PmodHB5 Library Reference Manual

Revision: February 16, 2012

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

This document describes the structure and the usage of the library functions that manage the DC motors using HB5 and HB3 Pmods. The purpose of this HB_MotorLib library is to provide user access to HB5 and HB3 Pmods functionality.

This approach allows the control of HB5 and HB3 Pmods inserted into any of the following connectors:

Motor	Connector
_MOTOR1	JH 7-12
_MOTOR2	JD 1-6
_MOTOR3	JD 7-12
_MOTOR4	JE 7-12

*Note: Assumed board in use is the **Cerebot MX4cK***

The number of the motor is given by the number of Output Compare and Input Capture module in the specified connector.
The header file HB_MotorLib.h provides required pin definitions for all 4 possible motors.

Read PmodHB5_RevD_rm.pdf or PmodHB3_rm_RevD.pdf for HB5 or HB3 Pmods specifications.

Overview

DC motor is driven by implementing a PWM on the EN pin and by setting the direction on DIR pin of the Pmod. According to a specific duty factor, PWMs are generated using Output Compare modules. In order to generate the PWM, the secondary Compare Register is set to a value between 0 and max period (0xFFFF), according to the desired duty: 0xFFFF * Duty, where Duty is between 0 and 1.

General Approach

Output Compare Module Configuration

Plib macro used:

OpenOC1(OC_ON | OC_TIMER_MODE16 | OC_PWM_FALUT_PIN_DISABLE | OC_TIMER2_SRC, hwPwm, hwPwm);

- OC_ON – OC1 module is ON
- OC_TIMER_MODE16 – 16 bits mode
- OC_PWM_FAULT_PIN_DISABLE – Fault pin disable
- OC_TIMER2_SRC – Timer 2 as source
- Primary and secondary register set to specified value stored in hwPwm

The provided sample is for OC1. For others OC modules configuration is similar.

The Output Compare is configured using PLIB macros. Still, for OC1 (in HB_MotorConfigure1 function) an example is provided (in the commented code) of how to configure OC using SFRs.

Timer2 Configuration

Plib macro used:

OpenTimer2(T2_ON, cntPulsePWMMax);

- Period cntlPulsePWMMax (= 0xFFFF)
  - Peripheral Bus Freq is 1/8 * SYSCLK = 8 MHz, so the Timer 2 period is (0xFFFF + 1) * (1/8 Mhz) = 8.192ms.
- T2_ON – Timer 2 is ON

Real Time

In order to compute the RPM (rotations per minute) a Real Time Counter is needed, incremented at a certain frequency (pace). The HB_MotorLib module defines a counter and provides Timer4 to increment it at 100 us. Since most of the user's applications are using a timer to control the interface (for example buttons, display,), user may choose to avoid the usage of Timer4 only for real time counter. This is done by defining REALTIME_CALLEDBYUSER (using #define). In this situation it is user's responsibility to increment the real time counter, and this is done by including mIncrementRealTime macro in the specific ISR code. Also, it is user's responsibility to set the REAL_TIME_FACTOR (the ratio between 1 s and specific period) according to the frequency on which the specific ISR is called.

Timer4 Configuration

Plib macro used:

OpenTimer4(T4_ON | T4_SOURCE_INT | T4_PS_1_8, 99);

ConfigIntTimer4(T4_INT_ON | T4_INT_PRIOR_7 | T4_INT_SUB_PRIOR_3);

- T4_ON – Timer 4 is ON
- T4_PS_1_8 - (Prescaler 1/8) so the frequency is 1/8 of Peripheral Bus Frequency, which is 1/8 of SYSCLK (64 MHz) = 1 MHz
- Period = 00, so Timer period = (99 + 1) * 1/1MHz = 100 us
- T4_SOURCE_INT – Timer4 triggers interrupt
- T4_INT_ON_T4 – interrupt is ON
- T4_INT_PRIOR_7 , T4_INT_SUB_PRIOUR_3 – timer interrupt priority level 7, subpriority level 3

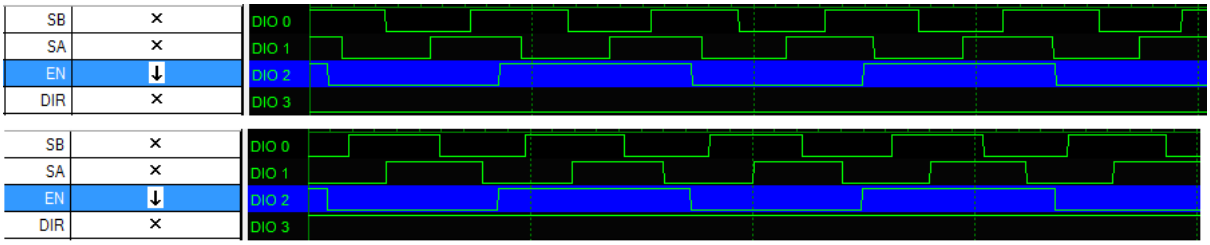
Reading Motor Speed and Direction

The library provides functionality to obtain the direction and rotation speed of the DC motor using Hall sensor signals SA and SB provided by the PmodHB5.

Acquiring Data

For this, Input Capture module is used on SA pin. It is configured to generate interrupts on every edge. This approach offers the possibility to deal with situations when the motor has low speed and just passes back and forth to a sensor position.
Then, in the ISR (interrupt service routine) the following actions are taken:

a. Direction is detected.



b. According to the direction, a **Rotation Counter** is incremented / decremented, thus counting SA transitions.

Input Capture Module Configuration

Plib macro used:

OpenCapture1(IC_EVERY_EDGE | IC_INT_1CAPTURE | IC_TIMER2_SRC | IC_ON);
ConfigIntCapture1(IC_INT_ON | IC_INT_PRIOR_7 |IC_INT_SUB_PRIOR3);

- IC_EVERY_EDGE – capture on every edge
- IC_TIMER2_SRC – Timer 2 is used as source
- IC_INT_1CAPTURE – IC triggers interrupt
- IC_ON – IC module is ON
- IC_INT_ON – IC interrupt is on
- IC_INT_PRIOR7, IC_INT_SUB_PRIOR3 – IC interrupt priority level 7, subpriority level 3

The IC modules are using as source Timer2, which has a period of 8.192 ms.

Data acquired will be later used for computing the speed. The provided example is for IC1. Configuration is similar for other IC modules.

Note: HB_MotorLib functionality is only using the Input Capture interrupts to count SA transitions. It is not using the core of Input Capture module (buffer values, for example). So, an external interrupt could have been used (the only difference would have been the way to use it on both edges). But no external interrupts are available on most of Pmods on positions corresponding to Hall sensors, so Input Capture modules were used.

Computing RPM (rotations per minute)

Computing the RPM is calculated using the following formula:
(<Rotation Counter difference> / (<Real Time difference> * 3 * 2)) * 60 * REAL_TIME_FACTOR / reductionMotor1

- <Counter difference> is the difference between the actual counter value and the saved (at the previous speed calculation) counter value.
- <Real Time difference> is the difference between the actual real time counter value and the saved (at the previous speed calculation) real time counter value.
- Divided by 3 because there are 3 cycles of Hall sensor / motor rotation
- Divided by 2 because both rising and falling transitions of SA are counted

- Multiplied by REAL_TIME_FACTOR because this is the ratio between Real Time counter values and seconds
- Multiplied by 60 because RPM is referring to minutes.
- Divided by reduction factor because the hall sensor information refers to engine rotations and the RPM refers to final rotations (after reduction).

After the speed is computed the rotation counter and real time counter are saved to be used next time when the speed will be computed.

Note:This approach is using the time between two consecutive computation of the speed as the time base for the speed. So, it's user responsibility to compute the speed at appropriate time intervals (not too often and not too rare).

Using HB_MotorLib in Applications

Include the Needed Files in the Project

File	Containing
HB_MotorLib_Defs.h	Definitions used by HB_MotorLib functions (to be edited)
HB_MotorLib.h	Declarations and pins configurations used for HB_MotorLib functions. No need to edit.
HB_MotorLib.c	Definitions for HB_MotorLib functionality. No need to edit.

Define Motors in Use

Edit HB_MotorLib_Defs.h and, according to the motors you want to use, define any of the following:
#define _MOTOR1
#define _MOTOR2
#define _MOTOR3
#define _MOTOR4

Real Time Counter Management

If you already use a timer ISR and want to use it for incrementing the real time counter as well, do the following:

- Edit HB_Motor_defs.h and #define REALTIME_CALLEDBYUSER
- Define REAL_TIME_FACTOR to its corresponding value

Call the Initialization Function

HB_MotorConfigureX(dDuty, fFalse);

Call the Command Motor Function

HB_MotorCommandX(dDuty, fTrue);

Read the Motor RPM

HB_MotorReactionX(&dDCMotorReactionContext, dDuty);

Note: DCMotorReactionContext structure also contains possible useful information: the rotation counter (wRotationCounter), saved rotation counter (wSavedRotationCounter) and saved real time counter (wSavedRealTime).

HB_MotorLib Library Functions

For each motor the following functions are defined (X stands for the number of motor 1-4):

void HB_MotorConfigureX(double dDutySubunit, BOOL fOnlyPWM)

Parameters:
dDutySubunit – the initial value for the duty factor (value between 0 and 1)
fOnlyPWM – fTrue if only PWM functionality should be initialized, or fFalse if all functionality should be initialized

Initializes DC motor X functionality:
- Configures pin directions (if(fOnlyPWM == fFalse))
- Configures IC module (if(fOnlyPWM == fFalse))
- Configures OC module
- If necessary, configures Timer 4 interrupt, used for real time counter

void HB_MotorCommandX(double dDutySubunit, BOOL fDir)

Parameters:
dDutySubunit – the value for the duty factor (value between 0 and 1)
fDir – the direction to be sent to the motor

Commands the motor according to the provided duty and direction. The corresponding Output Compare (OC) module is programmed so that it generates a PWM corresponding to the duty. The function also ensures a minimum of 1 ms of low on EN before and after direction changes.

void HB_MotorReactionX(DCMotorReactionContext * pDCMotorReactionContext, double dDuty)

Parameters:
pDCMotorReactionContext – NULL when function is only called to manage rotation counter, otherwise, it is a pointer to a structure that maintains reaction data (speed, counter, timestamps). Acts as an output parameter when speed is read.
dDuty – This is used to determine the sign of RPM

This function's purpose is to detect the direction / manage rotation counter used to detect the speed of the motor, and to read the speed with the reduction in consideration.