AN4241
Application note

L6470 and L6472: fully integrated stepper motor drivers

Enrico Poli

Introduction

The L6470 and L6472 are fully integrated motor drivers providing a complete standalone solution for the high-end stepper motor applications. The devices can be controlled by a host microcontroller through a fast SPI interface and are able to execute a complete set of motion commands.

This document describes how the devices can be configured and gives some suggestions about the operation and the application design.

The current control algorithm of the devices (the L6470 voltage mode driving and the L6472 advanced current control) is not investigated in this document.

For further details, please, refer to the respective application notes AN4144 “Voltage mode control operation and parameter optimization” and AN4158 “Peak current control with automatic decay adjustment and predictive current control: basics and setup”.

Contents

1 The L6470 and L6472 communication interface

The device (always slave) can be driven by a MCU (always master) sending commands through an 8-bit SPI interface. The 8-bit shift register of the device is kept enabled while $\overline{CS}$ input is forced low. During this time, at every raising edge of the serial clock (CK), the SDI input is stored into the shift register. At CK falling edges the SDO output is updated according to the last bit of the shift register.

When the $\overline{CS}$ input is raised, the device catches the shift register content and interprets its value as a command or an argument of the previously received command.

All the bytes are sent through the SPI data lines starting from the most significant bit.

1.1 Communication protocol

The communication protocol is based on single byte commands that can be followed by a command argument up to 3 byte long which must be transmitted starting from the most significant byte.

Part of the information needed to execute the target operation could be embedded into the command byte, for example the target register address in the GetParam and SetParam commands, and the argument provides extra data as well as the target position of the GoTo command.

By default the response byte of the device is h00 (hexadecimal format). Some commands, for example those used to read the value of a register, generate a response from the device up to 3 byte long which is transmitted during the following transmission cycles starting from the most significant byte.

1.2 Daisy chain

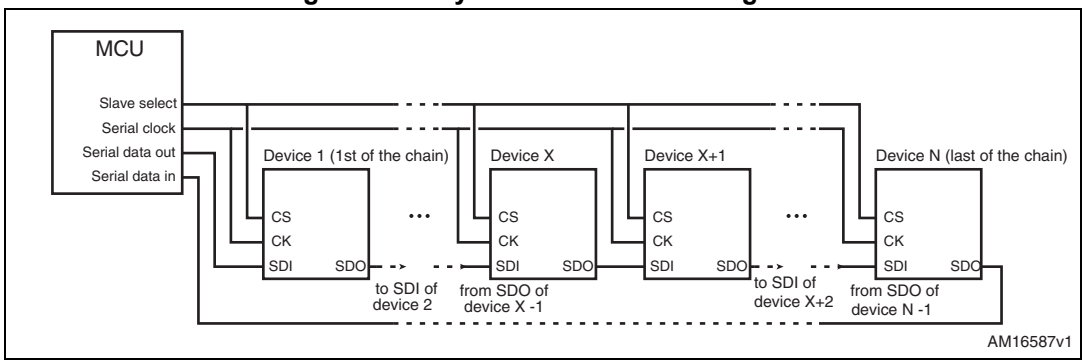
The device is compatible with the daisy chain architecture allowing the MCU to drive multiple devices with a single SPI interface.

The daisy chain architecture is obtained as follows:

- Master serial clock line is connected to the CK input of each slave device;
- Master slave select line is connected to the $\overline{CS}$ input of each slave device;
- Master serial data output (MOSI) is connected to the SDI input of the first slave of the chain;
- The SDO output of each slave device is connected to the SDI input of the next one, last slave excluded;
- Master serial data input (MISO) is connected to the SDO output of the last slave of the chain,

The connection diagram of the configuration is shown in *Figure 1*.

Figure 1. Daisy chain connection diagram



In this configuration, the chain of slaves acts as a single slave with an SPI device of N byte. Each communication cycle, for example when the master needs to transmit/receive a byte from/to a slave, the master must fill all the shift registers of the slaves before raising the $\overline{CS}$ line.

The devices are addressed according to the position of the byte in the communication cycle: the first byte transmitted by the master is received by the last device of the chain; the second one is received by the last-but-one slave and so on down to the last transmitted byte which is received by the first slave of the chain. The response bytes from the device chain are addressed to the same way: the first byte received by the master has been transmitted by the last device of the chain; the second one has been transmitted by the last-but-one slave and so on down to the last received byte which has been transmitted by the first slave of the chain.

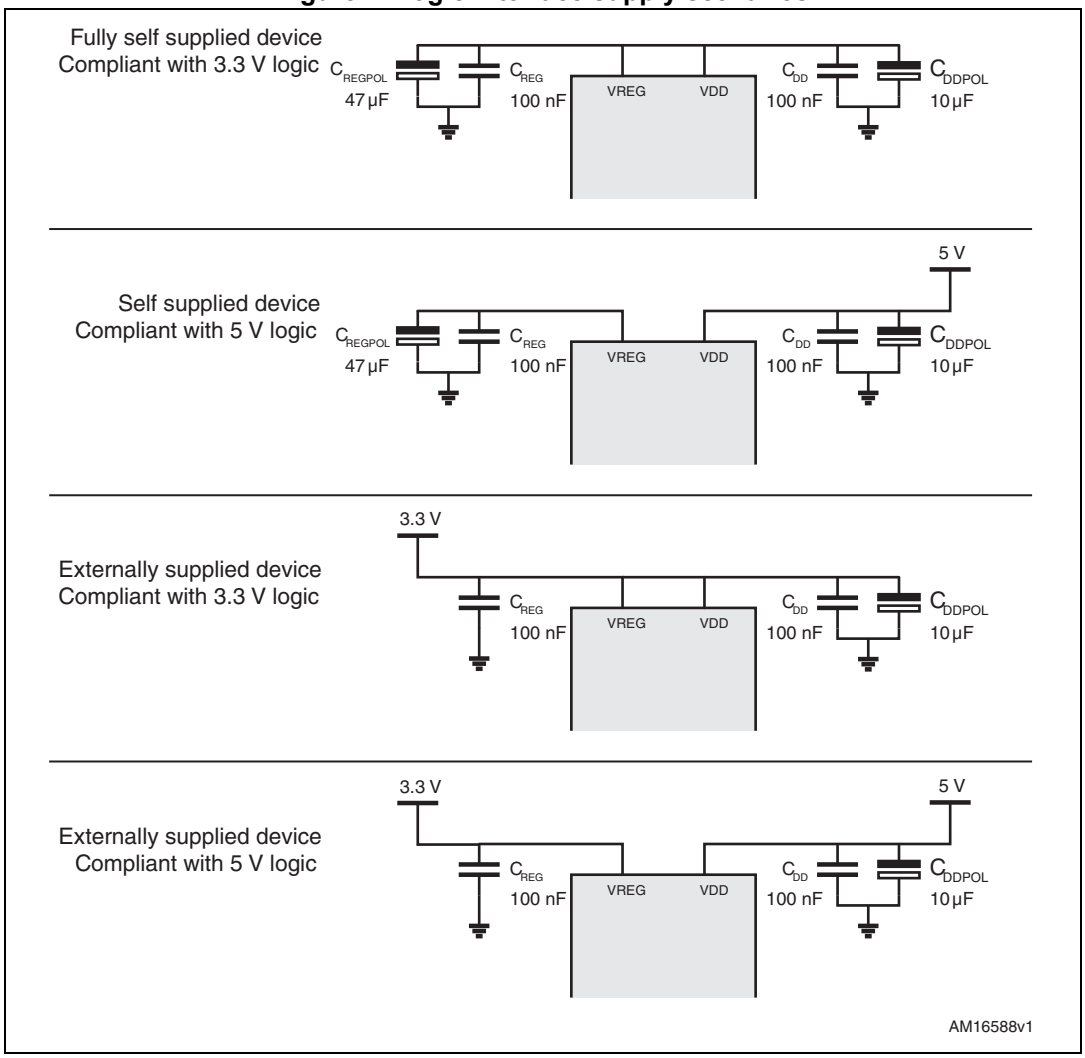
In theory, the number of slaves that an MCU can drive using the daisy chain configuration is unlimited; in practice the maximum number of devices connected to the same SPI depends on the clock skew.

The number of slaves limits the communication speed also because every time a byte has to be transmitted to a device, the whole N slave chain has to be filled transmitting N - 1 extra bytes.

1.3 5 V and 3.3 V communication interface

The device can be configured to operate both with 3.3 V and 5 V standard logic as shown in *Figure 2*.

Figure 2. Logic interface supply scenarios



2 Motion engine

The L6470 and L6472 devices integrate a motion engine providing a full set of commands. The motion engine generates the step sequence according to the programmed speed profile and the requested command.

The speed profile represents the operation boundaries, defined by the acceleration, deceleration, maximum and minimum speed, which should be respected to ensure the proper functioning of the application.

The devices allow all the parameters to be set independently:

- Acceleration and deceleration values range from 14.55 up to 59590 steps/s2. The device can also be set to use an infinite acceleration and deceleration value; in this case, both the acceleration and deceleration phases are totally skipped.
- Maximum speed value ranges from 15.25 steps/s up to 15610 step/s.
- Minimum speed value ranges from 0 up to 976 steps/s.

The acceleration, deceleration and minimum speed parameters can be modified when the motor is stopped only. The maximum speed can be also changed when the motor is running, but the new value is only considered at next command execution.

The commands supported by the motion engine are listed in *Table 1*.

Table 1. Command list

Name	Length (bytes)	Description	Notes
Move	4 (including 3 of arguments)	Performing the target number of microsteps as per requested direction.	Can be executed when the motor is stopped only.
GoTo	4 (including 3 of arguments)	Reaching the absolute target position (ABS_POS register) using the shortest path.	Not accepted while another command is under execution.
GoTo_DIR	4 (including 3 of arguments)	Reaching the absolute target position (ABS_POS register) running as per requested direction.	Not accepted while another command is under execution.
GoHome	1	Reaching the home position (all zeroes) using the shortest path.	Not accepted while another command is under execution.
GoMark	1	Reaching the position stored into the MARK register using the shortest path.	Not accepted while another command is under execution.
Run	4 (including 3 of arguments)	Reaching the target speed in the requested direction.	Always accepted and immediately executed (if present, the previous command is aborted).
StepClock	1	Switching the device in step-clock mode imposing the direction.	Can be executed when the motor is stopped only.
GoUntil	4 (including 3 of arguments)	Reaching the speed target in the requested direction and stopping when SW input is forced low (falling edge).	Always accepted and immediately executed (if present, the previous command is aborted).
ReleaseSW	1	Running the motor at low speed in the requested direction and stopping when SW input is forced high (rising edge).	Always accepted and immediately executed (if present, the previous command is aborted).

Table 1. Command list (continued)

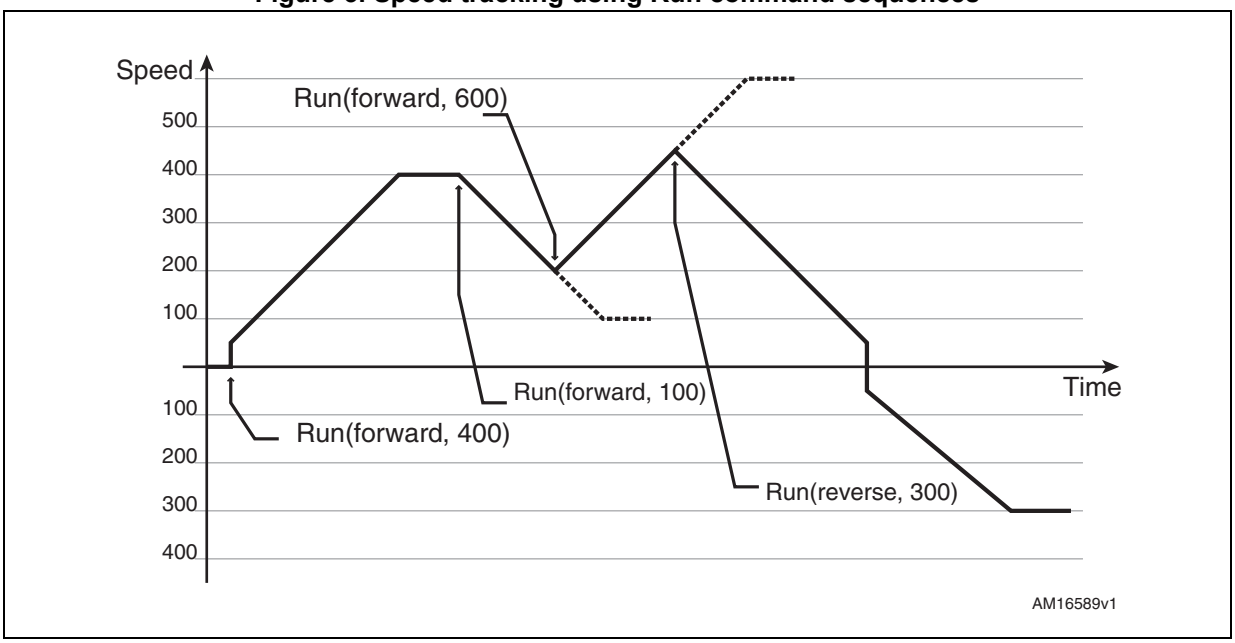
Name	Length (bytes)	Description	Notes
SoftStop	1	Stopping the motor in accordance to the programmed speed profile.	Always accepted and immediately executed (if present, the previous command is aborted).
HardStop	1	Stopping the motor immediately (infinite deceleration).	Always accepted and immediately executed (if present, the previous command is aborted).
SoftHiZ	1	Stopping the motor in accordance to the programmed speed profile and then disabling the power bridges.	Always accepted and immediately executed (if present, the previous command is aborted).
HardHiz	1	Disabling the power bridges immediately.	Always accepted and immediately executed (if present, the previous command is aborted).

2.1 Speed tracking commands

During the speed tracking, the device dynamically changes the motor speed according to the application requirements. The Run command can be used to achieve this result.

The Run command sets the speed target and direction which the motor has to reach. Both speed target and direction can be changed in real time through a new Run command (see *Figure 3*).

Figure 3. Speed tracking using Run command sequences



2.2 Positioning commands

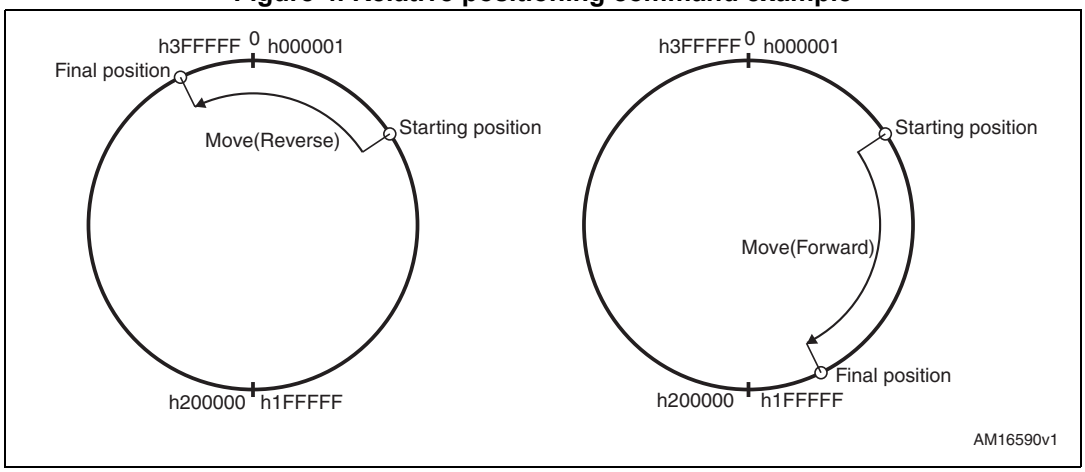
The motion engine, integrated into the devices, allows the position of the motor in a target position based on integrated ABS_POS register.

The ABS_POS register traces all the motion performed by the motor adding a unit to each microstep completed in forward direction and subtracting a unit when the microstep is performed in reverse direction.

The target positioning can be directly imposed indicating the ABS_POS register value (absolute position) or the distance between the current position and the target one (relative position).

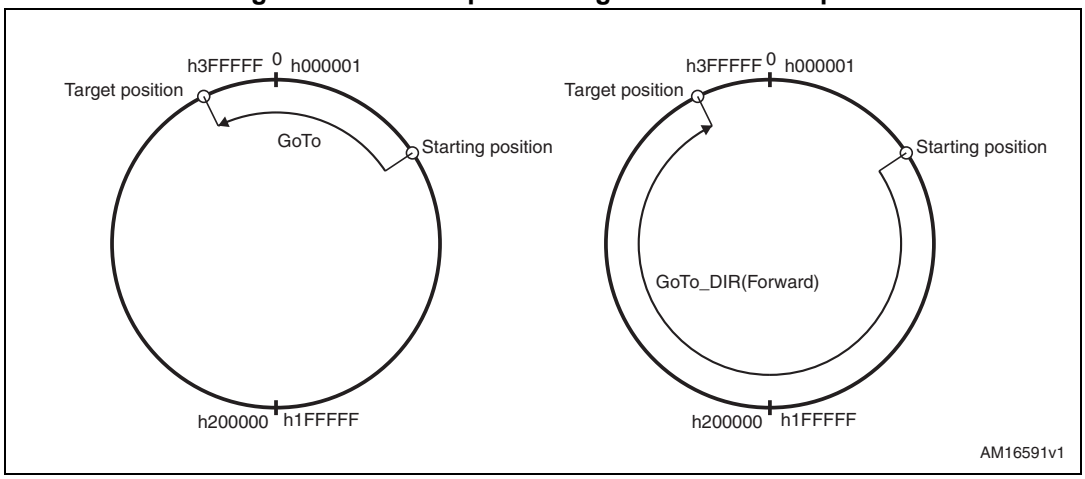
The relative positioning command is Move. The motion engine executes this command only when the motor is stopped in order to avoid unexpected behaviors (e.g.: the target of number of steps is not enough to allow the speed profile compliance).

Figure 4. Relative positioning command example



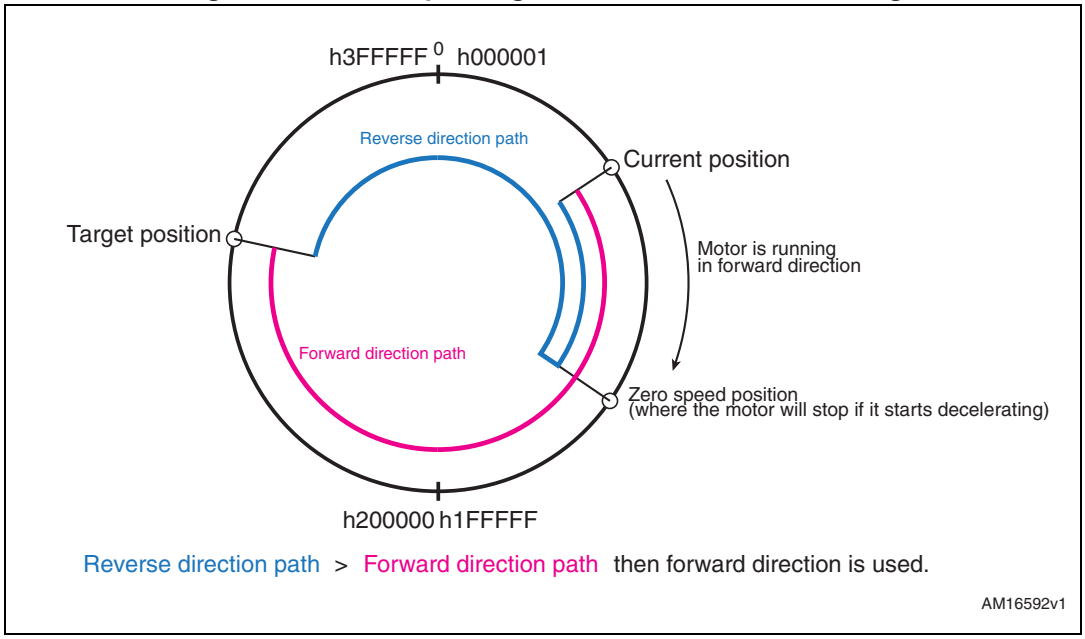
The absolute positioning commands are GoTo and GoTo_DIR. The former moves the motor to the position target choosing the rotation direction according to a minimum path algorithm (i.e.: the lower number of microsteps is executed) whereas the latter imposes the direction directly.

Figure 5. Absolute positioning command example



If a GoTo command is requested when the motor is running, the minimum path algorithm also considers the steps required to reverse the direction (see *Figure 6*).

Figure 6. Minimum path algorithm when motor is running



The GoTo and GoTo_DIR can only be executed when no other commands are under execution.

Change the target position of the on the fly command

If a Run command is sent to the device during the execution of a GoTo command, it aborts the previous command. This effect can be used to change the target position of the motion engine on the fly.

Following the suggested sequence of operations:

1. Read the current motor speed (SPEED register) and direction (DIR bit in the STATUS register).
2. Send a Run command setting the target speed and direction equal to the values obtained at point 1.
3. Wait for the execution command monitoring the BUSY pin or the BUSY flag in the STATUS register.
4. Send the new positioning command.

This operation could introduce a small error in the generation of the speed profile. If the target position is changed a high number of times (tracking position) the error increases and anomalous behaviors could occur.

2.3 Stop commands

The motor can be stopped through the stop commands. These commands can be sent at any time and they are executed immediately.

The SoftStop command stops the motor fitting the deceleration value of the speed profile whereas HardStop command stops the motor immediately (infinite deceleration).

The SoftHiZ and HardHiZ commands operate similarly, but the power bridges are disabled as soon as the zero speed is reached (the high impedance status is forced).

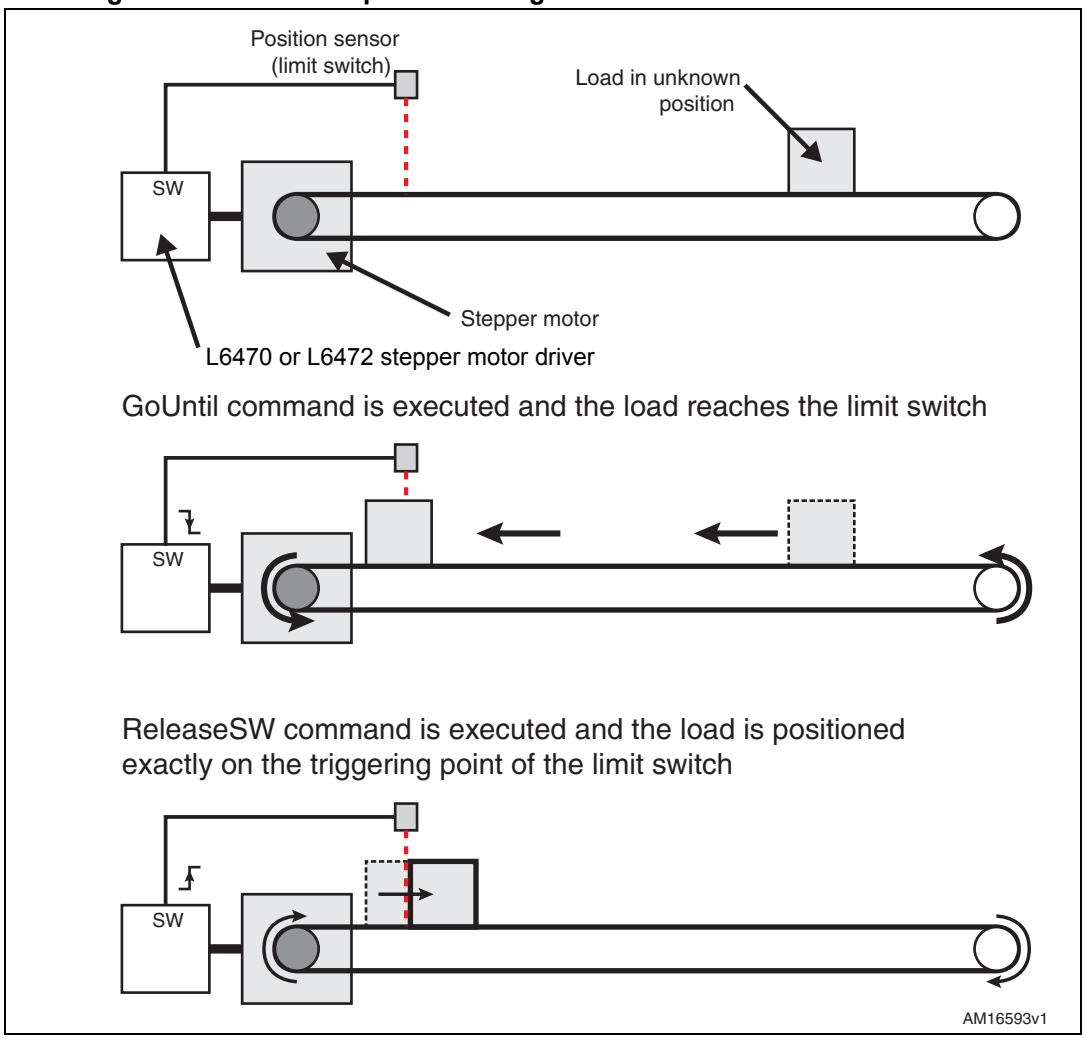
2.4 Initializing position using GoUntil and ReleaseSW commands

The GoUntil and ReleaseSW commands can be used to initialize the absolute position information stored into the ABS_POS register according to an external position sensor. In this way a relation between ABS_POS value and mechanical position of the motor can be established.

The position initialization sequence, as shown in *Figure 7*, is the following:

1. In power-up status, the load position is unknown.
2. Using GoUntil command the load is moved to the limit switch at high speed.
3. When the load reaches the limit switch the SW input of the device is forced low. The motor decelerates and then stops.
4. Considering the high speed used to approach the load to the limit switch, a significant error in the positioning could happen.
5. Using ReleaseSW command the load is moved away from the limit switch at low speed.
6. As soon as the threshold position of the limit switch is crossed by the load the SW input of the device is forced high. The motor stops immediately.
7. The load is positioned in correspondence to the threshold position of the limit switch with a high precision.

Figure 7. Initialization position using GoUntil and ReleaseSW commands



3 Protections

The L6470 and L6472 devices provide a complete set of protections designed to prevent from damaging the device in critical conditions.

The implemented protections are:

- Overtemperature (see *Section 3.1*)
- Overcurrent (see *Section 3.2*)
- Undervoltage (see *Section 3.3*)

3.1 Overtemperature protection

The overtemperature protection disables the power stage of the device when the temperature of the chip exceeds the safe operation conditions.

When the overtemperature protection is triggered the device is locked in a safe condition (all MOSFETs are turned off) and is kept in this condition until the junction temperature decreases below 130 °C (typical). The thermal shutdown event occurrence is signaled through the respective flag (TH_SD) in the STATUS register which is kept low until it is released by a GetStatus command (more details are available in the datasheet of the device).

A warning threshold is also present allowing the host to control the device to prevent the shutdown.

3.2 Overcurrent protection

The overcurrent protection monitors the current in all power MOSFETs of the device and disables the power stage when the programmed current threshold is reached. No information about the specific MOSFET or bridge causing the failure is available.

As soon as the overcurrent protection is triggered, the device is locked in a safe condition (all MOSFETs are turned off) and is kept in this condition until the OCD failure flag is released by a GetStatus command.

When the device is locked in safe state no command enabling the bridges can be executed (e.g.: Move, Run, GoTo, HardStop, etc.). The commands are simply ignored, no non-performable signaling is returned by the device.

Warning: The overcurrent protection can be disabled setting the OC_SD pin of the CONFIG register to zero. However it is not recommended this protection to be disabled.

3.3 Undervoltage

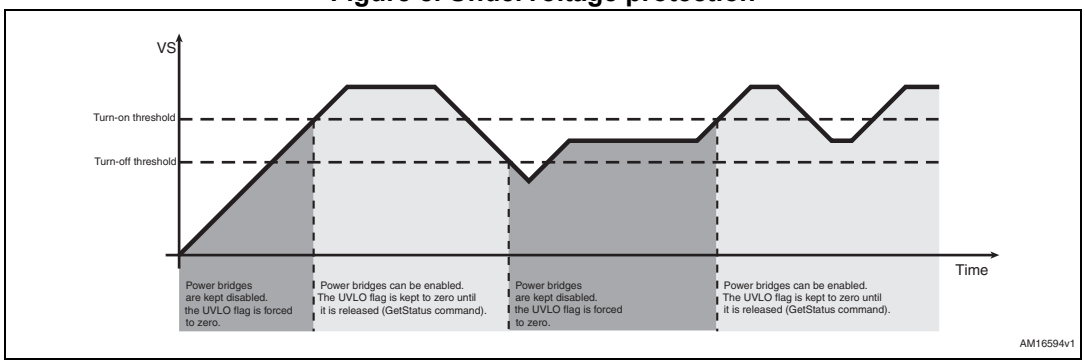
The undervoltage protection avoids the power stage of the device to operate with a supply voltage below the safe conditions.

At power-up, the device is in undervoltage status: the power bridges of the device are kept disabled until the supply voltage is below the turn-on threshold. In this condition all the commands enabling the bridges (e.g.: Move, Run, GoTo, HardStop, etc.) are ignored and the UVLO failure flag is forced low.

When the turn-on threshold is reached the power bridges are operative. The UVLO flag is kept low until it is released through a GetStatus command.

The device returns in undervoltage status if the supply voltage falls below the turn-off threshold.

Figure 8. Undervoltage protection



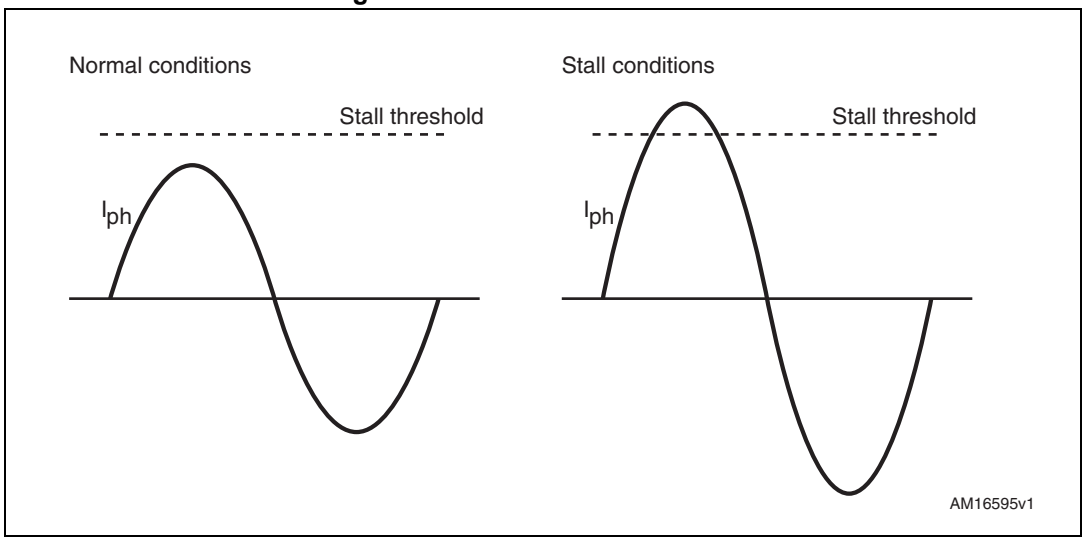
4 Stall detection

The L6470 also includes a sensorless stall detection system. This feature allows the device to detect the stall condition of the motor measuring the increase of phase current caused by the sudden cancellation of the back electromotive force.

The voltage mode control applies a sinusoidal voltage to the motor phases obtaining a sinusoidal current. The amplitude of the voltage sinewave increases according to the speed in order to compensate the back electromotive force effect and keep the amplitude of the phase current constant. If the motor stalls (i.e.: stops its rotation) the back electromotive force becomes null, but the voltage applied to the phase is still increased to face its effect. In this condition the phase current results higher than the nominal one and the stall condition can be easily detected.

The stall detection threshold must be set above the nominal peak current.

Figure 9. Stall detection threshold



5 Main clock source

The motion engine integrated into the L6470 and L6472 devices generates the internal step clock starting from the main clock source of the system. As a consequence, the perturbations on the main clock affect the accuracy with which the device generates the speed of the motor (step rate).

In the L6470 the main clock also affects the frequency of the PWM modulators and in the L6472 it is used for the measurement of the timings of the advanced current control.

The devices can operate with both external and internal clock sources. By default the clock source is the integrated oscillator operating at 16 MHz. This solution, if compared to a high precision external source, allows the maximum integration with a lower precision of the generation of the speed profile (acceleration, deceleration and target speed). The effects on the current control systems (voltage mode or advanced current control) are negligible.

The device can be configured to use an external clock source or to drive a crystal/resonator. In both cases the clock frequency must be chosen among four values: 8, 16, 24 or 32 MHz.

Warning: Setting the wrong value or applying a clock frequency significantly different from the nominal value might cause failures.

The external clock source allows a higher precision of the speed profile to be obtained.

6 Layout suggestions

The L6470 and L6472 devices integrate a power stage which can dissipate a significant amount of power compared to the die dimension. For this reason, the devices are proposed in packages with high thermal performance and exposed pads in order to ease the heat dissipation.

In this situation, the layout of the board is a significant part of the thermal design of the application.

The exposed pad of the device must be directly connected to the ground layer of the board, which acts as a heat sink. For the same reason, the size of the ground layer must be as wider as possible and, if more layers are present, the top and bottom ones should be preferred. In multi-layer designs the exposed pad must be connected to all the ground layers using multiple vias equally spaced.

7 Revision history

Table 2. Document revision history

Date	Revision	Changes
25-Jan-2013	1	Initial release.
15-Apr-2015	2	Replaced main title *on page 1* by new title “L6470 and L6472: fully integrated stepper motor drivers“. Replaced “dSPIN™ family devices“ and “dSPIN™“ by “L6470 and L6472 devices“ in the whole document. Updated *Figure 7 on page 11* (replaced “dSPIN family motor driver“ by “L6470 or L6472 stepper motor driver“). Minor modifications throughout document.