PXMC Documentation

Isabelle Rieucros

Pavel Pisa

Michal Sojka

Konrad Skup

PXMC Documentation

by Isabelle Rieucros

by Pavel Pisa

by Michal Sojka

by Konrad Skup

Table of Contents

List of Figures

Chapter 1. Introduction

The PXMC is Portable, highly eXtendable Motion Control library and system core. The PXMC project started as the next generation, ground up rewrite of motion control systems used and evolved at PiKRON Ltd. company. The previous generation of the code has been used in MARS-2 motion control units. The variants of the new code has been used successfully on many targets for robotic, laboratory and medical instruments. The core of the motion system has been provided to CTU university for building motion controllers for Eorobot competition and other tasks and CTU contributes to the project for many years already.

The next platforms has been tested with version of the PXMC core code at PiKRON Ltd:

- FreeScale M68376 with DC and BLDC motors
- Renesas H8S2633/H8S2638 with stepper feedback and feedback-less motors
- Ti MSP430 - PXMC subset with adaptive current stepper motor control
- NXP LPC1768 Cortex-M3 based PiKRON's LMC1 board with BLDC motor

Next targets support has been developed and used at CTU university:

- Renesas H8S2638 with DC and BLDC motors

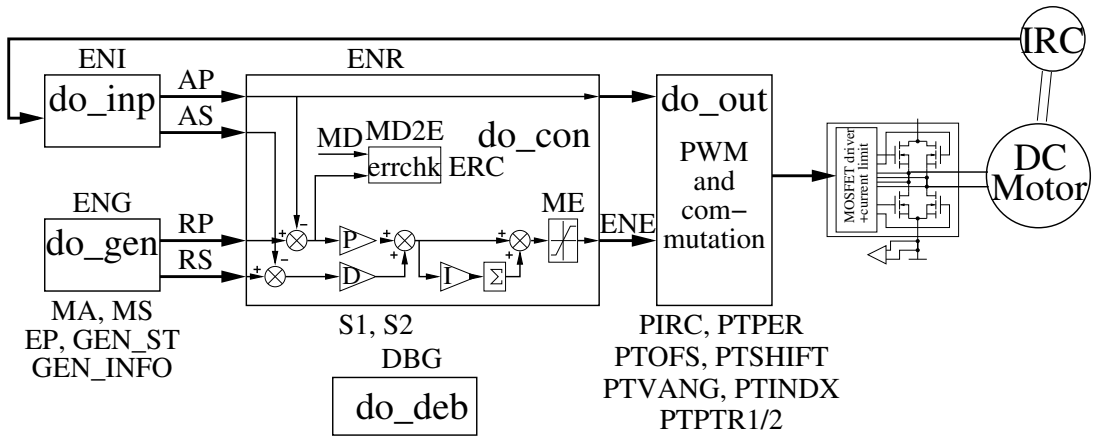
This document presents PXMC in the particular case of `pxmc_test.c` application which uses Command Processor to write motor control commands in a terminal.

The first part explains PXMC file organisation, structure and function roles, whereas the second part wants to be a simple tutorial to get onto Command Processor characteristics and PXMC related features.

Chapter 2. PXMC

This section presents a general survey about PXMC organisation and functioning. First it will describe the main files to use, then detail pxmc_state_t structure which is the heart of PXMC and it will finish PXMC global process.

Figure 2-1. PXMC Block Diagram



Block Diagram of PXMC Axis Controller

2.1. Required files

The most important files, those which contain PXMC code are `pxmc.c` and its header `pxmc.h`. They define all functions and the heart of this code: pxmc_state_t structure. They were modified to work on `pxmc_test.c` application. Now they contain DC motor control and Hitashi H8S2638 related functions.

`h8s2638h.h` is a library listing all registers, flags, masks, etc. with their respective hardware address.

2.2. pxmc_state_t structure

PXMC code is based on pxmc_state_t structure (its implementation is called motor controller state: `mcs`). It holds all state information for motion control of one axis equipped with DC motor, with or without encoder feedback. It also lists stepper or brush-less motor control characteristics but they were not implemented in the DC motor test application.

This structure comprises, among other things:

- axis flags to enable the controller, trajectory generator and for status information,
- pointers to functions unavoidable when dealing with motor control : trajectory generators (*pxms_do_gen), controllers (`*pxms_do_con`), output power (`*pxms_do_out`), input feedback (`*pxms_do_inp`). These are sample time handling routines.
- axis actual position and speed (`pxms_ap` and pxms_as),
- position and speed controller requests (`pxms_rp`, `pxms_rs`, `pxms_rpfg`, `pxms_rsfg`),
- axis parameters (`pxms_md`, `pxms_ms`, `pxms_ma`, `pxms_inp_info`, `pxms_out_info`),
- controller outputs (`pxms_ene`, `pxms_erc`),
- basic parameters for PID or other controller (`pxms_p`, `pxms_i`, `pxms_d`, `pxms_s1`, `pxms_s2`, `pxms_me`),
- informations for brushless motors,
- error codes and debugging helpers
- ...

struct pxmc_state

Name

`struct pxmc_state` — Motor Controller State Information

Synopsis

```
struct pxmc_state {
  pxmc_flags_t pxms_flg;
  pxmc_call_t * pxms_do_inp;
  pxmc_call_t * pxms_do_con;
  pxmc_call_t * pxms_do_out;
  pxmc_call_t * pxms_do_deb;
  pxmc_call_t * pxms_do_gen;
  pxmc_call_t * pxms_do_ap2hw;
  long pxms_ap;
  long pxms_as;
  long pxms_rp;
#ifdef PXMC_WITH_FINE_GRAINED
  pxmc_fg_t pxms_rpfg;
#endif
  long pxms_rs;
#ifdef PXMC_WITH_FINE_GRAINED
  pxmc_fg_t pxms_rsfg;
#endif
#ifndef PXMC_WITH_FIXED_SUBDIV
```

```
  short pxms_subdiv;
#endif
  long pxms_md;
  long pxms_ms;
  long pxms_ma;
  pxmc_info_t pxms_inp_info;
  pxmc_info_t pxms_out_info;
#ifdef PXMC_WITH_EXTENDED_STATE
  short pxms_ene_d;
#endif
  short pxms_ene;
  short pxms_erc;
  short pxms_p;
  short pxms_i;
  short pxms_d;
  short pxms_s1;
  short pxms_s2;
  short pxms_me;
#ifndef PXMC_WITH_FOI_FOD_LONG_TYPE
  long pxms_foi;
  long pxms_fod;
#else
  long pxms_foi;
  long pxms_fod;
#endif
  long pxms_tmp;
#ifdef PXMC_WITH_PHASE_TABLE
  short pxms_ptirc;
  short pxms_ptper;
  short pxms_ptofs;
  short pxms_ptshift;
  short pxms_ptmark;
  short pxms_ptvang;
  short pxms_ptindx;
  short * pxms_ptptr1;
  short * pxms_ptptr2;
  short * pxms_ptptr3;
  unsigned long pxms_ptscale_mult;
  unsigned short pxms_ptscale_shift;
  unsigned short pxms_ptamp;
  short pxms_pwm1cor;
  short pxms_pwm2cor;
  short pxms_pwm3cor;
#endif
#ifdef PXMC_WITH_EXTENDED_STATE
  long pxms_cur_d_act;
  long pxms_cur_q_act;
#endif
#ifdef PXMC_WITH_CURRENTFB
#endif
  short pxms_errno;
  short pxms_cfg;
  long pxms_ep;
```

```
  short pxms_gen_st;
  pxmc_info_t pxms_gen_info[8];
  short pxms_hal;
  short pxms_halerc;
};
```

Members

pxms_flg

Holds flag enabling encoder (PXMS_ENI_m), controller (PXMS_ENR_m) and trajectory generator (PXMS_ENG_m). Other flags represent status information - busy (PXMS_BSY_m), error (PXMS_ERR_m) and membership in coordinated group (PXMS_CMV_m). Flag PXMS_PHA_m is used for initial phase alignment. Flag PXMS_DBG_m selects debugging for axis.

pxms_do_inp

Pointer to function responsible for reading the encoder and updating *pxms_ap* and *pxms_as*.

pxms_do_con

Pointer to position controller which computes *pxms_ene* from axis state.

pxms_do_out

Transfers computed *pxms_ene* to PWM subsystem.

pxms_do_deb

Debugging support routine.

pxms_do_gen

Trajectory generator

pxms_do_ap2hw

Preset new actual position into HW (is not mandatory)

pxms_ap

Actual motor position multiplied by (1<<PXMC_SUBDIV)

pxms_as

Actual motor speed multiplied by (1<<PXMC_SUBDIV)

pxms_rp

Required motor position *(1<<PXMC_SUBDIV)

pxms_rpfg

Position extension for Fine Grained generator

pxms_rs

Required motor speed *(1<<`PXMC_SUBDIV`)

pxms_rsfg

Speed extension for Fine Grained generator

pxms_subdiv

Optional per axis subdivision of hardware (IRC) unit if `PXMC_SUBDIV` is not fixed value

pxms_md

Maximal accepted position difference

pxms_ms

Maximal speed in same units as *`pxms_as`* and *`pxms_rs`*

pxms_ma

Maximal acceleration

pxms_inp_info

Additional info for *`pxms_do_inp`* to select which irc to use

pxms_out_info

Additional info for *`pxms_do_out`* where *`pxms_ene`* should be sent

pxms_ene_d

Output PWM direct (D) component

pxms_ene

Computed output energy / value of PWM signal (quadrature Q component for DQ)

pxms_erc

Axis error counter

pxms_p

Controller proportional constant

pxms_i

Controller integration constant

pxms_d

Controller derivative constant

pxms_s1

Controller special 1 constant

pxms_s2

Controller special 2 constant

pxms_me

Maximal allowed output energy or PWM

pxms_foi

Temporary for I computation

pxms_fod

Temporary for D computation

pxms_foi

Temporary for I computation

pxms_fod

Temporary for D computation

pxms_tmp

Temporary for help and debugging

pxms_ptirc

IRC count per phase table

pxms_ptper

Number of periods per table

pxms_ptofs

Offset between table and IRC counter (0 <= irc - pxms_ptofs < pxms_ptirc)

pxms_ptshift

Shift of generated phase curves (high speed correction)

pxms_ptmark

Position of IRC index mark in the phase table

pxms_ptvang

Angle (in irc) between rotor and stator mag. fld.

pxms_ptindx

Index into commutation table

pxms_ptptr1

Pointer to commutation table for phase 1

pxms_ptptr2

Pointer to commutation table for phase 2

pxms_ptptr3

Pointer to commutation table for phase 3

pxms_ptscale_mult

Multiplication factor for ROM phase table scaling

pxms_ptscale_shift

Right shift for ROM phase table scaling

pxms_ptamp

Amplitude of phase table profile (max value)

pxms_pwm1cor

Correction for PWM1 generator

pxms_pwm2cor

Correction for PWM2 generator

pxms_pwm3cor

Correction for PWM3 generator

pxms_cur_d_act

Measured current D component

pxms_cur_q_act

Measured current Q component

pxms_errno

Error code

pxms_cfg

Config information for axis

pxms_ep

End position of movement

pxms_gen_st

Status for generators

pxms_gen_info[8]

Field available for trajectory generators computations.

pxms_hal

Last value read from HALL sensors

pxms_halerc

Error counter of hall errors

Description

This structure holds all state information for motion control of one axis equipped with stepper or brush-less motor with or without encoder feedback.

2.3. PXMC Process

Since pxmc_state_t structure is the heart of PXMC, the first thing to do is to initialize this structure implementation. The DC motor test application uses `mcs0` which is defined in `pxmc.c`. Then, the function `pxmc_axis_mode()` is called and sets the kind of motion control chosen, mode 4 for DC motor control with IRC feedback and PWM.

Next, `mcs->pxms_do_con` is set to point to the controller to use and the proper initialization function `pxmc_dcm_init_fbmode(mcs0)` (for DC motor application) is called. This function deals with register configuration (Timer Pulse Unit , PWM Unit) and sets `mcs->pxms_do_inp` and `mcs->pxms_do_out` pointer.

When these configurations are done, the timer in charge of the sampling period starts and activates an interrupt handler when a compare match occurs. This interrupt handler `pxmc_sfi_isr(void)` calls the functions pointed by mcs pointers:

- `mcs->pxms_do_inp` function `pxmc_tpuirc_inp()` updates actual position and speed,
- `mcs->pxms_do_con` function `pxmc_pid_con()` generates pxms_ene value according to the controller calculations,
- `mcs->pxms_do_out` function `pxmc_pwm1f_out()` send PWM signal to the motor,
- `mcs->pxms_do_deb` function helps for debugging,
- `mcs->pxms_do_gen` function generates trajectories.

2.4. Core PXMC Functions

The next section lists most of PXMC functions. [extern API] characteristic means that the function is declared in the header `pxmc.h` so it is part of the external application programming interface.

pxmc_set_const_out

Name

`pxmc_set_const_out` — Sets direct constant output.[extern API]

Synopsis

```
void pxmc_set_const_out (pxmc_state_t * mcs, int val);
```

Arguments

`mcs`

Motion controller state information

`val`

New value of energy/PWM output limited by *`pxms_me`*.

pxmc_connect_controller_prep

Name

`pxmc_connect_controller_prep` — Preparation of connection of controler to axis.

Synopsis

```
int pxmc_connect_controller_prep (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_connect_controller

Name

pxmc_connect_controller — Smooth connection of controler to axis.[extern API]

Synopsis

```
int pxmc_connect_controller (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_set_gen_prep

Name

pxmc_set_gen_prep — Prepares axis for change of generator.[extern API]

Synopsis

```
int pxmc_set_gen_prep (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_set_gen_smth

Name

pxmc_set_gen_smth — Smooth change to new generator.[extern API]

Synopsis

```
int pxmc_set_gen_smth (pxmc_state_t * mcs, pxmc_call_t * do_gen_gi, int flg);
```

Arguments

mcs

Motion controller state information

do_gen_gi

Initial state of next generator

flg

Additional flags to set (PXMS_BSY_m)

Description

Checks non-zero speed and provide smooth stop, than change to requested generator.

pxmc_go

Name

pxmc_go — Starts movement to requested position.[extern API]

Synopsis

```
int pxmc_go (pxmc_state_t * mcs, long val, int res, int mode);
```

Arguments

mcs

Motion controller state information

val

Requested target position

res

Left for future use

mode

Motion mode flags.

Description

Starts trapezoidal speed profile motion to the position *val*.

pxmc_go_spdfg

Name

pxmc_go_spdfg — Starts movement to position with selected speed.[extern API]

Synopsis

```
int pxmc_go_spdfg (pxmc_state_t * mcs, long endpos, long speed, int mode);
```

Arguments

mcs

Motion controller state information

endpos

Requested target position

speed

Fine-grained maximal speed of movement

mode

Motion mode flags

Description

Starts trapezoidal speed profile motion to requested end position with set up fine-grained speed

pxmc_stop

Name

`pxmc_stop` — Stops motion of axis.[extern API]

Synopsis

```
int pxmc_stop (pxmc_state_t * mcs, int mode);
```

Arguments

mcs

Motion controller state information

mode

Value 1 means emergency immediate stop, else smooth stop is proceeded.

pxmc_spd

Name

`pxmc_spd` — Starts constant speed motion.[extern API]

Synopsis

```
int pxmc_spd (pxmc_state_t * mcs, long val, int timeout);
```

Arguments

mcs

Motion controller state information

val

Requested speed

timeout

Non zero value results in smooth stop after *timeout* sampling periods

pxmc_spdfg

Name

pxmc_spdfg — Starts constant fine-grained speed motion.[extern API]

Synopsis

```
int pxmc_spdfg (pxmc_state_t * mcs, long val, int timeout);
```

Arguments

mcs

Motion controller state information

val

Requested fine-grained speed

timeout

Non zero value results in smooth stop after *timeout* sampling periods

pxmc_axis_set_pos

Name

pxmc_axis_set_pos — Set axis actual position

Synopsis

```
int pxmc_axis_set_pos (pxmc_state_t * mcs, long pos);
```

Arguments

mcs

Motion controller state information

pos

New forced position for IRC counter in the standard subdiv. format

pxmc_axis_release

Name

pxmc_axis_release — Release control of given axis.[extern API]

Synopsis

```
void pxmc_axis_release (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_pid_con

Name

pxmc_pid_con — Position PID Controller

Synopsis

```
int pxmc_pid_con (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_add_cspdfg

Name

pxmc_add_cspdfg — Adds fine-grained speed to requested position

Synopsis

```
void pxmc_add_cspdfg (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

Description

Adds fine-grained speed (*pxms_rs*,*pxms_rsfg*) to requested position (*pxms_rp*,*pxms_rpfg*). Written is asm.

pxmc_add_vspd

Name

pxmc_add_vspd — Adds variable speed to requested position

Synopsis

```
void pxmc_add_vspd (pxmc_state_t * mcs, long spd);
```

Arguments

mcs

Motion controller state information

spd

New requested speed

Description

Sets requested speed (*pxms_rs*) and adds it to requested position (*pxms_rp*).

pxmc_set_spd

Name

pxmc_set_spd — Sets new immediate value of speed

Synopsis

```
void pxmc_set_spd (pxmc_state_t * mcs, long spd);
```

Arguments

mcs

Motion controller state information

spd

New requested speed

Description

Sets requested speed (*pxms_rs*) and clears (*pxms_rsfg*)

pxmc_set_spdfg

Name

pxmc_set_spdfg — Sets new immediate value of fine-grained speed

Synopsis

```
void pxmc_set_spdfg (pxmc_state_t * mcs, long spd);
```

Arguments

mcs

Motion controller state information

spd

New requested fine-grained speed

Description

Sets requested speed (*pxms_rs*) and clears (*pxms_rsfg*)

pxmc_spdfg_gnr

Name

pxmc_spdfg_gnr — Constant speed fine-grained generator

Synopsis

```
int pxmc_spdfg_gnr (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_spd_gacc

Name

pxmc_spd_gacc — Smooth transition to requested speed

Synopsis

```
int pxmc_spd_gacc (long * spd, long rspd, long acc);
```

Arguments

spd

Pointer to controlled speed

rspd

Value of requested final speed

acc

Allowed acceleration

Description

This function modifies *spd* by *acc* increments until required value *rspd* is reached. It returns 0 when speed is reached, else returns 1.

pxmc_nop_gd

Name

pxmc_nop_gd — No-operation generator state

Synopsis

```
int pxmc_nop_gd (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_cont_gi

Name

pxmc_cont_gi — Initializes continuation generator

Synopsis

```
int pxmc_cont_gi (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

Description

This type of generator is temporarily used when new generator parameters are computed.

pxmc_trp_gi

Name

`pxmc_trp_gi` — Initializes trapezoid generator

Synopsis

```
int pxmc_trp_gi (pxmc_state_t * mcs);
```

Arguments

`mcs`

Motion controller state information

Description

This complex generator realizes motion to requested end position *`pxms_ep`* with trapezoid speed profile constrained by *`pxms_acc`* and *`pxms_ms`* parameters.

pxmc_trp_spdfg_gi

Name

`pxmc_trp_spdfg_gi` — Initializes fine-grained trapezoid generator

Synopsis

```
int pxmc_trp_spdfg_gi (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

Description

This complex generator realizes motion to requested end position *pxms_ep* with trapezoid speed profile constrained by *pxms_acc* and *pxms_gen_tspfg* parameters.

pxmc_spd_gi

Name

pxmc_spd_gi — Initializes constant speed generator

Synopsis

```
int pxmc_spd_gi (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

Description

Requested speed is smoothly changed to *pxms_gen_spd_sp* with acceleration defined by *pxms_gen_spd_ac*. New generator *pxms_gen_spd_next* can be selected after required speed is reached.

pxmc_spdnext_gi

Name

pxmc_spdnext_gi — Initializes transition to zero and then generator change

Synopsis

```
int pxmc_spdnext_gi (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_stop_gi

Name

pxmc_stop_gi — Initializes transition to zero speed and then stop

Synopsis

```
int pxmc_stop_gi (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_hh

Name

pxmc_hh — Starts reference search - Hard Home

Synopsis

```
int pxmc_hh (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

2.5. BDLC/Synchronous Drives Related Functions

struct pxmc_ptprofile

Name

struct pxmc_ptprofile — precomputed Phase Table Profile

Synopsis

```
struct pxmc_ptprofile {
  const char * ptname;
  unsigned ptid;
  short ptirc;
  short ptper;
  unsigned short ptamp;
  unsigned short ptphnum;
  const short * ptptr1;
  const short * ptptr2;
  const short * ptptr3;
};
```

Members

ptname

Textual identification of the phase profile

ptid

Numeric identification

ptirc

IRC count per phase table

ptper

Number of periods per table

ptamp

Amplitude of phase table profile (max value)

ptphnum

Number of phases in the table

ptptr1

Pointer to commutation table for phase 1

ptptr2

Pointer to commutation table for phase 2

ptptr3

Pointer to commutation table for phase 3

pxmc_init_ptable

Name

`pxmc_init_ptable` — Initializes phase tables

Synopsis

```
int pxmc_init_ptable (pxmc_state_t * mcs, int profile);
```

Arguments

mcs

Motion controller state information

profile

Phase profile selection

Description

This function initializes phase tables *pxms_ptptr1* and *pxms_ptptr2*. Profile is computed by one of profile functions - `pxmc_init_ptable_sin`, `pxmc_init_ptable_triang`, `pxmc_init_ptable_trapez`

pxmc_ptvang_deg2irc

Name

`pxmc_ptvang_deg2irc` — Converts phase shift from deg to IRC

Synopsis

```
short pxmc_ptvang_deg2irc (pxmc_state_t * mcs, int deg);
```

Arguments

mcs

Motion controller state information

deg

Phase shift angle in degree

pxmc_ptable_set_profile

Name

pxmc_ptable_set_profile — Initializes phase tables

Synopsis

```
int pxmc_ptable_set_profile (pxmc_state_t * mcs, const pxmc_ptprofile_t *
ptprofile, unsigned int ptirc, unsigned int ptper);
```

Arguments

mcs

Motion controller state information

ptprofile

Pointer to precompiled phase table descriptor

ptirc

IRC count per phase table

ptper

Number of periods per table

Description

The function setups PXMC state fields related to the phase table profiles according to provided precomputed provile data *ptprofile*. It sets *pxms_ptptr1*, *pxms_ptptr2* and *pxms_ptptr3* and computes phase table scaling quotient *pxms_ptscale_mult* and shift/divisor *pxms_ptscale_shift* according to provided parameters and table length.

pxmc_pthalalign

Name

pxmc_pthalalign — Starts HAL alignment measurement cycle

Synopsis

```
int pxmc_pthalalign (pxmc_state_t * mcs, long r2acq, long spd, pxmc_call_t *
fin_fnc);
```

Arguments

mcs

Motion controller state information

r2acq

Range to acquire HAL alignment information

spd

Speed to run acquire at

fin_fnc

Function to do final computation

2.6. Command Processor Support and Executive Functions

cmd_opchar_getreg

Name

cmd_opchar_getreg — selects the right axis

Synopsis

```
pxmc_state_t * cmd_opchar_getreg (cmd_io_t * cmd_io, const struct cmd_des *
des, char * param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

pxmc is designed for multi axis motion control, so each axis must be identificated. This done by a capital letter. The first axis must be A, the 2nd B, etc.

cmd_do_reg_go

Name

`cmd_do_reg_go` — checks the command format validity and calls pxmc_go.

Synopsis

```
int cmd_do_reg_go (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if pxmc_go returns -1, cmd_do_reg_go returns -1.

cmd_do_pwm

Name

cmd_do_pwm — checks the command format validity and calls pxmc_set_const_out.

Synopsis

```
int cmd_do_pwm (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

cmd_do_reg_hh

Name

cmd_do_reg_hh — checks the command format validity and calls pxmc_hh (home hardware).

Synopsis

```
int cmd_do_reg_hh (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if pxmc_hh returns -1, cmd_do_reg_hh returns -1.

cmd_do_reg_spd

Name

cmd_do_reg_spd — checks the command format validity and calls pxmc_spd.

Synopsis

```
int cmd_do_reg_spd (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if pxmc_spd returns -1, cmd_do_reg_spd returns -1.

cmd_do_reg_spdfg

Name

`cmd_do_reg_spdfg` — checks the command format validity and calls pxmc_spdfg.

Synopsis

```
int cmd_do_reg_spdfg (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if pxmc_spdfg returns -1, cmd_do_reg_spdfg returns -1.

cmd_do_reg_spdfgt

Name

cmd_do_reg_spdfgt — checks the command format validity and calls pxmc_spdfg.

Synopsis

```
int cmd_do_reg_spdfgt (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if pxmc_spdfgt returns -1, cmd_do_reg_spdfg returns -1.

cmd_do_stop

Name

cmd_do_stop — checks the command format validity and calls pxmc_stop.

Synopsis

```
int cmd_do_stop (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

cmd_do_release

Name

cmd_do_release — checks the command format validity and calls pxmc_axis_release(mcs).

Synopsis

```
int cmd_do_release (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

cmd_do_clrerr

Name

cmd_do_clrerr — checks the command format validity, clears the error flag.

Synopsis

```
int cmd_do_clrerr (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

it also stop the rotation calling pxmc_axis_release(mcs)

cmd_do_zero

Name

`cmd_do_zero` — checks the command format validity, sets axis position to 0.

Synopsis

```
int cmd_do_zero (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

it also stop the rotation calling pxmc_axis_release(mcs)

cmd_do_reg_rw_pos

Name

cmd_do_reg_rw_pos — read or write function, param is converted in 'long' and shifted

Synopsis

```
int cmd_do_reg_rw_pos (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if the command typed is a write function, records the value, if it is a read function returns the value asked.

cmd_do_reg_short_val

Name

cmd_do_reg_short_val — read or write function, param is converted in 'integer'

Synopsis

```
int cmd_do_reg_short_val (cmd_io_t * cmd_io, const struct cmd_des * des, char
* param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if the command typed is a write function, records the value, if it is a read function returns the value asked.

cmd_do_reg_long_val

Name

cmd_do_reg_long_val — read or write function, param is converted in 'long'

Synopsis

```
int cmd_do_reg_long_val (cmd_io_t * cmd_io, const struct cmd_des * des, char
* param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if the command typed is a write function, records the value, if it is a read function returns the value asked.

cmd_do_regsfrq

Name

`cmd_do_regsfrq` — sets or returns smapling frequency

Synopsis

```
int cmd_do_regsfrq (cmd_io_t * cmd_io, const struct cmd_des * des, char *
param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

cmd_do_regptmod_short_val

Name

`cmd_do_regptmod_short_val` — read or write axis 'short int' parameter and re-initialize axis mode

Synopsis

```
int cmd_do_regptmod_short_val (cmd_io_t * cmd_io, const struct cmd_des * des,
char * param[]);
```

Arguments

cmd_io

-- undescribed --

des

-- undescribed --

param[]

-- undescribed --

Description

if the command typed is a write function, records the value, if it is a read function returns the value asked. in addition to regular axis parameter write, recompute of the phase table and control mode state is invoked and commutator state is reinitialized.

2.7. PXMC Debugging Functions

pxmc_dbg_histfree

Name

pxmc_dbg_histfree — Frees motion history buffer

Synopsis

```
int pxmc_dbg_histfree (pxmc_dbg_hist_t * hist);
```

Arguments

hist

Motion history buffer

pxmc_dbg_histalloc

Name

pxmc_dbg_histalloc — Allocates new motion history buffer

Synopsis

```
pxmc_dbg_hist_t * pxmc_dbg_histalloc (int count);
```

Arguments

count

Number of allocated slots in motion history buffer

pxmc_dbg_ene_as

Name

pxmc_dbg_ene_as — Stores actual speed and output energy

Synopsis

```
int pxmc_dbg_ene_as (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_dbg_gnr

Name

pxmc_dbg_gnr — Generator of speed profile stored in history buffer

Synopsis

```
int pxmc_dbg_gnr (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_dbgset

Name

pxmc_dbgset — Selects debugging options for axis

Synopsis

```
int pxmc_dbgset (pxmc_state_t * mcs, pxmc_call_t * do_deb, int dbgflg);
```

Arguments

mcs

Motion controller state information

do_deb

Debugging callback function

dbgflg

0 .. disables debugging, 1 .. enables debugging for axis

2.8. Board Support/Platform Provided Functions

pxmc_tpuirc_inp

Name

pxmc_tpuirc_inp — IRC encoder connected to H8S 2638 TPU

Synopsis

```
int pxmc_tpuirc_inp (struct pxmc_state * mcs);
```

Arguments

mcs

Motion controller state information

Description

This version of input routine updates *pxms_ap* from TPU counter. Acquired value is then used for phase commutation..

pxmc_tpuirc_nofb_inp

Name

pxmc_tpuirc_nofb_inp — TPU IRC input without feedback commutation control

Synopsis

```
int pxmc_tpuirc_nofb_inp (struct pxmc_state * mcs);
```

Arguments

mcs

Motion controller state information

Description

This version of input routine updates *pxms_ap* from TPU counter, but commutation is controlled by *pxms_rp*.

pxmc_tpupwm2f_out

Name

pxmc_tpupwm2f_out — Two phase stepper motor phase PWM output

Synopsis

```
int pxmc_tpupwm2f_out (struct pxmc_state * mcs);
```

Arguments

mcs

Motion controller state information

Description

calls pxmc_tpupwm2f_tpuwr which, ifndef USE_BUF_PWM, uses (TPU_TGR0A, TPU_TGR0B) timers for PWM registers else (TPU_TGR0C, TPU_TGR0D)

pxmc_nofb_out

Name

pxmc_nofb_out — Phase output for open loop direct stepper motor control

Synopsis

```
int pxmc_nofb_out (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_nofb_inp

Name

pxmc_nofb_inp — Dummy input for direct stepper motor control

Synopsis

```
int pxmc_nofb_inp (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_nofb_con

Name

pxmc_nofb_con — Empty controller for direct stepper motor control

Synopsis

```
int pxmc_nofb_con (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_sfi_isr

Name

pxmc_sfi_isr — Stepper motor interrupt service routine

Synopsis

```
void pxmc_sfi_isr ( void);
```

Arguments

void

no arguments

Description

(I do not understand the following comment, maybe these are features that will be written later??)

This is basic routine calling motion controller functions.

pxmc_pwm1f_out

Name

pxmc_pwm1f_out — two phase PWM output for DC motor

Synopsis

```
int pxmc_pwm1f_out (struct pxmc_state * mcs);
```

Arguments

mcs

Motion controller state information

Description

This function sets directly duty cycle register in PWM_PWBFR1n n=(A,C,E,G) buffer register.

pxmc_axis_rdmode

Name

pxmc_axis_rdmode — Reads actual axis mode.[extern API]

Synopsis

```
int pxmc_axis_rdmode (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_get_sfi_hz

Name

pxmc_get_sfi_hz — Reads sampling frequency of axis

Synopsis

```
long pxmc_get_sfi_hz (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

pxmc_sfi_sel

Name

pxmc_sfi_sel — Setting sampling frequency of PXMC subsystem (TPU_TGR4A)

Synopsis

```
long pxmc_sfi_sel (long sfi_hz);
```

Arguments

sfi_hz

Requested sampling frequency in Hz

Description

Function returns newly selected sampling frequency after rounding or -1 if there is problem to set requested frequency.

pxmc_axis_mode

Name

pxmc_axis_mode — Sets axis mode.[extern API]

Synopsis

```
int pxmc_axis_mode (pxmc_state_t * mcs, int mode);
```

Arguments

mcs

Motion controller state information

mode

0 .. previous mode, 1 .. stepper motor mode, 2 .. stepper motor with IRC feedback and PWM , 3 .. stepper motor with PWM control 4 .. DC motor with IRC feedback and PWM

pxmc_dcm_init_fbmode

Name

pxmc_dcm_init_fbmode — Initializes feedback mode control

Synopsis

```
void pxmc_dcm_init_fbmode (pxmc_state_t * mcs);
```

Arguments

mcs

Motion controller state information

Description

Function initializes TPU for IRC input and one phase PWM output, generated by PWM unit, not TPU.

-> IRC inputs are TCLKC and TCLKD, Timer 2 is in phase counting mode 1. Timer 4 is used to count sampling period (set in TPU_TGR4A) and activate pxmc_sfi_isr interrupt handler. -> PWM outputs are A and B, the frequency of the signal is set in PWM_PWCYR1.

pxmc_stm_init

Name

pxmc_stm_init — Initializes stepper or brush-less motor subsystem

Synopsis

```
int pxmc_stm_init ( void);
```

Arguments

void

no arguments

Description

Configuration of IO registers, TPU for phase counting mode. Returns 0 if the axis mode is set correctly.

Timers 2 or 5 are in phase counting mode 1. Choose which one defining PXMC_IRC_TPUN. Timer 4 is used to count sampling period (set in TPU_TGR4A) and activate pxmc_sfi_isr interrupt handler.

Chapter 3. Terminal Commands

The test of PXMC targeted to DC motor control was done calling PXMC functions directly from a terminal, typing text line commands of Command Processor , but this application is not dedicated to PXMC. The Command Processor permits to associate functions to the commands that will be typed in a terminal.

This theme is divided in two parts; the first one is related to general Command Processor functionalities and the second part deals with PXMC related features.

3.1. Serial Communication Setting

The text line Command Processor is designed for instruments control and setup over RS-232 line. This is why serial communication must be well configured, defining `SCI_RS232_PORT_NUM` in `sci_rs232.c` file, and initialised, calling `sci_rs232_setmode()`. On the Hitashi H8S2638 board, you can use one serial port for loading the application and another one for terminal communication.

3.2. Command Processor

This section is getting first onto general Command Processor code contained in `cmd_proc.c` and `cmd_proc.h` files (they define multilevel tables of commands and send reply to respective I/O stream outputs); then it is describing PXMC dedicated commands, written in `cmd_pxmc.c` file.

3.2.1. Command Format

The generic command format is the following:

<COMMAND>[<`PARAM1`>][<`OPCHAR`>][<`VALUE`>]

where

COMMAND

is the name of the command

`PARAM1`

is the first parameter, facultative

`OPCHAR`

is for "operation character", facultative but if used must be ’ : ’, ’ ? ’

`VALUE`

is the value you want to set, facultative.

Looking to `cmd_opchar_check()` function in `cmd_pxmc.c` file, three format utilisations can be deduced:

- **<COMMAND>**[<`PARAM1`>]`?`

  used in case of reading commands

- **<COMMAND>**[<`PARAM1`>]`:`[<`VALUE`>]

  used in case of writing commands

- **<COMMAND>**

  used in case of "displaying" commands. At the actual state, only the **HELP** command has this format.

3.2.2. General Error Codes

This error listing brings together some error codes due to Command Processor bad use.

- ERROR 2

  "CMDERR_BADCMD" is the first error easy to make: a typing error (!). Even a back arrow to correct a mistake will handle an error. Be careful when typing a command.

- ERROR 10

  "CMDERR_OPCHAR" means that the operation character is wrong, only ’:’ or ’?’ are accepted.

- ERROR 11

  "CMDERR_WRPERM" means that the function used in the command line does not have the permission to set a value.

These are the most common errors, if you encounter other error codes look in `cmd_proc.h`.

3.3. PXMC Related Commands

Programmers should use PXMC external application programming interface (API), i.e. the functions listed in `pxmc.h` header file. `pxmc_test.c` application goes too deep in software level because commands of Command Processor call some internal functions, for example about the controller parameters, that should be invisible. But this is to be a test application of PXMC so it must test some internal setting functions.

Note: PXMC is designed for multi axis motion control, so each axis must be identificated. This done by a capital letter: the first axis must be A, the 2nd B, etc. This value is selected by <PARAM1> in command format.

3.3.1. Commands without motor feedback control

To display the available commands, type **HELP**: the names of the different commands and their function appear.

```
>HELP
```

Now, let's start with a first command, to make the motor rotate. Typing **PWM?** command sets directly the duty cycle of the PWM signal, there is no feedback loop. This is only a writing command.

```
>PWMA:300
```

Note: PWM value must be included in [256, 8000].

The motor is turning. To stop it, simply type **RELEASE?** command. This is only a writing command. Here, no value is needed and it can be written:

```
>RELEASEA:
```

3.3.2. Commands to use a controller

- Speed control

  Using a controller, it is possible to ask for a speed with **SPD?** or **SPDFG?** (Fine grained speed) commands.These are only a writing commands

  >**SPDA:15**
  >SPDFGA:500000

  > **Note: SPD** value must be included in [1, 1500].
  >
  > **SPDFG** value must be included in [100 000, 98 500 000].
  >
  > **SPDFG**100 000 = **SPD**1.

  To stop it , simply type **STOP?** command. This is only a writing command. Here, no value is needed and it can be written:

  >**STOPA:**

- Position control

  It is also possible to control the position. The reference position is taken just before the motor begins to rotate. For the DC motor used in the test application, one revolution counts 400 positions (from the 100 IRC sensor notches and the phase counting mode at every edge).

  The actual position is returned by **AP?** reading command.

  >**APA?**
  >APA=6614

  Ask to go to a position with **G?** writing command.

  >**GA:200**

  ZERO? command sets a new reference position (writing command). Here, no value is needed and it can be written:

  >**ZEROA:**

- Controller and IRC parameters

  Trough Command Processor terminal, other parameters can be set.

  PID constants or special parameters for other controllers can be set by Read/Write **REGP?**, **REGI?**, **REGD?**, **REGS1?**, **REGS2?** commands.

  IRC parameters can be changed with **REGPT...?** family commands. (not tested)

  Same thing for constraints and maximal output values: **REAMD?** for maximal position error, **REGMS?** for maximal speed, **REGACC?** for maximal acceleration, **REGME?** for maximal PWM output energy.

3.3.3. General debugging commands

There are two other useful reading commands in this test application:

- **AXERR?** returns the last axis error code, see PXMC ERRORS section for error code descriptions.

  >**AXERRA?**

- **PURGE?** clears error flags so that you can go on controlling the motor

  >**PURGEA**?

You can also use some debugging commands, **REGDBG..?** family ,not tested at this moment.

3.3.4. PXMC errors

AXERR? returned value are PXMC related errors, see AXERR? command definition.They are defined in `pxmc.h` where they are coded in hexadecimal.

- >AXERRA=262
  equals 0x106, named PXMS_E_MAXPD. Means that the difference of position is over the limit.
- >AXERRA=263

equals 0x107, named PXMS_E_OVERL. Means that an overload error has occurred.

Here are listed encountered errors, but an offset of commutation error, when using a stepper motor, is also defined.

Chapter 4. Conclusion - Perspectives

After having been written for stepper or brushless motor control for Hitashi H8S2633 processor, now PXMC files contain DC motor control and Hitashi H8S2638 related functions ans this first documentation was drawn up. It wants to be an helper in understanding PXMC functioning, so that this code can go on being enhanced and quickly used in future motor control applications.

The first improvements that could be performed are very concrete:

- to modify PXMC `pxmc_set_const_out()` or `pxmc_pwm1f_out()` so that the duty cycle value could be in %. (see **PWM?** command),
- to improve hardware initialisation functions in order to be possible to choose which timer/channel to use,
- to split the files following hardware dependant/independant functions, separating DC and stepper motor related functions, so that PXMC could be more understandable and modular.