# Moduly pro řízení procesů – Regulators – Dokumentace

Knihovna je určena….

## Použití knihovny

Předpokládané použití je následující:

Uživatel vloží do svého main.c hlavičku:

#include "ucp\_app.h"

Někde na začátku main() volá ucp\_app\_init();

V pravidelných intervalech (s periodou vzorkování) volá z main.c funkci:

ucp\_app\_on\_sample();

Do projektu přidá „šablony“ souborů ucp\_app.h a ucp\_app.c (poskutnuty v knihovně) – **vytvoří si vlastní kopie těchto souborů v projektu**. Tyto vlastní kopie souborů pak edituje:

Ucp\_app.h – nevyžaduje editaci.

V Ucp\_app.c:

Definuje proměnnou (proměnné) pro regulátor (regulátory). Např. :

UCP\_PSD\_REG psd1;

Definuje parametry regulátoru (případně může přímo zapsat v místě volání, ale vhodnější je definice předem).

#define Q0 (0.009)

#define Q1 (-0.008)

#define Q2 (0.0)

#define Q3 (0.0)

Definuje minimální a maximální hodnotu akčního zásahu:

#define MINU (0.0)

#define MAXU (100.0)

Ve funkci void ucp\_app\_init(void) volat inicializaci regulátoru:

ucp\_psd\_init(&psd1, Q0, Q1, Q2, Q3 );

Ve funkci void ucp\_app\_on\_sample(void) upravit obsah podle potřeby. Je třeba provést:

* Naměření nové hodnoty výstupu řízené soustavy
* Získání aktuální hodnoty požadované hodnoty (set point)
* Výpočet akčního zásahu
* Realizaci akčního zásahu

Příklad obsahu funkce void ucp\_app\_on\_sample(void):

float input, setpoint;

float u;

/\* Namereni vstupu \*/

input = ucphal\_read\_input(1); /\* kanal 1 \*/

/\* Nacteni aktualni pozadovane hodnoty \*/

setpoint = ucphal\_read\_setpoint(1); /\* pro kanal 1 \*/

/\* Vypocet akcnich zasahu pro dany kanal \*/

u = ucp\_psd\_step(&psd1, input, setpoint, MINU, MAXU);

/\* Provedeni akcnich zasahu > Zapis vypoctenych hodnot nekam, do

"registru", odkud je realizuje napr. nezavisly proces \*/

ucphal\_write\_output(1, u);

Pro výpočet aktuálního akčního zásahu se volá funkce ucp\_x\_step() daného regulátoru, např.

ucp\_psd\_step(&psd1, input, setpoint, MINU, MAXU);

Pro ostatní kroky je v knihovně podpora přes standardizované rozhraní – tzv. **HAL** = Hardware abstraction Layer. Uživatel by měl pouze definovat vlastní obsah funkcí podle své potřeby v souboru ucp\_hal.c.

Funkce HAL:

ucphal\_read\_input(1); - čtení výstupu soustavy pro daný regulovaný kanál.

ucphal\_read\_setpoint(1); - získání žádané hodnoty pro daný kanál.

ucphal\_write\_output(1, u); - provedení (nastavení) akčního zásahu, který je vypočten regulátorem.

Vložit do projektu soubory (šablony) ucp\_hal.h a ucp\_hal.c.

V souboru ucp\_hal.c doplnit obsah jednotlivých funkcí. Postupovat přitom podle komentářů v kódu.

V nastavení kompilátoru přidá cestu k include souborům knihovny UCP tj. [umisteni\_ucp]\support\s08 popř. [umisteni\_ucp]\support\kinetis.

V případě, že pro realizaci akčního zásahu je použito PWM, nabízí knihovna podporu pro jednoduché softwarové PWM. Uživatelská aplikace v tom případě musí zajistit volání funkce ucp\_pwm\_tick() a to s takovou periodou, aby 100 volání ucp\_pwm\_tick() odpovídalo periodě PWM. Např. pokud požadujeme periodu PWM 1 s, pak je nutno volat ucp\_pwm\_tick() každých 10 ms.

## Příklad

Příklad funkce main s pravidelným volání ucp\_app\_on\_sample() a také ucp\_pwm\_tick():

#include <hidef.h> /\* for EnableInterrupts macro \*/

#include "derivative.h" /\* include peripheral declarations \*/

#include "ucp\_app.h"

#include "ucp\_swpwm.h"

interrupt void timer\_int(void); // prototyp funkce pro obslouzeni preruseni

// definujeme globalni promennou jako ukazatel na funkci a to na dane misto pameti

// (na adresuvektoru preruseni) a do teto promenne nastavime adresu nasi funkce timer\_int

void (\*const obsluha)(void) @0xFFEE = timer\_int;

/\* Kolik volani obsluhy preruseni casovace odpovida 1 periode vzorkovani.

Zde perioda TOF je 20 ms, perioda vzorkovani je 2 s \*/

#define TICKS\_IN\_TV (100)

uint8\_t counter;

volatile uint8\_t status;

void main(void) {

// Nastavit casovac pro generovani preruseni kazdych 20 ms

// Pri kazdem preruseni zavolame ucp\_pwm\_tick(). (perioda PWM = 2 s)

// Z obsluhy preruseni take budeme volat ucp\_app\_on\_sample() (perioda vzorkovani).

TPM1SC = 0x4F; // source fbus (20 MHz), delicka 128

TPM1MOD = 3125; // modulo registr pro 20 ms TOF

counter = 0;

status = 0;

EnableInterrupts; /\* enable interrupts \*/

/\* Inicializace UCP aplikace (ta vola init regulatoru apod.) \*/

ucp\_app\_init();

for(;;) {

// Obsluha preruseni nam pomoci "status" signalizuje, ze uplynula doba

// pro volani on\_sample()

if ( status == 1 )

{

status = 0;

ucp\_app\_on\_sample();

}

\_\_RESET\_WATCHDOG(); /\* feeds the dog \*/

} /\* loop forever \*/

/\* please make sure that you never leave main \*/

}

/\* Obsluha preruseni od casovace 1 \*/

interrupt void timer\_int(void)

{

TPM1SC &= 0x7F; // nuluj priznak preruseni

counter++;

ucp\_pwm\_tick();

if ( counter == TICKS\_IN\_TV )

{

counter = 0;

/\* Poznamka: neni vhodne zde volat on\_sample() primo, protoze funkce muze trvat

dlouho (vypocty s float) a zpusobovala by vypadek casovani PWM; proto jen nastavime

priznak a volani se provede z hlavni smycky \*/

//ucp\_app\_on\_sample(); /\* NEVHODNE! \*/

status = 1;

}

}