eTPU Motor Control Demo

BLDC Motor with Speed Control Loop

June 2004

Launched by Motorola

freescale semiconductor

Overview

BLDC Motor with Speed Control Loop

Launched by Motorola

freescale semiconductor

BLDC Motor with Speed Loop - Demo Application

eTPU is used as a motor control coprocessor – eTPU drives the motor independently on the CPU

CPU Tasks:

- Initialize eTPU and other peripherals
- Drive application state machine (INIT/ON/STOP/OFF/FAULT)
- Write Required Speed (from buttons or FreeMaster) to the eTPU

eTPU Tasks (all time-critical tasks):

- Generate 3 phases of top-bottom PWM pairs with dead-time
- Process Hall sensor signals, and commutate PWM phases
- Calculate actual motor speed
- Control PWM duty-cycle based on actual and required motor speed

eTPU Processing

Motor control algorithm is built using several eTPU functions, that cooperate together

PWMM + PWMC (PWM Generation)

- Generation of PWM signals for switching IGBT transistors
- 3 phases = 3 complementary top/bottom pairs with dead-times

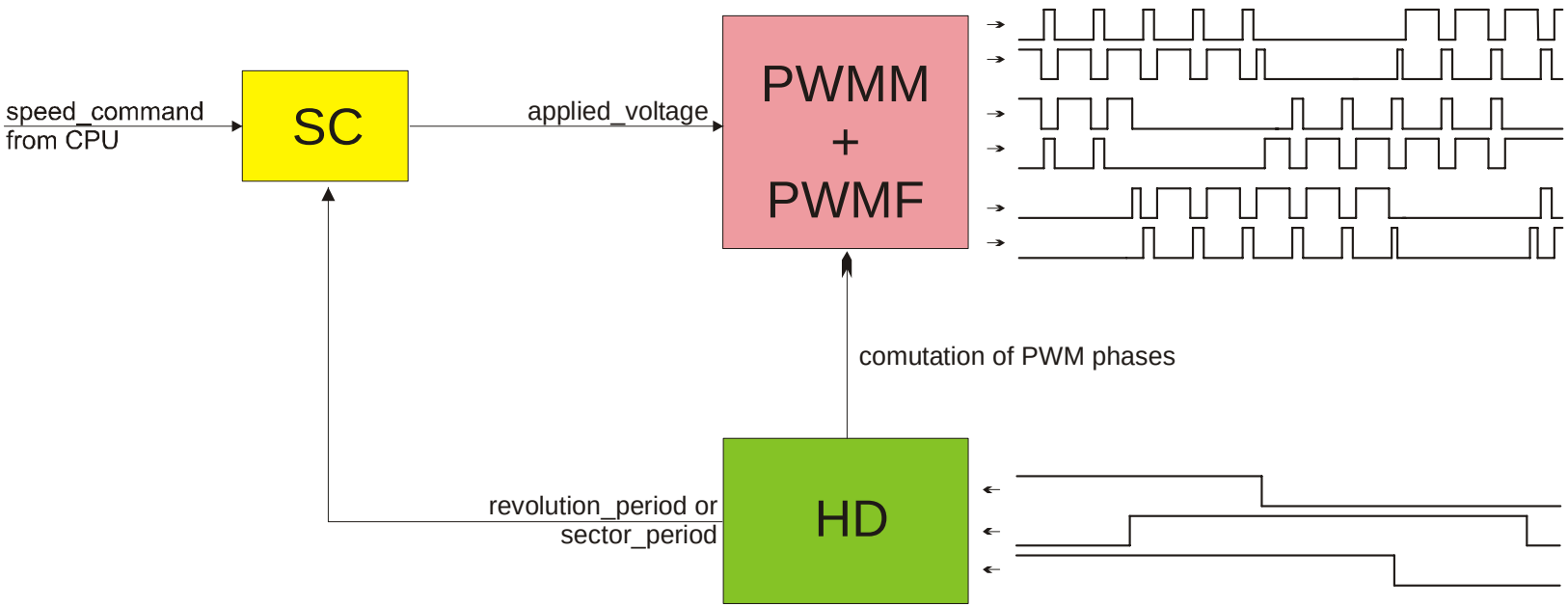
HD (Hall Decoder)

- Decoding of Hall sensor signals, driving commutation of PWM phases

SC (Speed Controller)

- Calculation of motor speed from HD sector period, speed PI controller

Block Diagram of eTPU Processing



eTPU Function Details

BLDC Motor with Speed Control Loop

Launched by Motorola

freescale semiconductor

PWMM + PWMC

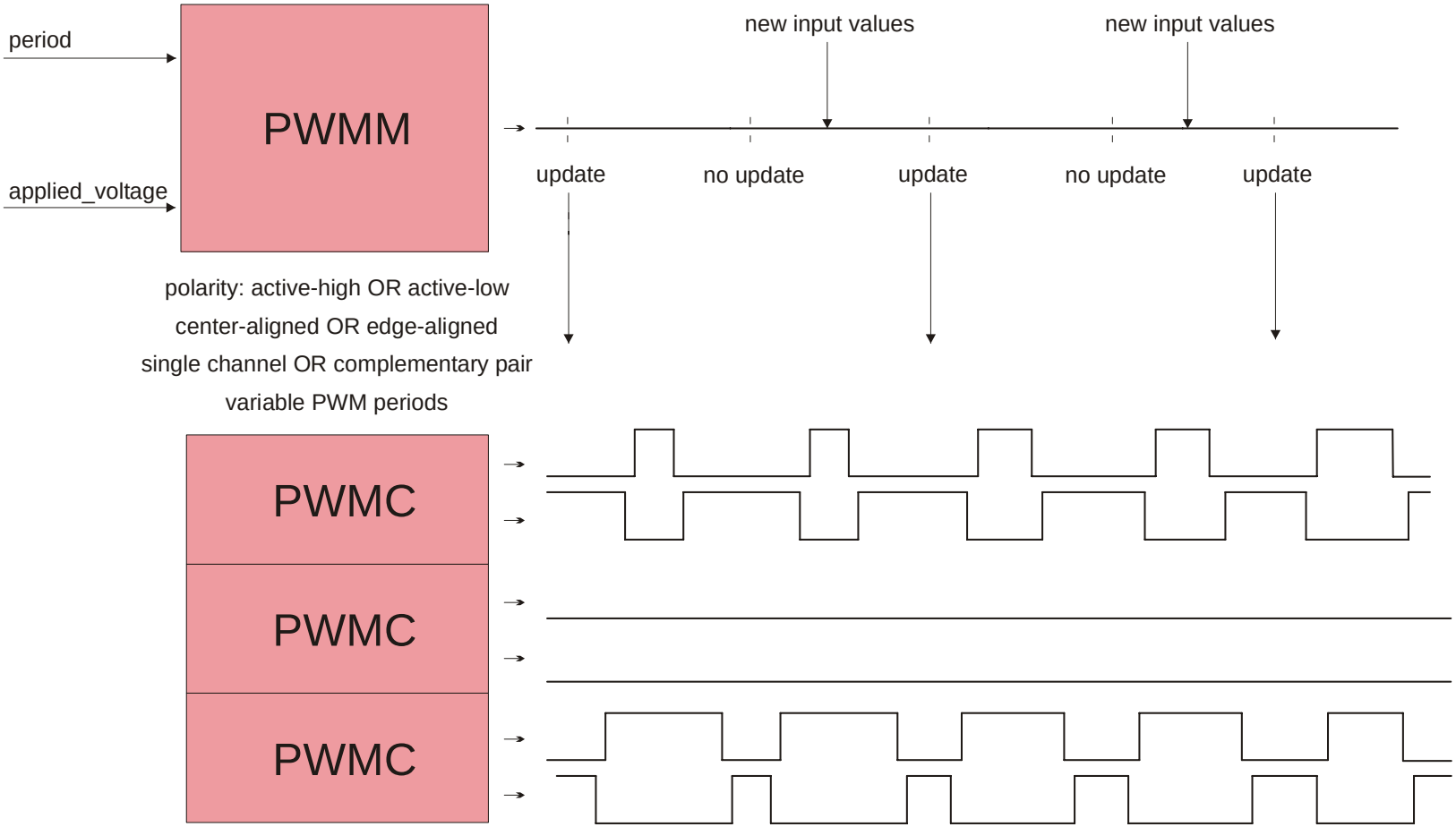
PWMM – PWM Master

- Does not generate any signal itself, but
- If an input parameter “applied_voltage” or “period” is updated, updates all edge times of the PWM signals generated by PWMC slaves
- Enables to master up to 4 phases

PWMC – PWM Slave with Commutations

- Generates one phase (top/bottom complementary pair with dead-time)
- Can be switched ON and OFF anytime = enables commutations
- Without updates from the PWMM keeps generating PWM signals of a constant duty-cycle and period

PWMM + PWMC



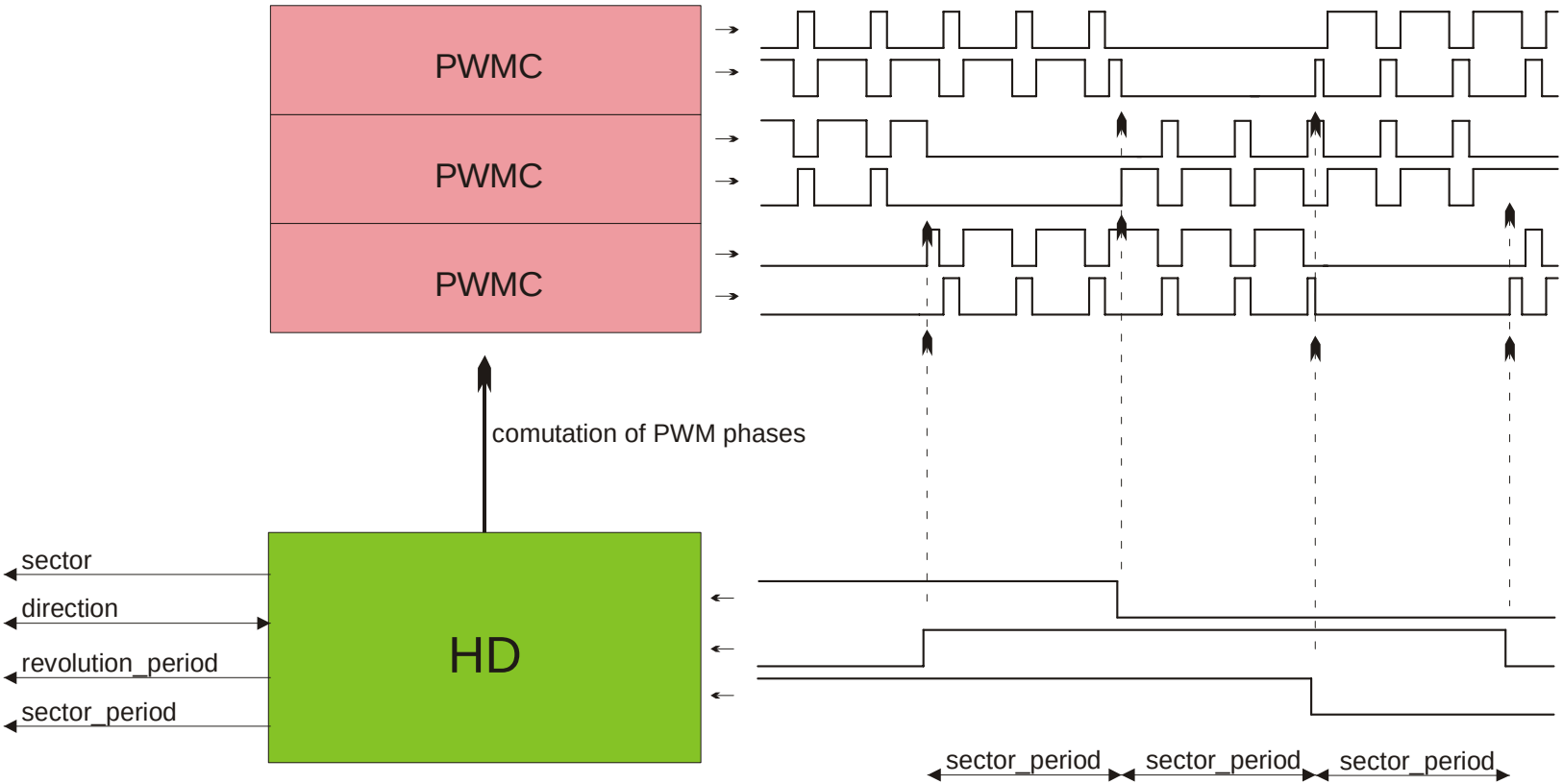
HD – Hall Decoder

Decodes 3 Hall sensor signals and produce

- **Sector = position in one of 6 sectors**
- **Direction**
- **Revolution period – time of the last revolution**
- **Sector period – time of (being in) the last sector**
- **Revolution counter**

On each transition turns one PWM phase OFF and one ON
= Commutation of PWM phases

HD – Hall Decoder



SC – Speed Controller

Calculates actual motor speed from the sector period measured by HD (division!).

speed_actual = scaling_factor * 1/sector_period

Passes the speed_command from CPU through a ramp, in order to slow down step changes.

continues …

SC – Speed Controller

Based on the error between required speed and actual speed calculates, using PI controller algorithm, applied motor voltage for PWMM.

$$error(k) = speed\_required(k) - speed\_actual(k)$$

$$applied\_voltage(k) = Kp * error(k) + \underbrace{Ki * error(k) + I(k-1)}_{I(k)}$$

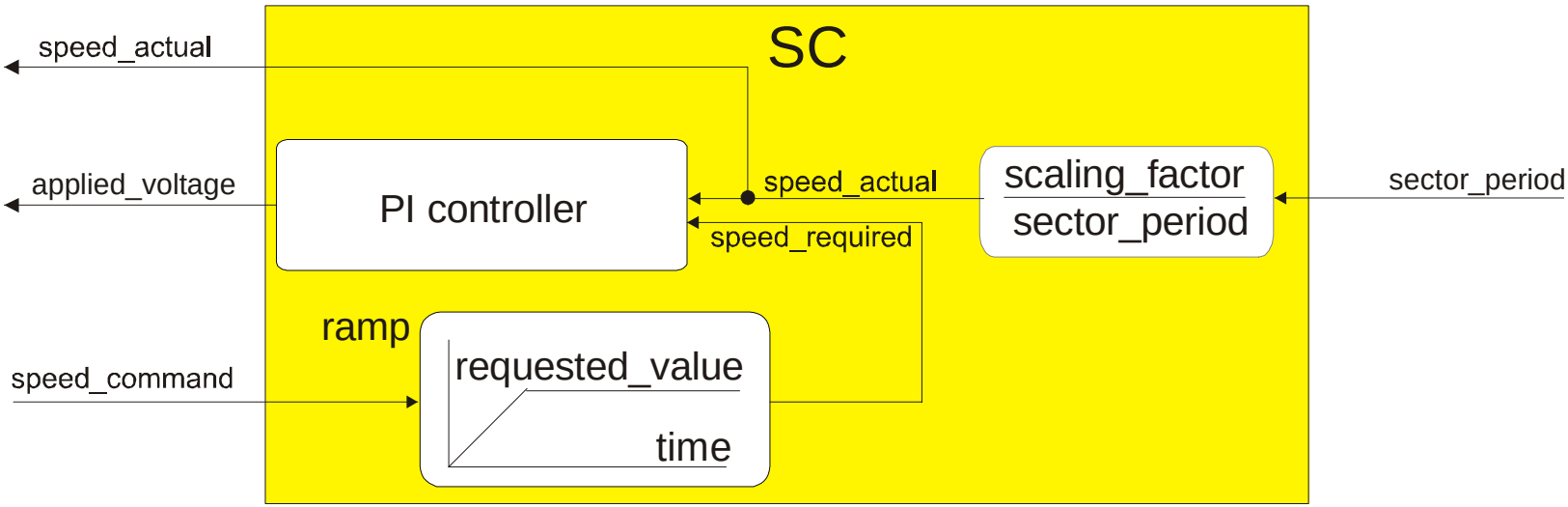
Kp … proportional gain

Ki … integrational gain

I(k) … integral portion in k^{th} step

- **Limitation of applied_voltage(k) and I(k), and scaling of Kp and Ki are implemented.**

SC – Speed Controller



Timing

PWM frequency … 16kHz

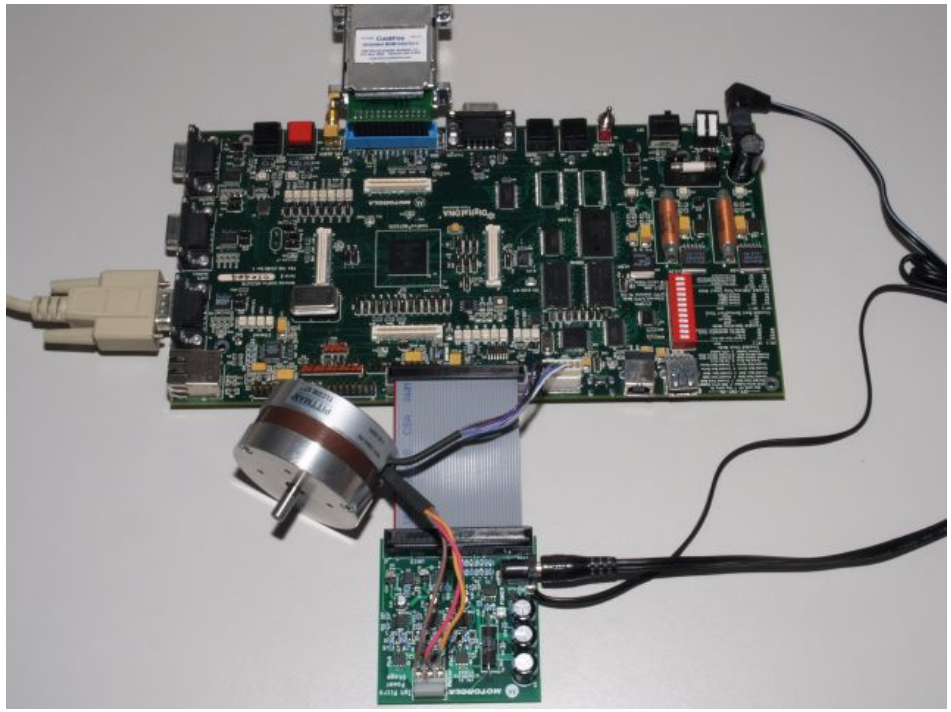
SC frequency …… 8kHz

PI controller is calculated and PWM duty-cycle is updated every second period.

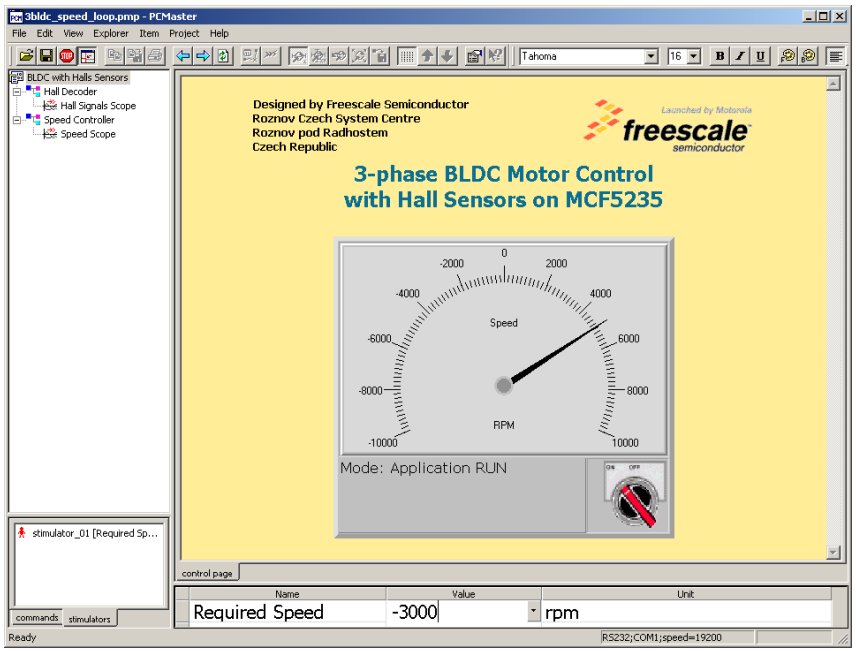
Demo

Demo ready with MCF523x EVB

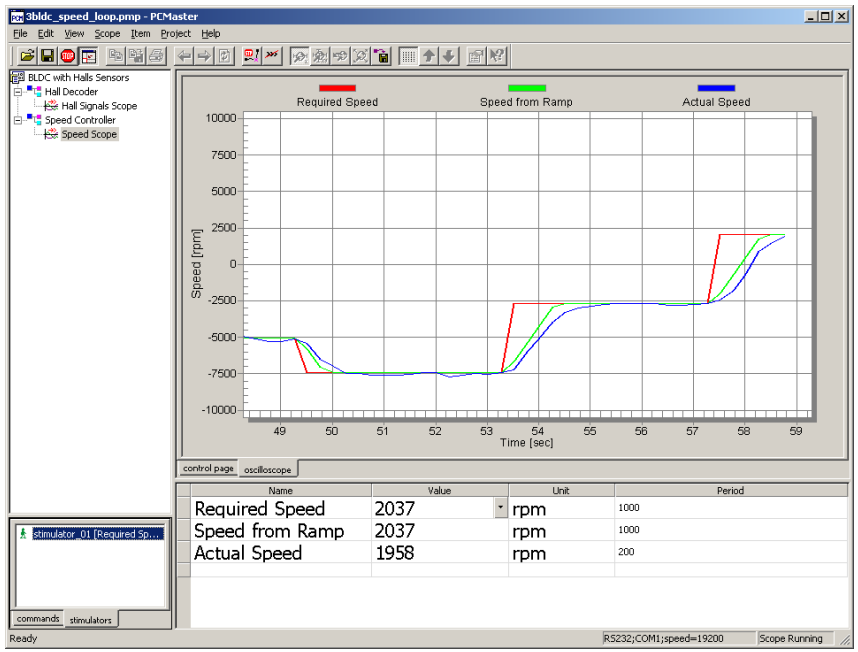
- **MCF523x EVB**
- **Pittman Motor**
- **Micro Power Stage**
- **Power Supply**
- **Serial cable to PC**



Application User Interface on PC using FreeMaster



Control Panel



Real-Time Speed Scope

Launched by Motorola
freescale™
semiconductor