Implementing Overcurrent and Overload Protection for L6470 Based Systems

By Dennis Nolan

Main components	
L6470	Fully integrated microstepping motor driver with motion engine and SPI

Purpose

This tip presents a simple implementation of overcurrent and overload protection, implemented using the over current detection and stall detection circuit in the L6470.

Overcurrent protection

First, some general philosophy regarding overcurrent protection in motor control systems. When devising an overcurrent protection scheme for an electronic motor drive system we are generally concerned about protecting both the motor and the power semiconductors. This usually complicates the design philosophy because the motor and power transistors, generally speaking, have vastly different thermal time constants. Also, when we consider the subject of thermal time constants, we are inevitably driven to a consideration of what I will call instantaneous overcurrent (or just overcurrent) vs. overload. Traditionally, in motor control systems, the ideal protection scheme will detect either an overcurrent (instantaneous) or an overload (too much current for too much time).

When we define an instantaneous overcurrent limit (either for a semiconductor chip or a motor) we are defining a level of current that will destroy (usually burn out either a die attach wire in the chip or a coil wire in the motor) in a matter of just a few milliseconds. Generally speaking, overcurrent must be hardware based since software checks would be too slow. The L6470 has a current rating (very short term) of 3 amps RMS or 4.242 sinusoidal peak. The overcurrent threshold (which can instantaneously shut down the part via hardware) can be set between 375 mA and 6A. The OCD_TH register contains the overcurrent threshold value. The available range is from 375 mA to 6 A, in steps of 375 mA, as shown in Table 1. Therefore, if we are solely concerned with overcurrent (instantaneous) protection for the L6470, a good choice for the threshold might be 4.125 amps.

OCD-TH [3 … 0]				Overcurrent detection threshold
0	0	0	0	375 mA
0	0	0	1	750 mA
…	…	…	…	…
1	1	1	0	5.625 A
1	1	1	1	6 A

Table 1. Overcurrent detection threshold

The L6470 allows for the instantaneous overcurrent detection to either be enabled to automatically latch and shut down both bridges, or just latch a status flag and allow the bridges to continue operating. Because, in the case of a true instantaneous overcurrent, we are dealing with possible destruction within a few milliseconds, the presumption here is that the overcurrent bridge shutdown would always be enabled.

Mitigating factors for the selection of overcurrent threshold

If the system is running a lower current motor and would never be expected to get anywhere close to 4.125 amps, then my all means a lower threshold is a prudent choice. Generally speaking, the lower the threshold setting the safer for the components. The real world, however, does have noise and imprecision to deal with so some margin must be added to the threshold so that we do not experience nuisance trips. The main point here: a setting as high as 4.125 amps should be safe to protect the L6470 from instantaneous overcurrent events.

Our next consideration, can the motor withstand instantaneous (a few milliseconds) current of as high as 4.125 amps (or wherever we have decided is a good threshold with regard to the L6470 protection). Only the motor manufacturer can give us this information (or extensive, destructive, testing of many motors). Clearly, if the motor cannot survive instantaneous over currents at a level we have deemed appropriate for the chip, then the threshold must be reduced commensurate with what the motor can withstand.

Overload protection

Having developed some philosophy for selection of the instantaneous overcurrent threshold, let us turn our attention to the subject of long term overload protection. Long term overload protection is desirable if not essential in any high reliability system. We have established that 4.125 amps is probably an acceptable threshold for the instantaneous trip point (although lower would be better if it can be tolerated without nuisance trips). So, what happens if our system sits “all day” at 4.0 amps? The reality is, the chip burns up. Experience has shown that the best thermal package, best PCB thermal design (lots of exposed copper and a big array of thermal vias to get the heat to both sides of the PCB) can give us a continuous (time well beyond the thermal time constant of the system) current rating of 1.5 to perhaps 2.0 amps RMS.

So, if we are operating somewhere greater than 2 but less than 3 amps for a long period of time, the chip will get too hot and may be damaged. A careful thermal analysis, using the specified Rdson of the MOSFETs, the load current, and the effective junction to ambient thermal resistance of the system will bear this conclusion out. This is a classic overload situation. We can get away with “sneaking” the operating current into the region between the continuous rating and the instantaneous overcurrent level for a short time (before the system has enough time to get too hot), but this level will eventually cause a failure.

Since power dissipation in the FETs is primarily proportional to the Rdson times the square of the current ($R * i^2$), many more sophisticated motor drive systems have traditionally used the “I squared T“ method of overload protection. In this scheme, since these more sophisticated systems will generally have an internal variable within the microcontroller which represents actual motor current, this numerical value can be squared and the result accumulated into an integration variable so that it represents the product of time and the square of load current. When this integrand reaches some pre-determined threshold, the system shuts down on an overload trip. How is the threshold determined? Generally, by empirical methods where measurements have established a correlation between the integrand value and chip temperature.

One other practical issue must be addressed regarding the I squared T method. As previously described, no matter how small the current level, the integral will always eventually hit the threshold as this is the nature of integrating a number which is only ever positive. Practical schemes must allow the integrand to attempt to somewhat track the chip temperature. After all, our ultimate goal is to be sure that the chip does not get too hot. We know that, if the part gets very hot and then we reduce the operating current, then the temperature will start to reduce.

One popular variation on the scheme is to select a second threshold. If the load current is above this second threshold, we take the square of the excess (the amount by which we are above the threshold) and add it into the integrand. If the load current is below this set point, we square that difference and subtract it from the integrand. Generally speaking, we will not let the integrand go below some pre-determined lower limit (perhaps zero).

So far we have considered that the accumulated I squared T would attempt to track the chip to protect it against excessive temperature. This need not be the case. It may indeed turn out, in some systems, that the motor temperature would become excessive long before that of the driver. In that case, we can use the same technique to attempt to model the motor heating (the power dissipation also being largely proportional to I squared T) and determine a threshold that will protect the motor (even if the chip would not be in danger). Add to that the fact that the L6470 already has an over temperature feature, which will monitor the internal temperature and shut down if pre-determined safe limits are exceeded, and it may indeed be more prudent to develop an overload algorithm geared toward protecting the motor.

Consider some more practical aspects of providing an overload protection feature in the case of the typical L6470 based system. In such a system, a true numerical representation of the load current is probably not available. One suggestion is to utilize the L6470 stall detection feature, along with a somewhat more complex algorithm running on the microcontroller.

The stall detection is really nothing more than an almost duplicate of the overcurrent feature operating with a separately selectable threshold. This threshold operates over a smaller range and with a much finer resolution than that of the overcurrent. The STALL_TH register contains the stall detection threshold value. The available range is from 31.25 mA to 4 A with a resolution of 31.25 mA, as shown in Table 2.

STALL-TH [3 … 0]				Stall detection threshold
0	0	0	0	31.25 mA
0	0	0	`	62.5 mA
…	…	…	…	…
1	1	1	0	3.969 A
1	1	1	1	4 A

Table 2. Stall current detection threshold

The other major difference is that while OCD is represented by one latched status flag in the status register and can autonomously disable the bridge (if selected to do so), exceeding the stall threshold will latch either the STEP_LOSS_A or STEP_LOSS_B (depending on the offending bridge) but this cannot automatically shut down the bridges. Reading the status register will clear the latched status bits.

The original intent of the stall detection feature is that the controller would regularly poll the status of the step loss flags to determine if the motor has encountered excessive load and stalled, thus “slipping poles” or losing steps. This is based on the well-known phenomena that, in a voltage mode driver, the loss of motor bemf which occurs when a motor stalls will generally cause the current to rise above normal operating levels. In many systems, this phenomenon enables us to detect a motor stall. There is nothing, however, to prevent us from using this stall detection feature as the basis for an overload protection scheme.

First, let us assume that the controller is executing some sort of “heartbeat” regularly timed interrupt service routine. We will assume that the run interval is one millisecond. We can then, once per millisecond, poll the status register. Of course, if we find that the OCD bit is true, we will want to execute an appropriate response routine knowing that the drive is no longer operating. Let us also assume that we have created an overload integrand variable similar to that which we have previously discussed. Furthermore, let us assume that if our reading of the status register indicates that either of the STEP_LOSS flags is true, then we will add 10 counts to the integrand.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
SCK_MOD	STEP_LOSS_B	STEP_LOSS_A	OCD	TH_SD	TH_WRN	UVLO	WRONG_CMD
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
NOTPERF_CMD	MOT_STATUS		DIR	SW_EVN	SW_F	BUSY	HiZ

Table 3. Status Register

The TH_WRN, TH_SD, OCD flags are active low and indicate, respectively, thermal warning, thermal shutdown and overcurrent detection events.

STEP_LOSS_A and STEP_LOSS_B flags are forced low when a stall is detected on bridge A or bridge B respectively.

Also, if we find that neither of the STEP_LOSS flags is true, then we will subtract 2 counts from the integrand. In this way, we will have created a “fast charge, slow discharge” system. If the current is repeatedly in excess of our second “overload current” threshold, then the integrand will quickly increase until it crossed an “overload shut down threshold”. If we just briefly exceed the threshold, we could build up some value in the integrand which is substantial but still less than the trip threshold. If we then settle into a longer period where current is below the overload threshold, the integrand will be “discharged” at a rate of 2 counts per interrupt, eventually coming back to a lower limit.

Of course the “up count”, “down count” and overload and trip thresholds would all have to be determined experimentally. Since we are accumulating the rough equivalent of amp-seconds rather than amp^2 seconds, this algorithm does not model temperature rise as well as a true I squared T scheme. It has, however, been my experience that with some care as to the selection of the adjustable parameters, this algorithm can provide a reliable overload protection feature.

Support material

Documentation
Datasheet, L6470 Fully integrated microstepping motor driver with motion engine and SPI

Revision history

Date	Version	Changes
22-Oct-2018	1	Initial release