AN10898

BLDC motor control with LPC1700

Rev. 01 — 05 January 2010 Application note

Document information

Info	Content
Keywords	LPC1700, BLDC, Motor control, QEI, MCPWM, CMSIS
Abstract	This application note describes the implementation of Brushless DC motor control in the LPC1700 using the dedicated motor control PWM.

Revision history

Rev	Date	Description
01	20100105	Initial version.

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

Brushless DC (BLDC) motors are rapidly gaining popularity. They offer longer life and less maintenance than conventional brushed DC motors. Some other advantages over brushed DC motors and induction motors are: better speed versus torque characteristics, noiseless operation and higher speed ranges. In addition, the ratio of torque delivered to the size of the motor is higher, making them useful in applications where space and weight are critical factors.

This application note demonstrates how to implement six-step commutation or Brushless DC motor control on the LPC1700 family. The LPC1700 is equipped with a dedicated Motor Control PWM which reduces the CPU utilization during motor control while also reducing development time. This application note also intends to be a reference and starting point for motor control system developers using LPC1700 family microcontrollers.

Application note AN10661 “Brushless DC motor control using the LPC2141” is used as a reference for this application note, and may be downloaded from http://www.nxp.com/.

The LPC1700 user manual is referenced, so keep the LPC1700 user manual (UM10360) at hand.

As with the system described in AN10661, this application note shows a complete motor control system solution, not only showing the usage of NXP microcontrollers but also NXP MOSFET driver circuits and MOSFETs.

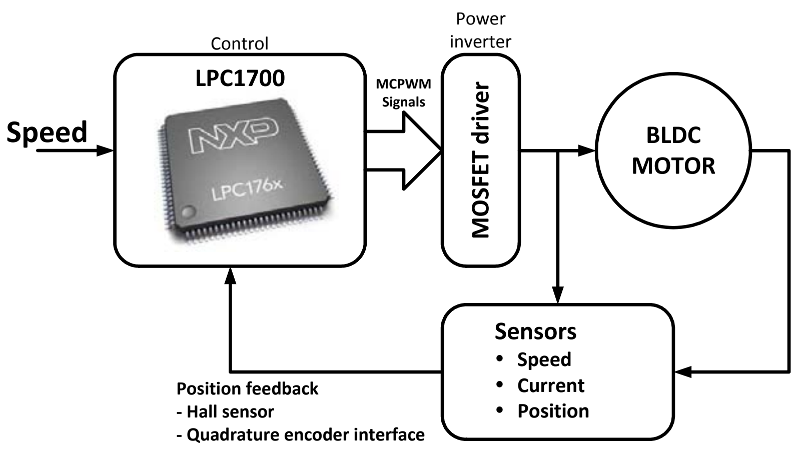


Fig 1. LPC1700 Motor control application block diagram

Fig 1 shows a Motor Control block diagram, which is suitable for a wide variety of applications, including air conditioners, electric pumps, fans, printers, robots, electric bikes, mixers, food processors, blenders, vacuum cleaners, toothbrushes, razors, coffee grinders, etc.

2. The brushless DC motor fundamentals

2.1 The brushless DC motor

Brushless DC motors consist of a permanent magnet rotor with a three-phase stator winding. As the name implies, BLDC motors do not use brushes for commutation; instead, they are electronically commutated. Typically, three Hall sensors (Fig 2) are used to detect the rotor position and commutation is based on these sensor inputs.

In a brushless DC motor, the electromagnets do not move; instead, the permanent magnets rotate and the three-phase stator windings remain static (see Fig 2). This solves the problem of transferring current to a moving rotor. In order to do this, the brush-commutator assembly is replaced by an intelligent electronic “controller”. The controller performs the same power distribution as found in a brushed DC motor, but is uses a solid-state circuit rather than a commutator/brush system.

The speed and torque of the motor depends on the strength of the magnetic field generated by the energized windings of the motor, which in turn depends on the current flow through each winding. Therefore, adjusting the voltage (and current) will change the motor speed.

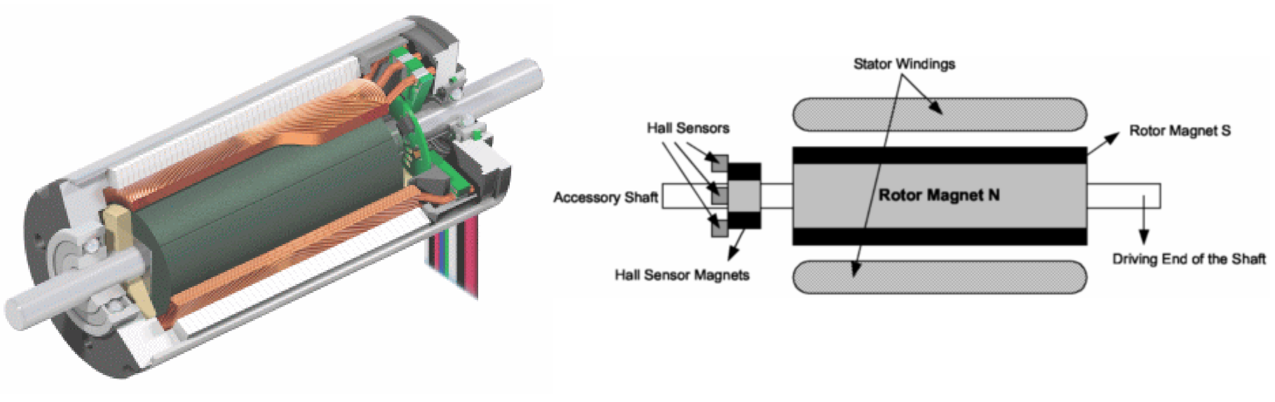


Fig 2. The brushless DC motor

2.2 Electrical commutation

A BLDC motor is driven by voltage strokes coupled with the given rotor position. These voltage strokes must be properly applied to the active phases of the three-phase winding system so that the angle between the stator flux and the rotor flux is kept close to 90° to maximize torque. Therefore, the controller needs some means of determining the rotor's orientation/position (relative to the stator coils.)

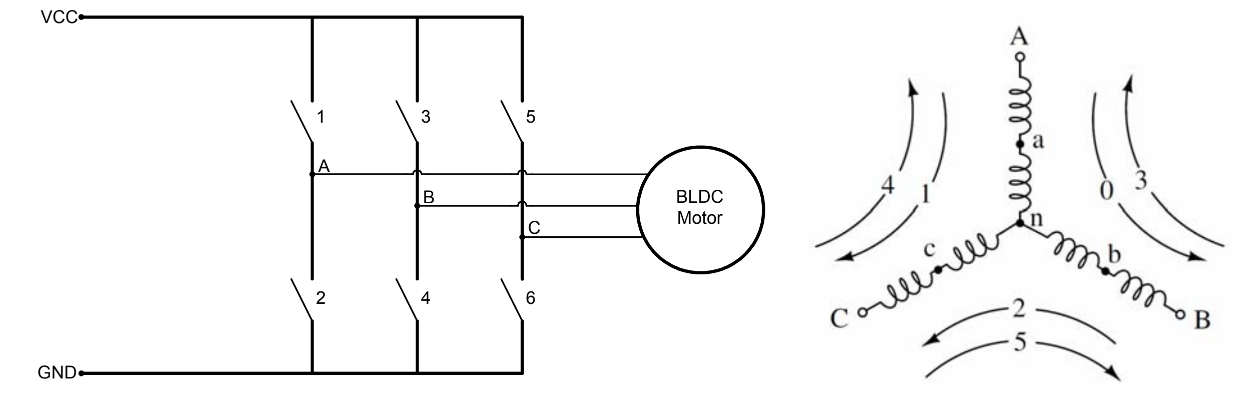


Fig 3. Three phase bridge and coil current direction

Fig 3 depicts a systematic implementation on how to drive the motor coils for a correct motor rotation. The current direction through the coils determines the orientation of the stator flux. By sequentially driving or pulling the current though the coils the rotor will be either pulled or pushed. A BLDC motor is wound in such a way that the current direction in the stator coils will cause an *electrical* revolution by applying it in six steps. As also shown in Fig 3 each phase driver is pushing or pulling current through its phase in two consecutive steps. These steps are shown in Table 1. This is called trapezoidal commutation. Fig 5 shows the relation between the definitions six-step commutation (six Hall sensor edges H1, H2 and H3), block commutation (i_a, i_b, i_c) and trapezoidal commutation (e_a, e_b, e_c).

Table 1. Switching sequence

Sequence number	Switching interval	Phase current			Switch closed	
		A	B	C		
0	0° - 60°	+	-	OFF	1	4
1	60° - 120°	+	OFF	-	1	6
2	120° - 180°	OFF	+	-	3	6
3	180° - 240°	-	+	OFF	3	2
4	240° - 300°	-	OFF	+	5	2
5	300° - 360°	OFF	-	+	5	4

2.3 Revolution speed control

Varying the voltage across the motor can simply control the rotor speed. This can be achieved by pulse width modulation (PWM) of the phase voltage. By increasing or decreasing the duty-cycle, more or less current per commutation step will flow through the stator coils. This affects the stator flux and flux density, which changes the force between the rotor and stator.

This means that the rotation speed is determined by the load of the rotor, the current during each phase, and the voltage applied.

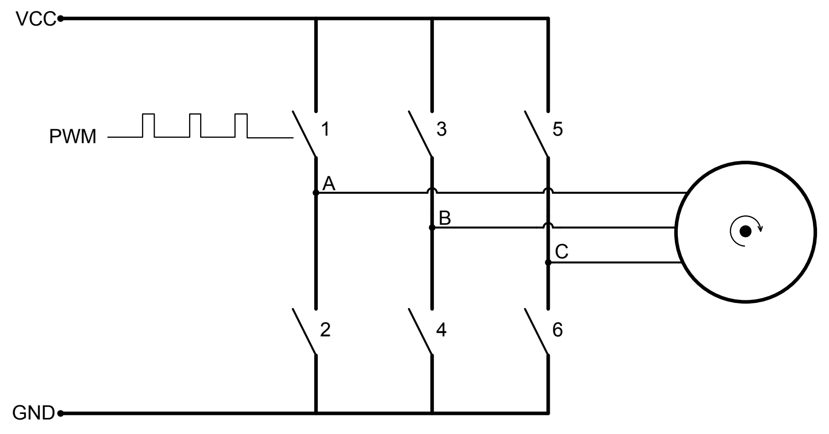


Fig 4. Speed control through PWM

2.4 Torque control

Just like speed control, torque is controlled by the amount of the current through the stator coils. For maximum torque the angle between the stator and rotor flux should be kept at 90°. With trapezoidal commutation, the control resolution is 60° and the angle between the stator and rotor flux is from -30° to +30°, which introduces a torque ripple.

2.5 Position feedback

The rotor position feedback can be accomplished though a couple of techniques. Most commonly is the hall sensor feedback, but other techniques include using an encoder or even eliminating sensors entirely. This application note will only focus on the hall sensor feedback and encoder feedback and will not explore sensorless operation.

2.5.1 Hall sensor feedback

The hall sensors are placed such that they generate an edge at each switching interval as explained in Chapter 2.2. This makes it very easy to determine the current rotor orientation, and to activate each phase in the right sequence.

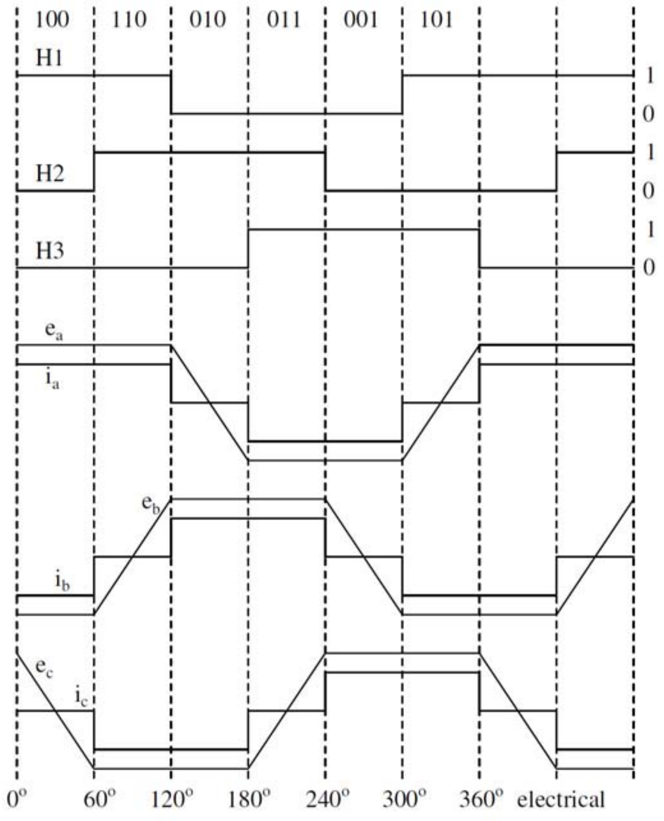


Fig 5. Trapezoidal control with Hall sensor feedback

2.5.2 Encoder feedback

The most commonly used encoder is the quadrature encoder. For more detailed information on quadrature encoders, please see chapter 3.2.

3. The LPC1700

The LPC1700 is an ARM Cortex-M3 based microcontroller for embedded applications requiring a high level of integration and low power dissipation. The ARM Cortex-M3 is a next generation core that offers system enhancements such as modernized debug features and a higher level of support block integration.

The LPC1700 operates at up to 120 MHz CPU frequency. The peripheral complement of the LPC1700 includes up to 512 kB of flash memory, up to 64 kB of SRAM, Ethernet MAC, a USB interface that can be configured as either Host, Device, or OTG, 8 channel general purpose DMA controller, 4 UARTs, 2 CAN channels, 2 SSP controllers, SPI interface, 3 I^2C interfaces, 2-input plus 2-output I2S interface, 8 channel 12-bit ADC, 10-bit DAC, motor control PWM, Quadrature Encoder interface, 4 general purpose 32-bit timers, 6-output general purpose PWM, ultra-low power RTC with separate battery supply, and up to 70 general purpose I/O pins.

This application note will deal with the motor control specific peripherals, which are the motor control, PWM and the Quadrature Encoder Interface.

3.1 LPC1700 Motor Control PWM (MCPWM)

The NXP LPC1700 family is equipped with a dedicated Motor Control PWM (MCPWM). This MCPWM is optimized for three-phase AC and DC motor control applications.

The motor control PWM architecture is designed such that there are three independent channels, of which each channel controls a modulated output pair with opposite polarities. All channels can be set-up and controlled independently, but can be interconnected using the ‘Three phase DC mode’ or the ‘Three phase AC mode’.

Each channel includes:

- a 32-bit Timer/Counter (TC)
- a 32-bit Limit register (LIM)
- a 32-bit Match register (MAT)
- a 10-bit dead-time register (DT) and an associated 10-bit dead-time counter
- a 32-bit capture register (CAP)
- two modulated outputs (MCOA and MCOB) with opposite polarities
- a period interrupt, a pulse-width interrupt, and a capture interrupt

In total the MCPWM has 10 I/O lines to fully control the power inverter stage in the motor control system. Each channel includes a PWM output pair, called MCOAx and MCOBx, and an input MCIx, where x is the channel number ranging from 0 to 2. The MCPWM also has a low-active Fast Abort pin, MCABORT. This pin will put the output pins to the passive state and can be used for a safety state.

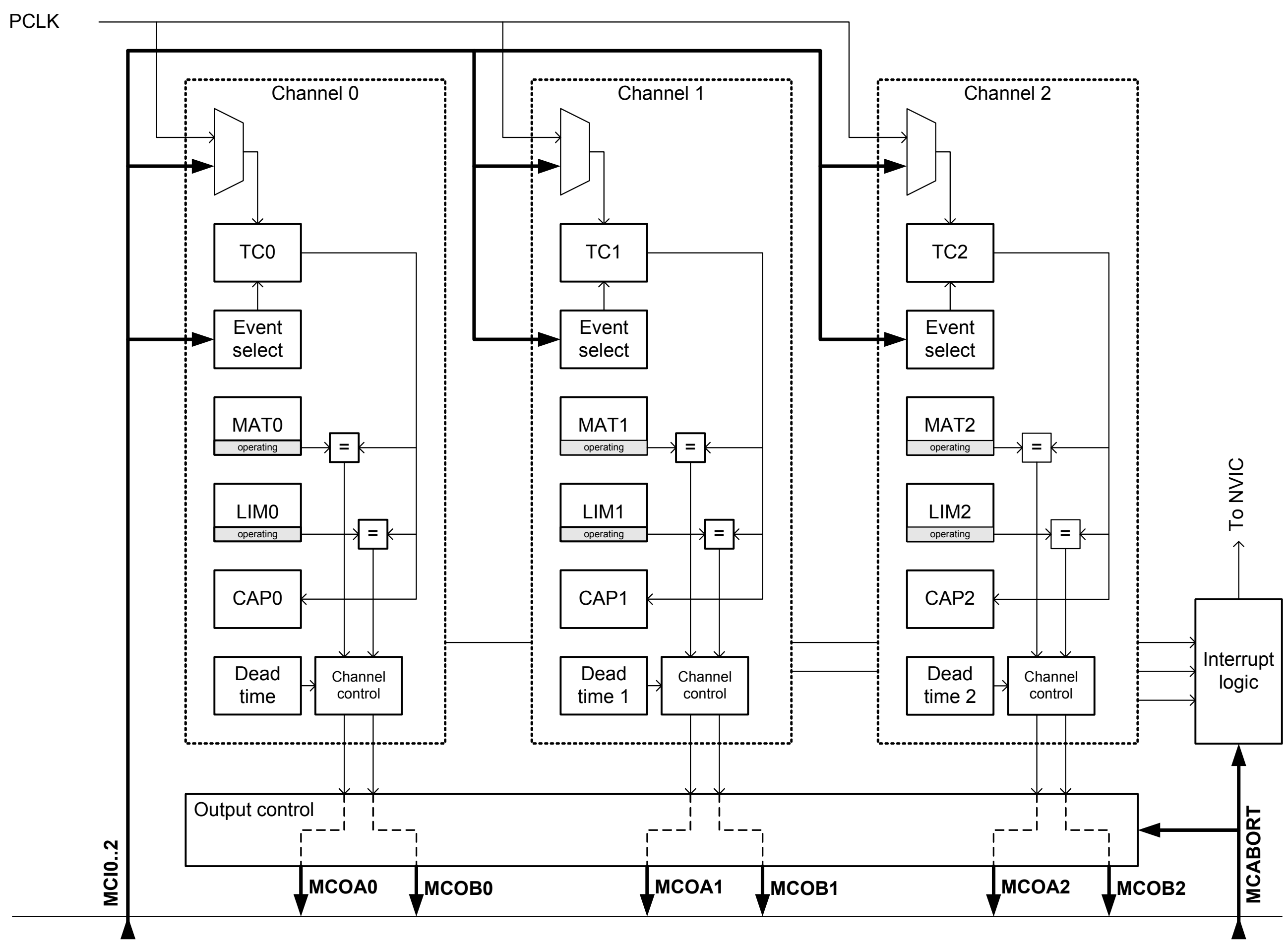


Fig 6. Simplified MCPWM block diagram

Fig 6 depicts a simplified block diagram of the MCPWM. The three independent channels are shown in the block diagram. Each channel has its own timer counter (TCn[1]) which can be clocked either by the peripheral clock (PCLK) or the motor control input pins 0 to 2 (MCI0..2). The match registers (MATn) and the Limit registers (LIMn), together with the dead-time generator, are responsible for the state of the channel outputs. Valid states for the channel outputs can be *'passive'* or *'active'* and the output control determines whether these states are HIGH or LOW.

For the detailed MCPWM block diagram, please see the LPC1700 User Manual (UM10360).

3.1.1 Operation modes

The MCPWM supports many different modes of operation. These are briefly summarized in the following section. For more detail on these modes, please refer to the LPC1700 User Manual.

1. n indicates the channel number, 0 to 2

3.1.1.1 PWM

The first mode of operation is the regular PWM mode. Within this mode, one can select edge- or centre-aligned PWM waveform generation. For both edge- and centre-aligned waveforms, dead-time delay can be implemented on the passive-to-active states.

Please note that after reset all *A* outputs are set to 'passive' state and the mapping of 'passive' is LOW.

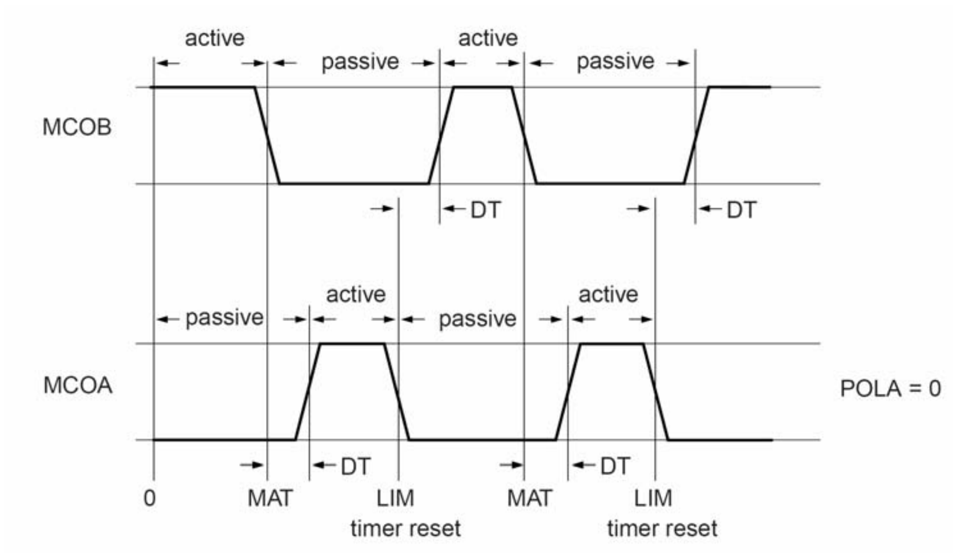


Fig 7. Dead time generation on the passive to active state

3.1.1.2 External event counting

The second mode is the 'External event counting' mode. Instead of using the PCLK, the external MCIx trigger pin will cause the TC to rise and/or fall.

3.1.1.3 Three-phase DC mode

In the 'Three-phase DC mode', the internal MCOA0 signal can be routed to any or all of the MCO outputs. The current commutation pattern register (MCCP) determines to which output the MCOA0 signal is routed.

Fig 8 shows with the red dashed line the change in signal flow with respect to the normal PWM operation mode.

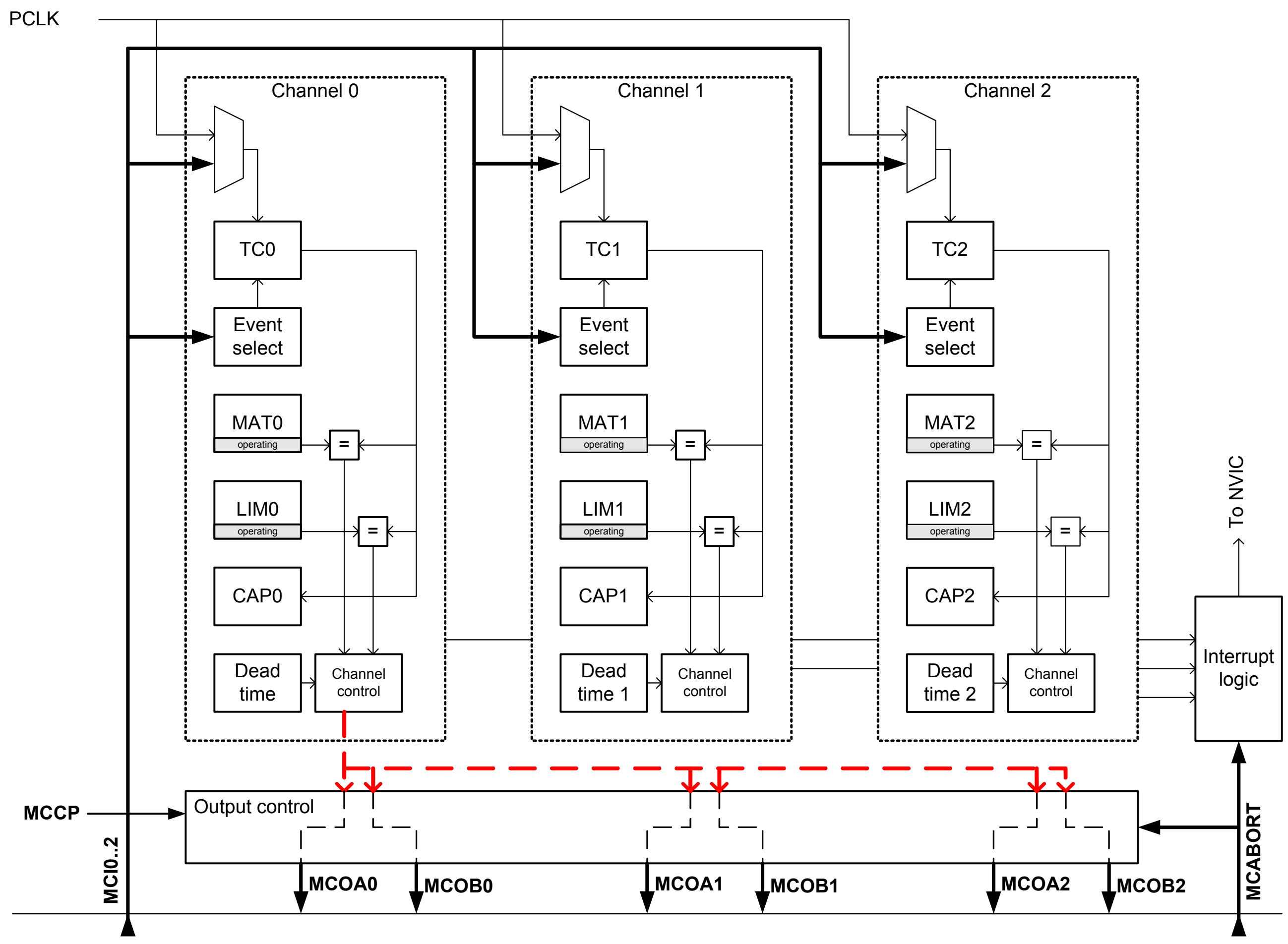


Fig 8. Simplified MCPWM block diagram (Three phase DC mode)

3.1.1.2 Three-phase AC mode

The ‘Three-phase AC mode’ compares the MAT register of all channels to the channel 0 Timer counter (TC0).

Fig 9 shows the three-phase AC mode operation.

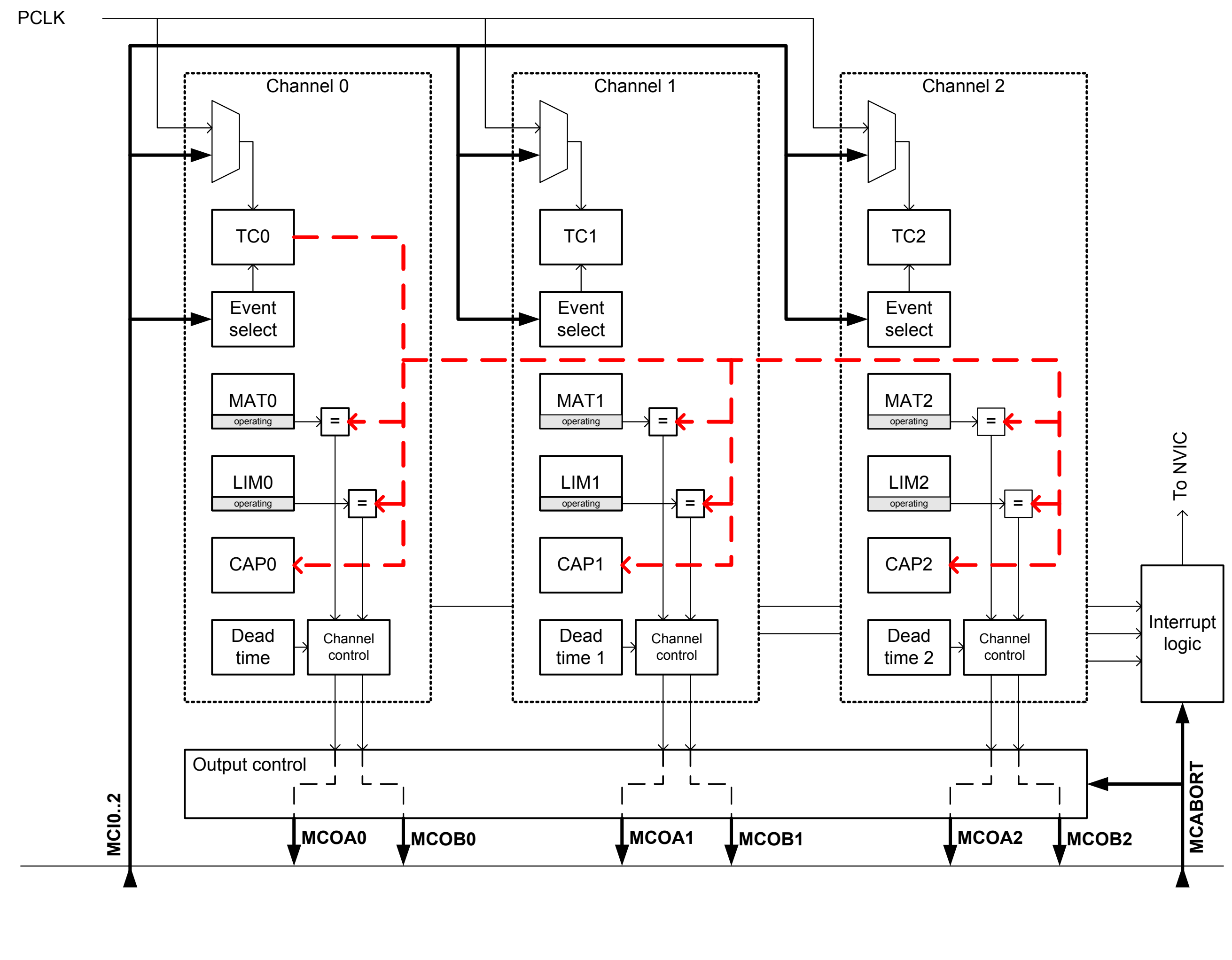


Fig 9. Simplified MCPWM block diagram (Three phase AC mode)

3.1.1.3 Fast Abort

The last operation mode is the Fast Abort mode. This mode is entered when the active-low pin MCABORT is externally pulled low. All MCOAx and MCOBx pins will be put in the ‘passive’ state. The polarity of these pins can be set using the POLA bits in the MCPWM control register (MCCON).

3.1.2 MCPWM pin description

As described in the previous chapters, the MCPWM has in total 10 I/O pins used in operation. See Table 2 for a description of these 10 pins.

Table 2. MCPWM pin description

Function Name	Pin name	Description
MCOA0	P1.19	MCPWM channel 0 output A
MCOB0	P1.22	MCPWM channel 0 output B
MCOA1	P1.25	MCPWM channel 1 output A
MCOB1	P1.26	MCPWM channel 1 output B
MCOA2	P1.28	MCPWM channel 2 output A
MCOB2	P1.29	MCPWM channel 2 output B
MCI0/MCFB0[1]	P1.20	MCPWM input 0 or MCPWM Feedback 0
MCI1/MCFB1[1]	P1.23	MCPWM input 1 or MCPWM Feedback 1
MCI2/MCFB2[1]	P1.24	MCPWM input 2 or MCPWM Feedback 2
$\overline{\text{MCABORT}}$	P1.21	Low-active fast abort

[1] Please note that the MCPWM input pins are shared with the QEI input pins

3.2 Quadrature Encoder Interface (QEI)

The quadrature encoder, also known as a 2-channel incremental encoder, converts angular displacement into two pulse signals. By monitoring both the number of pulses and the relative phase of the two signals, you can track the position, direction of rotation, and velocity. In addition, a third channel, or index signal, can be used to reset the position counter. This quadrature encoder interface module decodes the digital pulses from a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture the velocity of the encoder wheel.

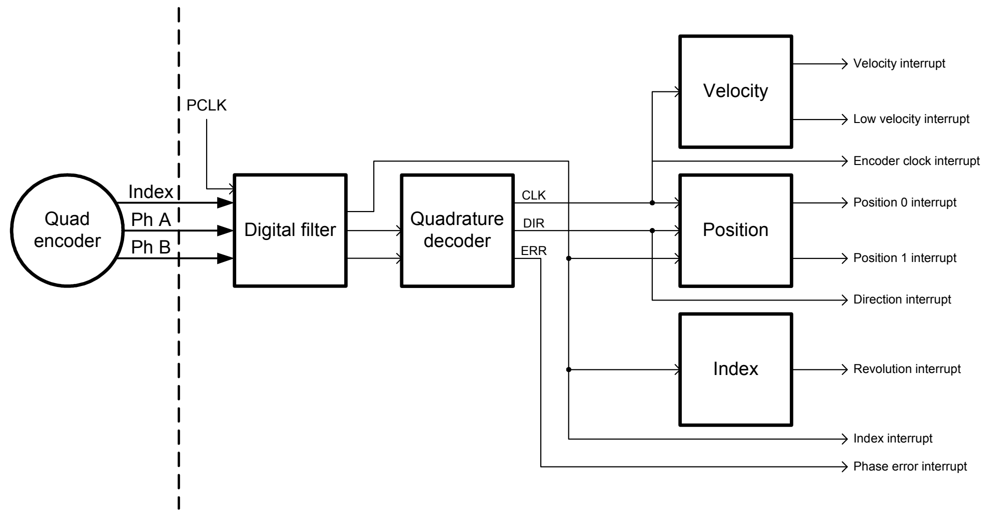


Fig 10. Simplified QEI block diagram

3.2.1 QEI pin description

As depicted in Fig 10 the three signals from the quadrature encoder (Phase A, Phase B and Index) are fed into the QEI. Pins used for these signals are:

Table 3. QEI pin description[1]

Function Name	Pin name	Description
MCFB0	P1.20	Used as the Phase A (PHA) input to the QEI
MCFB1	P1.23	Used as the Phase B (PHB) input to the QEI
MCFB2	P1.24	Used as the Index (IDX) input to the QEI

[1] Please note that the QEI input pins are shared with the MCPWM input pins.

For a detailed description on how the QEI is implemented in the application, see the following chapters. A detailed description of the usage can be found in the LPC1700 User Manual (UM10360).

4. Application setup

This chapter will describe how the application is setup.

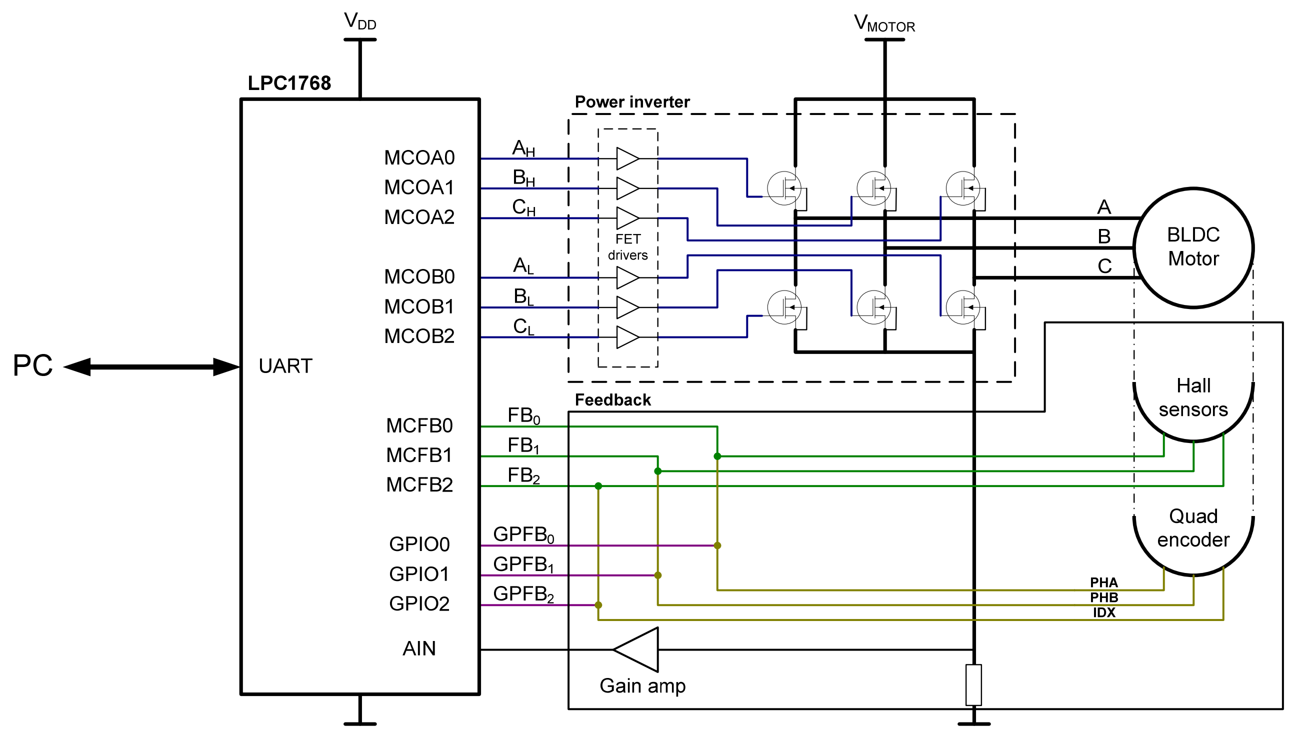


Fig 11. LPC1700 Motor control application block diagram

4.1 Connections

This application uses the MCPWM peripheral for controlling the power inverter. As Fig 11 depicts, the MCPWM A channels are used for driving the High-side switching FETs and the MCPWM B channels are driving the Low-side switching FETs.

For rotor orientation feedback, either a Hall sensor or a quadrature encoder can be chosen. The MCPWM input pins (MCI0..2) and the QEI pins (PHA, PHB and IDX) are shared pins, called MCFB0..2. In this application, no hardware is implemented to detect an ‘un-safe’ state for triggering the $\overline{\text{MCABORT}}$ pin.

Table 4. Application connections

Function Name	Pin name	Signal	Signal Purpose
MCOA0	P1.19	A_H	Motor phase A **high** side FET drive
MCOB0	P1.22	A_L	Motor phase A **low** side FET drive
MCOA1	P1.25	B_H	Motor phase B **high** side FET drive
MCOB1	P1.26	B_L	Motor phase B **low** side FET drive
MCOA2	P1.28	C_H	Motor phase C **high** side FET drive
MCOB2	P1.29	C_L	Motor phase C **low** side FET drive
MCFB0	P1.20	FB0	Hall0 or QE PHA feedback input
MCFB1	P1.23	FB_1	Hall1 or QE PHB feedback input
MCFB2	P1.24	FB_2	Hall2 or QE IDX feedback input
AD0.5	P1.31	A_{IN}	Motor current measurement input
GPIO0	P0.16	Hall0	Hall0 or QE PHA feedback input
GPIO1	P0.17	Hall1	Hall1 or QE PHB feedback input
GPIO2	P0.18	Hall2	Hall2 or QE IDX feedback input
$\overline{\text{MCABORT}}$	P1.21	Not Used	

As Fig 11 shows, the feedback pins are connected to both the MCFB pins and some GPIO pins, this is because of the MCPWM.1 errata in the LPC1700 errata sheet. Please see this errata for more information.

4.2 Power inverter

The power inverter in this application note is based on the same power inverter stage as described in AN10661, and is briefly explained in the following paragraphs

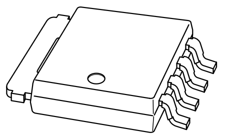
4.2.1 MOSFET selection

The NXP Semiconductors PH20100S N-channel TrenchMOS logic level FET is used for this system. It is chosen in relation with the selected motor, which is supplied with 24 V.

For a 24 V - supplied motor, the MOSFET V_{DS} needs to be at least 40 V, while the drain current needs to be high enough to deal with the motor (starting) current. The latter is already reduced thanks to a soft-acceleration mechanism (in small steps up towards the required speed) implemented in software. The PH20100S can deal with a maximum drain current of 34.3 Amps and a peak current of 137 Amps and is available in an SMD SOT669 (LFPAK) package

4.2.2 MOSFET driver selection

MOSFET drivers are needed to raise the controller’s output signal (driving the MOSFET) to the motor supply voltage level. In this application note we selected the PMD3001D and the PMGD400UN from NXP Semiconductors, as shown in Fig 12.



M1, M2 = 2 x LFPAK: PH20100S

Qa, Qb = 2 x PMD3001D

M1s = 1 x PMGD400UN

Dbst = 1 x BAS16VY

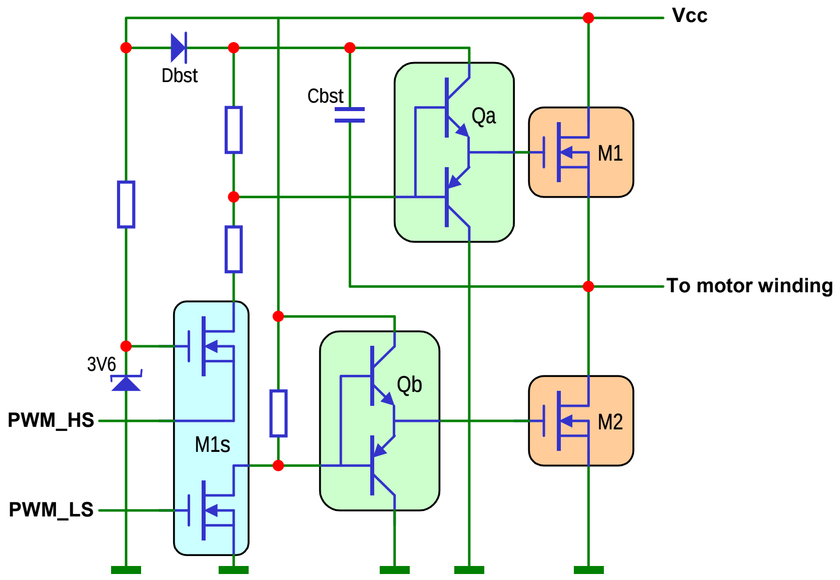


Fig 12. Simplified MOSFET – driver diagram for low and high side driver

4.3 BLDC motor selection

The MOSFETs used in this power inverter are rated at 12 V with a switching frequency of 20 kHz (50 μs) to have a safe operating current of about 40 A. Therefore this board is capable of driving a motor of approximately 480 W.

During development of this application note, a Maxon EC32 80W with Hall sensors and quadrature encoder extension was used.

5. Software

The software written for this application note is based on the Cortex Microcontroller Software Interface Standard (CMSIS) and therefore uses the driver CMSIS-Compliant standard peripheral firmware driver library. The latest version of this driver library can be found on the NXP microcontrollers support page.

Usage of the relevant driver library files and the application dedicated sources will be explained in this chapter.

Keil ARM-MDK version 4.01 is used as IDE during development of the LPC1768 firmware.

A UART is used as a feedback mechanism and parameter control interface allowing for your preferred terminal program to be used.

5.1 Folders and files

All files needed for this application note is are arranged in the following order

Table 5. Top folder structure

Folder	Description
apps_src	All application dedicated source files
Core	Core specific driver and initialization files, CMSIS compliant
Documentation	Documentation useful for this application note
Drivers	CMSIS compliant driver library
Lst	Destination folder for all listing files compiled by the compiler
Obj	Destination folder for all object files compiled by the compiler
LPC1700_BLDC.uvproj	Keil uVision4 project file

5.1.1 Firmware source

The application-dedicated source files written for this application notes can be found as indicated in Table 5 in the apps_src folder.

5.1.1.1 BLDC.c

The BLDC.c source file incorporates the initialization, controlling and interrupt handling of the BLDC motor control. Table 6 describes all functions in this file.

Table 6. Functions description in BLDC.c

Function	Description
`void BLDC_init (void)`	BLDC control initialization - MCPWM initialization - Hall feedback initialization through MCFB pins or GPIO - QEI initialization, if selected
`void BLDC_Enable (void)`	Enable the BLDC motor control
`void BLDC_setDuty (unsigned int duty)`	Set the PWM duty cycle
`void BLDC_calcRPM (PIDstr *ptr)`	Calculate the current RPM
`void BLDC_break (void)`	Break the BLDC motor
`void BLDC_commutate (unsigned char step)`	Commutate the BLDC motor according the right commutation sequence
`void MCPWM_IRQHandler (void)`	MCPWM interrupt handler function, for the Hall sensor feedback if selected
`void EINT3_IRQHandler (void)`	The external interrupt 3 or GPIO interrupt handler function, in this case it handles the Hall sensor feedback if selected through the GPIO interrupts
`void QEI_IRQHandler (void)`	QEI interrupt handler function, handles the QEI position compare interrupts

5.1.1.2 Main.c

The `Main.c` is the application entry file. Here all application-specific initializations are made and a small pre-emptive scheduler is started for task scheduling with a 1 ms resolution. It also includes the retarget function enabling usage of the `printf` function.

Table 7. Functions description in Main.c

Function	Description
`void Systick_Handler (void)`	The core systick interrupt handler, used for 1 ms counts throughout the system
`int sendchar (int c)`	Retargeting function for `printf` usage, which will be collected and send over the UART
`void UART0_IRQHandler (void)`	UART0 interrupt handler, it directs to the UART driver library.
`void UART0_IntReceive (void)`	UART0 receive callback function, handles all incoming UART data and put it on the ring buffer
`void Call_1ms(void)`	The 1 ms scheduler Tasks: - Start PID calculation - If motor is enabled but no speed, hard commutate
`void Call_10ms(void)`	The 10 ms scheduler Tasks: - Start processing incoming UART data on ring buffer
`void Call_100ms(void)`	The 100 ms scheduler Tasks: - Print the current RPM
`void Call_1s(void)`	The 1 s scheduler
`void initApplication(void)`	Initialize the application - Systick configuration - LED configuration - UART configuration - BLDC Motor control configuration
`int main(void)`	Main program entry function - Initialize system - Initialize the application - Enable the BLDC motor - Run scheduler in endless loop

5.1.1.3 PID.c

The PID consists only of one function, `PID_calc_cntr` which actually is a PI controller. The input of this function is a pointer to a PIDstr structure. This structure contains all variables needed for the PI controller to calculate the manipulated value or output.

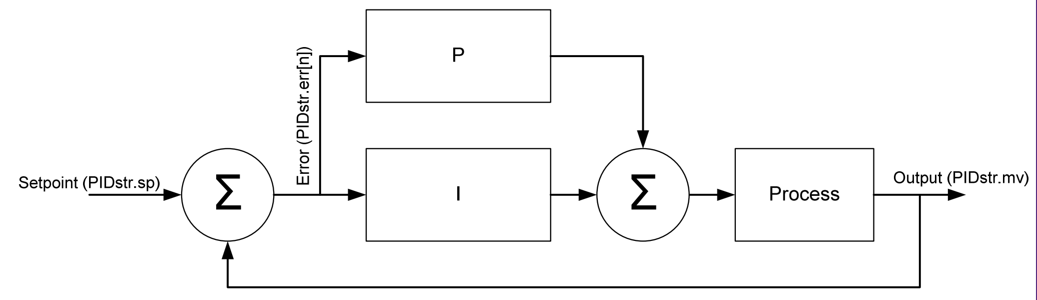


Fig 13. PI controller

5.1.1.4 Retarget.c

Retarget.c is the Keil standard solution for linking the stdio.h files to the low level processor dependent drivers.

By defining the `int sendchar (int c)` function in the application code, as in this application is done in the main.c, the stdio.h functions as printf can be used.

5.1.1.5 UART_ProcessData.c

The functions in the UART_ProcessData c source file takes care for proper UART command processing

Table 8. Functions description in UART_ProcessData.c

Function	Description
`void UART_processError (void)`	UART process error handler, gives '`Invalid command`' response to user.
`void UART_processMessage (UART_RING_BUFFER_T *rb, PIDstr *pid, uint8_t loc)`	Processes the incoming commands

The commands which can be used during operation are described in Table 9.

Table 9. UART command set[1]

Command	Description
start	Starts/Enables the motor
Stop	Stops the motor, but keeps it free running and therefore will not brake
break	Stops the motor and actively brake
speed=nnnn	Give the RPM setpoint,
Poles	The number of motor pole pairs
dir=n	Set the motor direction, 0 = CW and 1 = CCW
p=n.nn	Set the P value, point is floating
i=n.nn	Set the I value, point is floating

[1] All commands are case sensitive.

The UART_processMessage function is set to be very flexible and the user can easily add or delete commands from the command set.

5.1.1.6 Controlling structure PIDstr

```
typedef volatile struct _PIDstr {
  double   p;   /**< The PID Product-value */
  Double   i;   /**< The PID Integrating-value */
  Double   d;   /**< The PID Diff-value, not used but for future purpose */
  DWORD    sp;  /**< Setpoint in mechanical RPM */
  DWORD    pv;  /**< Process value, dt in sys counts, used for RPM calc */
  Double   err[3];    /**< Error in PI(D) calculations */
  DWORD    mv;  /**< Manipulated value, output of the PI(D) controller */
  BYTE     HALstate;  /**< Current HAL state              */
  DWORD    CMT_CNT;   /**< Commutation counter, motor commutated check*/
  DWORD    CMT_step;  /**< Current commutation step */
  DWORD    RPM;       /**< Current motor mechanical RPM */
  BYTE     Enable;    /**< Motor enable, 0 = DISABLE, 1 = ENABLE  */
  BYTE     Direction; /**< Motor rotation direction, Clockwise = CW or Counter
                         Clockwise = CCW */
  BYTE     Brake;     /**< Motor electrical break, 0 = DISABLE, 1 = ENABLE */
  DWORD    Period;    /**< Switching frequency of the phase drives */
  BYTE     Poles;     /**< Number of Pole-pairs in the motor */
  DWORD         Tick_cur;  /**< Current tick value, for RPM calculations */
  DWORD         Tick_old;  /**< Current tick value, for RPM calculations */
  DWORD         Tick_new;  /**< Current tick value, for RPM calculations */
} PIDstr;
```

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

7. Contents

Date of release: 05 January 2010
Document identifier: AN10898_1