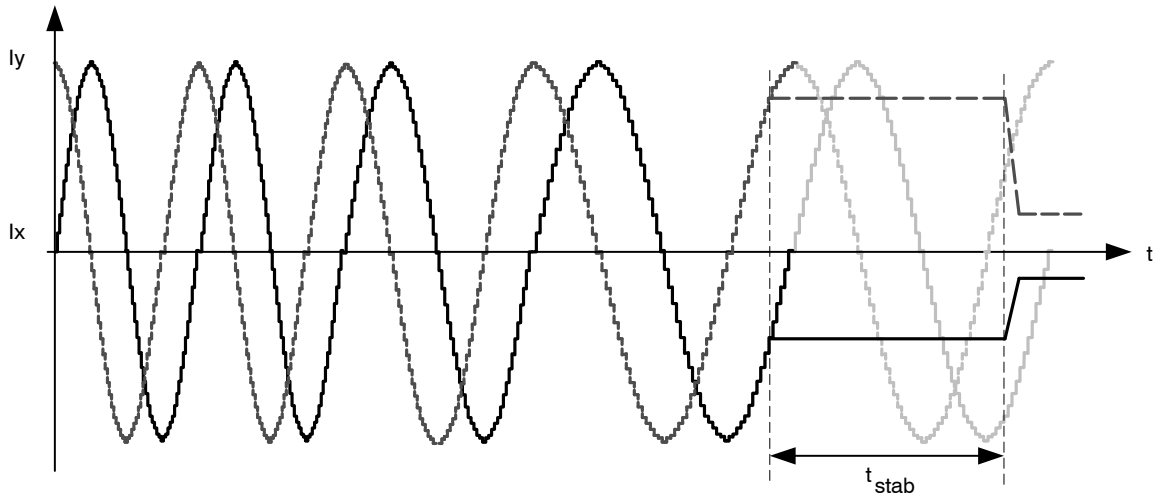


Figure 20. Motor Stopping Phase

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.

Electrical Defect on Coils, Detection and Confirmation

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

This allows the detection the following short circuits:

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

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

Table 22. ELECTRICAL DEFECT DETECTION

Pins	Fault mode
Yi or Xi	Short circuit to GND
Yi or Xi	Short circuit to Vbat
Yi or Xi	Open
Y1 and Y2	Short circuited
X1 and X2	Short circuited
Xi and Yi	Short circuited

Motor Shutdown Mode

A motor shutdown occurs when:

- The chip temperature rises above the thermal shutdown threshold Ttsd (see Thermal Shutdown Mode).
- The battery voltage goes below UV2 (see Battery Voltage Management).
- The charge pump voltage goes below the charge pump comparator level Flag <CPFail> = ‘1’, meaning there is a charge pump failure.
- Flag <ElDef> = ‘1’, meaning an electrical problem is detected on one or both coils, e.g. a short circuit.

A motor shutdown leads to the following:

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

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

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

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

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

Table 23. EXAMPLE OF POSSIBLE SEQUENCE USED TO DETECT AND DETERMINE CAUSE OF MOTOR SHUTDOWN

$T_J \geq$ Tsd or $V_{BB} \leq$ UV2 or <ElDef> = '1' or <CPFail> = '1' ↓	SetPosition frame ↓	GetFullStatus or GetStatus frame ↓	GetFullStatus or GetStatus frame ↓...
− The circuit is driven in motor shutdown mode − The application is not aware of this	− The position set−point is updated by the LIN Master − Motor shutdown mode ⇒ no motion − The application is still unaware	− The application is aware of a problem	− Possible confirmation of the problem
		− Reset <TW> or <TSD> or <UV2> or <StepLoss> or <ElDef> or <CPFail> by the application − Possible new detection of over temperature or low voltage or electrical problem ⇒ Circuit sets <TW> or <TSD> or <UV2> or <StepLoss> or <ElDef> or <CPFail> again at '1'	

Important: While in shutdown mode, since there is no hold current in the coils, the mechanical load can cause a step loss, which indeed cannot be flagged by the AMIS−30621.

If the LIN communication is lost while in shutdown mode, the circuit enters the sleep mode immediately (Note 1).

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.

Note 1: The Priority Encoder is describing the management of states and commands.

LIN CONTROLLER

General Description

The LIN (local interconnect network) is a serial communications protocol that efficiently supports the control of mechatronics nodes in distributed automotive applications. The physical interface implemented in the AMIS−30621 is compliant to the LIN rev. 2.0 & 2.1 specifications. It features a slave node, thus allowing for:

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

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

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.

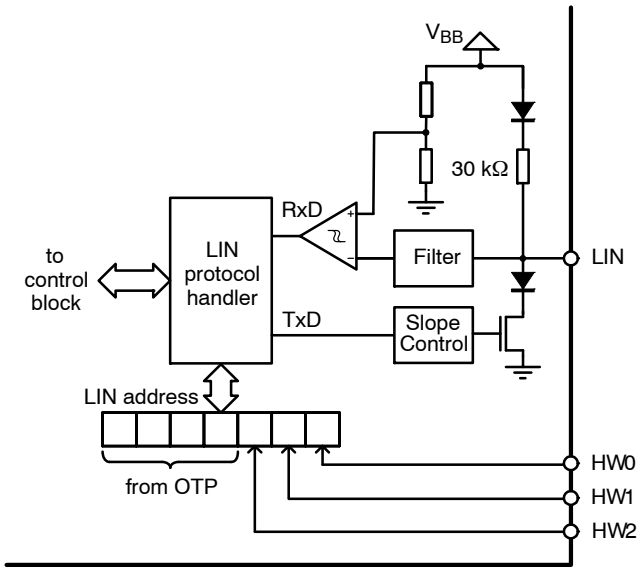


Figure 21. LIN Interface

Slave Operational Range for Proper Self Synchronization

The LIN interface will synchronize properly in the following conditions:

- Vbat ≥ 8 V V_{BB} ≥ 7.3 V
- Ground shift between master node and slave node < ±1 V

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

Functional Description

Analog Part

The transmitter is a low−side driver with a pull−up resistor and slope control. The receiver mainly consists of a comparator with a threshold equal to V_{BB}/2. Figure 5 shows the characteristics of the transmitted and received signal. See AC Parameters for timing values.

Protocol Handler

This block implements:

- Bit Synchronization
- Bit Timing
- The MAC Layer
- The LLC Layer
- The Supervisor

Error Status Register

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

Table 24. 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 sent and bit monitored on the LIN bus

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

Physical Address of the Circuit

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

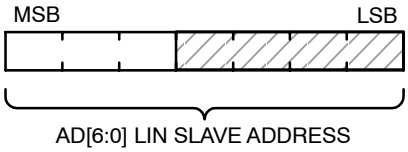


Figure 22. 7−bit LIN Address

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 23. The ADMbit allows the use of “SetPositionShort”. This give coverage for slaves with different PA3 // HW2 addresses which are attached to the same LIN bus.

The physical address AD[6:0] is a combination of four OTP memory bits PA[3:0] from the OTP Memory Structure and the hardwired address bits HW[2:0]. Depending on the addressing mode (ADM –bit in OTP Memory Structure) the combination is as illustrated in Figure 23.

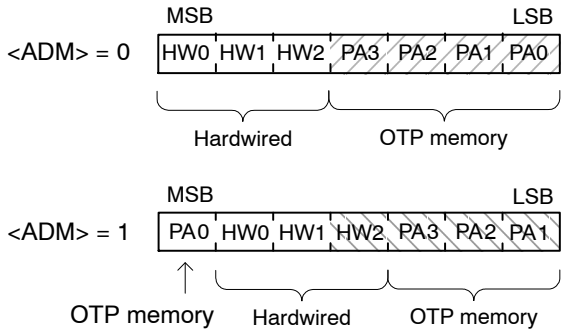


Figure 23. Combination of OTP and Hardwired Address Bits in Function of ADM

Note: Pins HW0 and HW1 are 5 V digital inputs, whereas pin HW2 is compliant with a 12 V level, e.g. it can be connected to Vbat or GND via a terminal of the PCB. For `SetPositionShort` operation: It is recommended to set HW0 and HW1 to ’1’. If the ADM bit is set to ’1’ the PA0 bit in OTP has to programmed to ’1’. If the ADM bit is set to ’0’, HW2 has to be set to ’1’.

LIN Frames

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

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

Writing frames will be used to:

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

Whereas reading frames will be used to:

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

Writing Frames

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

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

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

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

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

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

The writing frames used with the AMIS−30621 are the following:

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

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

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

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

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

Type #3: two data bytes writing frame with an identifier dynamically assigned to a particular slave node together with an application command. This type of frame requires that there are as many dynamically assigned identifiers as there are AMIS−30621 circuits using this command connected to the LIN bus.

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

Reading Frames

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

- Direct ID, which points at a particular slave node, indicating at the same time which kind of information is awaited from this slave node, thus triggering a specific command. This ID provides the fastest access to a read command but is forbidden for any other action.
- Indirect ID, which only specifies a reading command, the physical address of the slave node that must answer having been passed in a previous writing frame, called a preparing frame. Indirect ID gives more flexibility than a direct one, but provides a slower access to a read command.

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

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

Preparing Frames

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

The preparing frames used with the AMIS−30621 can be of type #7 or type #8 described below.

Type #7: two data bytes writing frame with dynamically assigned identifier. The identifier of the preparing frame has to be assigned to ROM pointer 1000, see Table 28.

Table 25. PREPARING FRAME #7

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

Where:
(*) According to parity computation

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

Table 26. PREPARING FRAME #8

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

Where:
AppCMD: If = ‘0x80’ this indicates that Data 2 contains an application command
CMD[6:0]: Application Command “byte”
AD[6:0]: Slave node physical address
Datan[7:0]: Data transmitted

Dynamic Assignment of Identifiers

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

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

Table 27. DYNAMIC IDENTIFIERS WRITING FRAME

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

Where:
CMD[6:0]: 0x11, corresponding to dynamic assignment of four LIN identifiers
Broad:If <Broad> = ‘0’ all the circuits connected to the LIN bus will share the same dynamically assigned identifiers.
Dyn_ID_x [5:0]: Dynamically assigned LIN identifier to the application command which ROM pointer is <ROMp_x [3:0]>

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

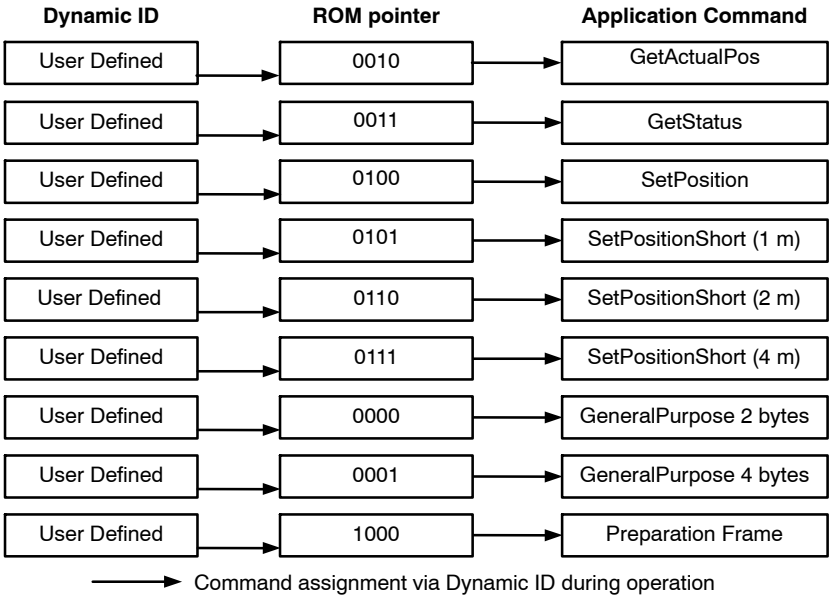


Figure 24. Principle of Dynamic Command Assignment

Commands Table

Table 28. LIN COMMANDS WITH CORRESPONDING ROM POINTER

Command Mmnemonic	Command Byte (CMD)		Dynamic ID (Example)	ROM Pointer
GetActualPos	000000	0x00	100xxx	0010
GetFullStatus	000001	0x01	n.a.	
GetOTPparam	000010	0x02	n.a.	
GetStatus	000011	0x03	000xxx	0011
GotoSecurePosition	000100	0x04	n.a.	
HardStop	000101	0x05	n.a.	
ResetPosition	000110	0x06	n.a.	
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
Preparing frame			011010	1000

NOTE: “Xxx” allows addressing physically a slave node. Therefore, these dynamic identifiers cannot be used for more than eight stepper motors. Only nine ROM pointers are needed for the AMIS−30621.

LIN Lost Behavior

Introduction

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

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 25 case A) or at (or before) start−up (See Figure 25 state SHUTDOWN):

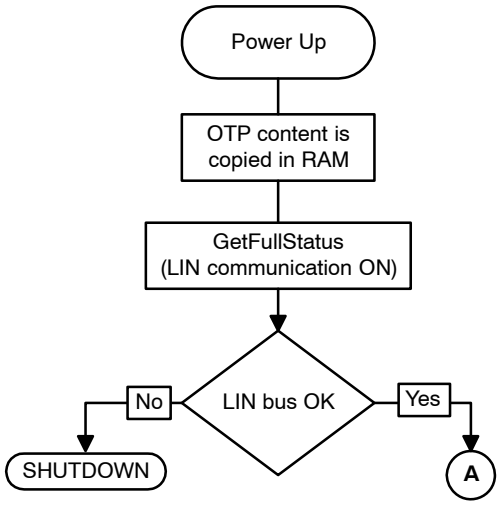


Figure 25. Flow Chart Powerup of AMIS−30621. Case A: LIN Lost During Operation and LIN Lost at Start−up Resulting in Shutdown

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 26.

“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`.

1. 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

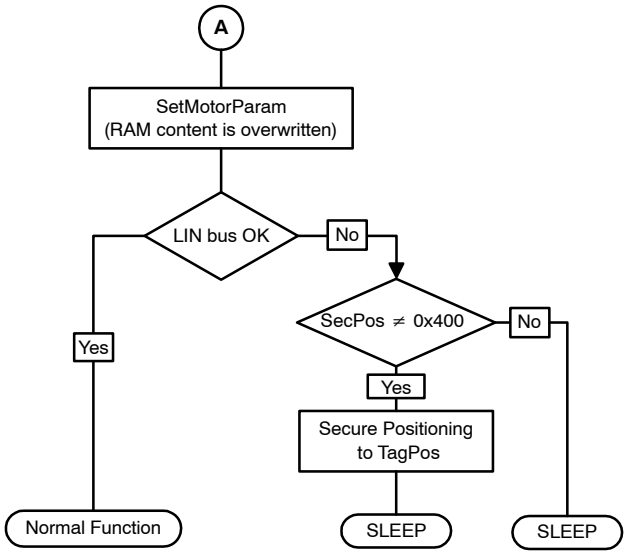


Figure 26. Case A: LIN Lost During Normal Operation

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.

LIN APPLICATION COMMANDS

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 29 below.

Table 29. 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			1	Ram Content reset
SetDualPosition	0x08			4	Drives the motor to 2 different positions with different speeds
SetMotorParam	0x09			4	Programs the motion parameters and values for the current in the motor's coils
SetOTPparam	0x10			4	Programs (and zaps) a selected byte of the OTP memory
SetPosition	0x0B			1, 3, 4	Drives the motor to a given position
SetPositionShort (1 m.)	0x0C			2	Drives the motor to a given position (half step mode only)
SetPositionShort (2 m.)	0x0D			2	Drives two motors to 2 given positions (half step only)
SetPositionShort (4 m.)	0x0E			2	Drives four motors to 4 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 shown below.

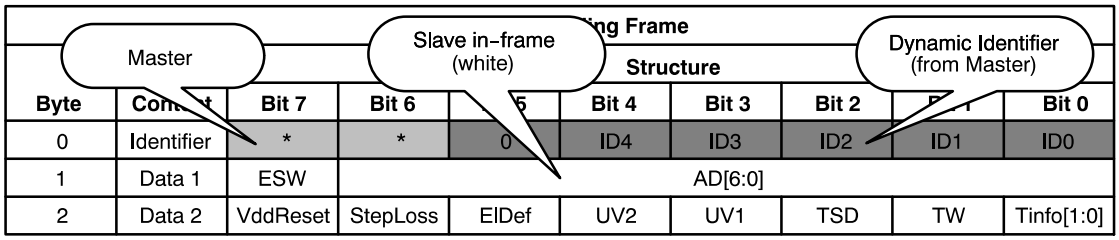


Figure 27. Color Code Used in the Definition of LIN Frames

Usually, the AMIS−30621 makes use of dynamic identifiers for general−purpose two, four or eight bytes writing frames. If dynamic identifiers are used for other purposes, this is acknowledged. Some frames implement a <Broad> bit that allows addressing a command to all the 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.

Application Commands

GetActualPos

This command is provided to the circuit by the LIN master to get the actual position of the stepper−motor. This position (<ActPos[15:0]>) is returned in signed two's complement 16−bit format. One should note that according to the programmed stepping mode, the LSB's of <ActPos[15:0]> may have no meaning and should be assumed to be '0', as 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 GetActualPos command will not attempt to reset any flag.

GetActualPos corresponds to the following LIN reading frames.

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

Table 30. 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	$V_{DDReset}$	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	
5	Checksum	Checksum over data							

Where:

(*) According to parity computation

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

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

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

Table 31. GetActualPos PREPARING FRAME TYPE #7

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

Table 32. GetActualPos READING FRAME TYPE #6

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	ESW	AD[6:0]						
2	Data 2	ActPos[15:8]							
3	Data 3	ActPos[7:0]							
4	Data 4	$V_{DDReset}$	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	
5	Data 5	0xFF							
6	Data 6	0xFF							
7	Data 7	0xFF							
8	Data 8	0xFF							
9	Checksum	Checksum over data							

Where:

(*) According to parity computation

Table 33. GetActualPos PREPARING FRAME TYPE #8

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

Table 34. GetActualPos READING FRAME TYPE #6

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	ESW	AD[6:0]						
2	Data 2	ActPos[15:8]							
3	Data 3	ActPos[7:0]							
4	Data 4	$V_{DDReset}$	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	
5	Data 5	0xFF							
6	Data 6	0xFF							
7	Data 7	0xFF							
8	Data 8	0xFF							
9	Checksum	Checksum over data							

GetFullStatus

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

Note: A `GetFullStatus` command will attempt to reset flags `<TW>`, `<TSD>`, `<UV2>`, `<ElDef>`, `<StepLoss>`, `<CPFail>`, `<OVC1>`, `<OVC2>`, `<VddReset>`.

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

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

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

Table 35. GetFullStatus PREPARING FRAME TYPE #7

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

Table 36. GetFullStatus READING FRAME TYPE #6 (1)

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	1	AD[6:0]						
2	Data 2	Irun[3:0]				Ihold[3:0]			
3	Data 3	Vmax[3:0]				Vmin[3:0]			
4	Data 4	AccShape	StepMode[1:0]		Shaft	Acc[3:0]			
5	Data 5	$V_{DDReset}$	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	
6	Data 6	Motion[2:0]			ESW	OVC1	OVC2	1	CPFail
7	Data 7	1	1	1	1	TimeE	DataE	HeadE	BitE
8	Data 8	0xFF							
9	Checksum	Checksum over data							

Table 37. GetFullStatus READING FRAME TYPE #6 (2)

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	1	AD[6:0]						
2	Data 2	ActPos[15:8]							
3	Data 3	ActPos[7:0]							
4	Data 4	TagPos[15:8]							
5	Data 5	TagPos[7:0]							
6	Data 6	SecPos[7:0]							
7	Data 7	1	1	1	1	1	SecPos[10:8]		
8	Data 8	0xFF							
9	Checksum	Checksum over data							

Where:

(*) According to parity computation

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

Table 38. GetFullStatus PREPARING FRAME TYPE#8

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

Table 39. GetFullStatus READING FRAME TYPE #6 (1)

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	1	AD[6:0]						
2	Data 2	Irun[3:0]				Ihold[3:0]			
3	Data 3	Vmax[3:0]				Vmin[3:0]			
4	Data 4	AccShape	StepMode[1:0]		Shaft	Acc[3:0]			
5	Data 5	$V_{DDReset}$	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	
6	Data 6	Motion[2:0]			ESW	OVC1	OVC2	1	CPFail
7	Data 7	1	1	1	1	TimeE	DataE	HeadE	BitE
8	Data 8	0xFF							
6	Checksum	Checksum over data							

Table 40. GetFullStatus READING FRAME TYPE #6 (2)

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	1	AD[6:0]						
2	Data 2	ActPos[15:8]							
3	Data 3	ActPos[7:0]							
4	Data 4	TagPos[15:8]							
5	Data 5	TagPos[7:0]							
6	Data 6	SecPos[7:0]							
7	Data 7	1	1	1	1	1	SecPos[10:8]		
8	Data 8	0xFF							
9	Checksum	Checksum over data							

GetOTPparam

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

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

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

Table 41. GetOTPparam PREPARING FRAME TYPE #7

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

Table 42. GetOTPparam READING FRAME TYPE #6

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	OSC3	OSC2	OSC1	OSC0	IREF3	IREF2	IREF1	IREF0
2	Data 2	1	TSD2	TSD1	TSD0	BG3	BG2	BG1	BG0
3	Data 3	ADM	(HW2) (Note 33)	(HW1) (Note 33)	(HW0) (Note 33)	PA3	PA2	PA1	PA0
4	Data 4	Irun3	Irun2	Irun1	Irun0	Ihold3	Ihold2	Ihold1	Ihold0 (Note 34)
5	Data 5	Vmax3	Vmax2	Vmax1	Vmax0	Vmin3	Vmin2	Vmin1	Vmin0
6	Data 6	SecPos10	SecPos9	SecPos8	Shaft	Acc3	Acc2	Acc1	Acc0
7	Data 7	SecPos7	SecPos6	SecPos5	SecPos4	SecPos3	SecPos2	SecPos1	SecPos0
8	Data 8					StepMode1	StepMode0	LOCKBT	LOCKBG
9	Checksum	Checksum over data							

Where:

(*) According to parity computation

33. Although not stored in the OTP memory the physical status of the hardware address input pins are returned by a read of the OTP contents.
34. The Ihold0 bit is read as '1' for product version AIMS30621C6217G and AMIS30621C6217RG.

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

Table 43. GetOTPparam PREPARING FRAME TYPE #8

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

Table 44. GetOTPparam READING FRAME TYPE #6

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	0	1	1	1	1	1	0	1
1	Data 1	OSC3	OSC2	OSC1	OSC0	IREF3	IREF2	IREF1	IREF0
2	Data 2	1	TSD2	TSD1	TSD0	BG3	BG2	BG1	BG0
3	Data 3	ADM	(HW2) (Note 35)	(HW1) (Note 35)	(HW0) (Note 35)	PA3	PA2	PA1	PA0
4	Data 4	Irun3	Irun2	Irun1	Irun0	Ihold3	Ihold2	Ihold1	Ihold0 (Note 36)
5	Data 5	Vmax3	Vmax2	Vmax1	Vmax0	Vmin3	Vmin2	Vmin1	Vmin0
6	Data 6	SecPos10	SecPos9	SecPos8	Shaft	Acc3	Acc2	Acc1	Acc0
7	Data 7	SecPos7	SecPos6	SecPos5	SecPos4	SecPos3	SecPos2	SecPos1	SecPos0
8	Data 8					StepMode1	StepMode0	LOCKBT	LOCKBG
9	Checksum	Checksum over data							

35. Although not stored in the OTP memory the physical status of the hardware address input pins are returned by a read of the OTP contents.
36. The Ihold0 bit is read as ‘1’ for product version AIMS30621C6217G and AMIS30621C6217RG.

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

Table 45. GetStatus READING FRAME TYPE #5

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
1	Data 1	ESW	AD[6:0]						
2	Data 2	$V_{DDReset}$	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	
3	Checksum	Checksum over data							

Where:

(*) According to parity computation

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

GotoSecurePosition

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

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

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

Table 46. GotoSecurePosition WRITING FRAME TYPE #1

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

Where:

(*) According to parity computation

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

HardStop

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

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

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

Table 47. HardStop WRITING FRAME TYPE #1

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

Where:

(*) According to parity computation

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

ResetPosition

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

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

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

Table 48. ResetPosition WRITING FRAME TYPE #1

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

Where:

(*) According to parity computation

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

ResetToDefault

This command is provided to the circuit by the LIN Master in order to reset to whole slave note 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 master to initialize a slave node in case of emergency, or simply to refresh the RAM content.

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

ResetToDefault will correspond to the following LIN writing frames (type #1).

Table 49. ResetToDefault WRITING FRAME TYPE #1

Byte	Content	Structure							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
1	Data 1	1	CMD[6:0] = 0x07						
2	Data 2	Broad	AD[6:0]						
3	Checksum		Checksum over data						

Where:

(*) According to parity computation

Broad: If broad = ‘0’ all the stepper motors connected to the LIN bus will 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. After Dual positioning the internal flag “Reference done” is set.

Note: 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).

Table 50. SetDualPositioning WRITING FRAME TYPE #4

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

Where:

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

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

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

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

Pos2[15:0]: Position of 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 parameter than <Vmax> while a motion is running, otherwise correct positioning cannot be guaranteed.

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

Table 51. SetMotorParam WRITING FRAME TYPE #4

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

Where:

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

SetOTPparam

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

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

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

Table 52. SetOTPparam WRITING FRAME TYPE #4

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

Where:

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

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) describes the cases where a `SetPosition` command will be ignored.

SetPosition corresponds to the following LIN write frames.

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

Table 53. SetPosition WRITING FRAME TYPE #3

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

Where:

(*) According to parity computation

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

2. Four (4) Data bytes frame with general purpose identifier (type #1). **Note:** the dynamic ID allocation has to be assigned to ‘General Purpose 4 Data bytes’ ROM pointer, i.e. ‘0001’.

Table 54. SetPosition WRITING FRAME TYPE #1

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

Where:

(*) According to parity computation

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

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

Table 55. SetPosition WRITING FRAME TYPE #4

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

Where:

Adn[6:0]: Motor #n physical address (n ∈ [1,2]). Posn[15:0]: Signed 16−bit position set−point for motor #n.

SetPositionShort

This command is provided to the circuit by the LIN Master to drive one, two or four motors to a given absolute position. 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 4 bits, hence `SetPositionShort` can only be used with a network implementing a maximum of 16 slave nodes. These 4 bits are corresponding to the bits `PA[3:0]` in OTP memory. For `SetPositionShort` operation: It is recommended to set HW0 and HW1 to ’1’. If the ADM bit is set to ’1’ the PA0 bit in OTP has to programmed to ’1’. If the ADM bit is set to ’0’, HW2 has to be set to ’1’.

Two different cases must be considered, depending on the programmed value of the ADMbit in the OTP memory.

ADM	AD[3]	Pin HW0	Pin HW1	Pin HW2	Bit PA0 in OTP memory
0	X	Tied to V_{DD}		Tied to V_{BB}	AD[0]
1	0			Tied to GND	1
1	1			Tied to V_{BB}	1

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

`SetPositionShort` corresponds to the following LIN writing frames:

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

Table 56. SetPositionShort WRITING FRAME TYPE #2

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

Where:

(*) According to parity computation

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

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

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

Table 57. SetPositionShort WRITING FRAME TYPE #2

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

Where:

(*) According to parity computation

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

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

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

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

Table 58. SetPositionShort WRITING FRAME TYPE #2

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

Where:

(*) According to parity computation

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

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

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

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.

Table 59. Sleep WRITING FRAME

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

SoftStop

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

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

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

Command `SoftStop` occurs in the following cases:

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

Table 60. SoftStop WRITING FRAME TYPE #1

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

Where:

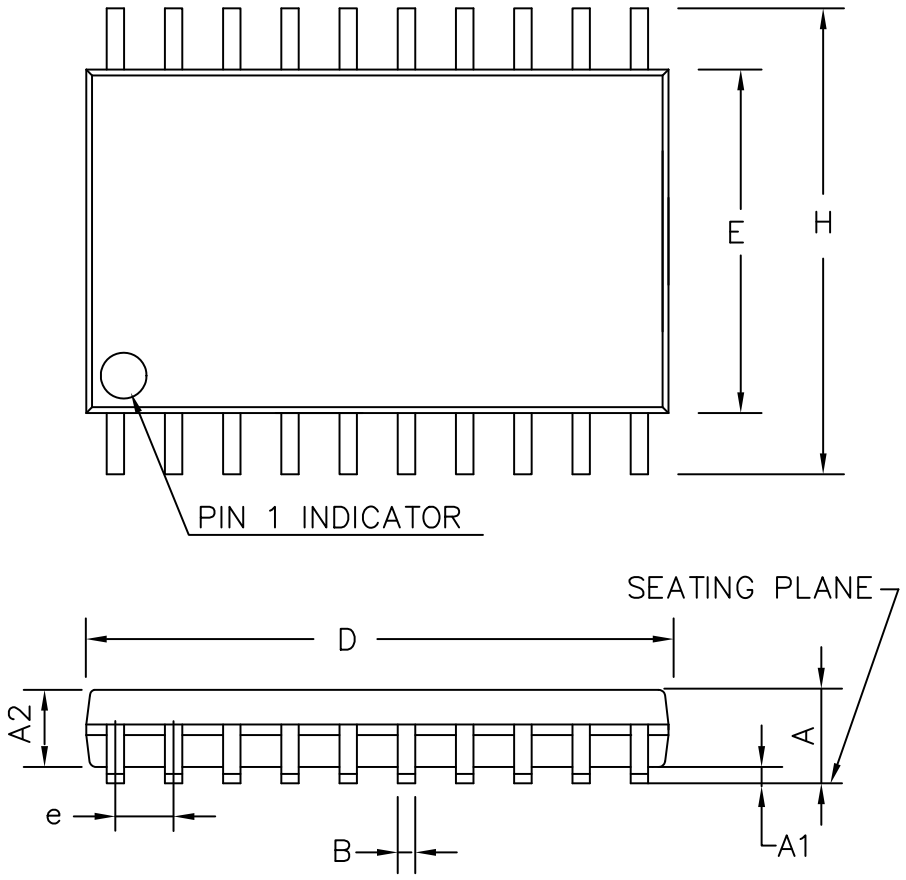
(*) According to parity computation

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

PACKAGE DIMENSIONS

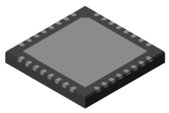
SOIC 20 W
CASE 751AQ−01
ISSUE O

DATE 19 JUN 2008



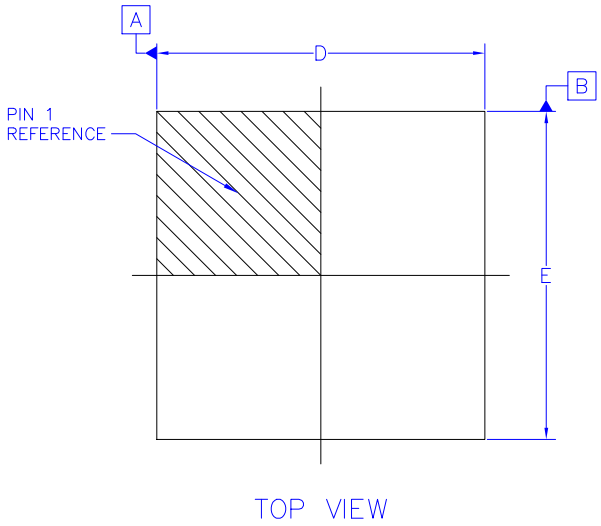
DIMENSIONS IN INCHES			
SYMBOL	MIN.	NOM.	MAX.
A	0.093	0.099	0.104
A1	0.004	0.008	0.012
A2	0.088	0.094	0.100
B	0.013	0.016	0.020
C	0.0090	0.0100	0.0125
D	0.496	0.503	0.510
E	0.292	0.296	0.299
e	.050 BSC.		
H	0.394	0.402	0.419
h	0.010	0.015	0.019
L	0.016	0.033	0.050
α	0°	5°	8°

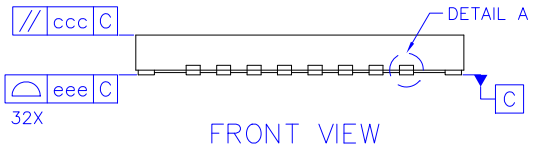
PACKAGE DIMENSIONS

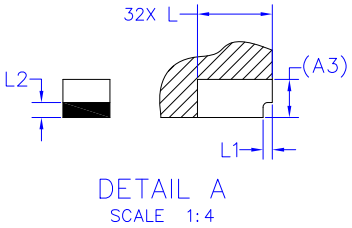


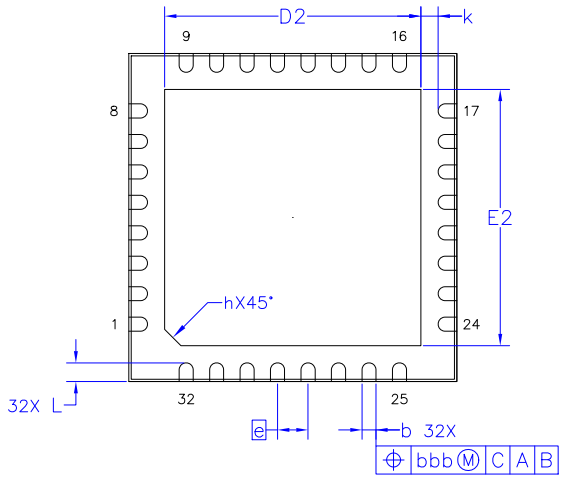
QFNW32 7x7, 0.65P
CASE 484BB
ISSUE O

DATE 14 DEC 2021









NOTES:

1. DIMENSIONING AND TOLERANCING PER ASME Y14.5, 2018.
2. ALL DIMENSIONS ARE IN MILLIMETERS AND IN DEGREES.
3. COPLANARITY APPLIES TO THE EXPOSED PAD AS WELL AS THE TERMINALS.
4. DIMENSIONS D AND E ARE CONSIDERED AS SPECIAL CHARACTERISTIC.
5. REFER TO JEDEC STANDARD MO−220.

DIM	MILLIMETERS		
	MIN.	NOM.	MAX.
A	0.80	0.85	0.90
A1	0.00	–	0.05
A3		0.203 REF.	
b	0.25	0.30	0.35
D	6.90	7.00	7.10
D2	5.37	5.47	5.57
E	6.90	7.00	7.10
E2	5.37	5.47	5.57
e		0.65 BSC	
h		0.35	
k		0.36 REF.	
L	0.35	0.40	0.45
L1	0.00	–	0.10
L2	0.10	–	–
bbb		0.10	
ccc		0.10	
eee		0.08	

GENERIC MARKING DIAGRAM*



XXXX = Specific Device Code
A = Assembly Location
WL = Wafer Lot
YY = Year
WW = Work Week
G = Pb−Free Package

*This information is generic. Please refer to device data sheet for actual part marking. Pb−Free indicator, “G” or microdot “▪”, may or may not be present. Some products may not follow the Generic Marking.

onsemi, ONSEMI , and other names, marks, and brands are registered and/or common law trademarks of Semiconductor Components Industries, LLC dba "**onsemi**" or its affiliates and/or subsidiaries in the United States and/or other countries. **onsemi** owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of **onsemi**'s product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent−Marking.pdf. **onsemi** reserves the right to make changes at any time to any products or information herein, without notice. The information herein is provided "as−is" and **onsemi** makes no warranty, representation or guarantee regarding the accuracy of the information, product features, availability, functionality, or suitability of its products for any particular purpose, nor does **onsemi** assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using **onsemi** products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by **onsemi**. "Typical" parameters which may be provided in **onsemi** data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. **onsemi** does not convey any license under any of its intellectual property rights nor the rights of others. **onsemi** products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use **onsemi** products for any such unintended or unauthorized application, Buyer shall indemnify and hold **onsemi** and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that **onsemi** was negligent regarding the design or manufacture of the part. **onsemi** is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Email Requests to: orderlit@onsemi.com

onsemi Website: www.onsemi.com

TECHNICAL SUPPORT
North American Technical Support:
Voice Mail: 1 800−282−9855 Toll Free USA/Canada
Phone: 011 421 33 790 2910

Europe, Middle East and Africa Technical Support:
Phone: 00421 33 790 2910
For additional information, please contact your local Sales Representative

◊