//! [single chip calibration]

/\*\*

\*/

#include <stdio.h>

#include <stdint.h>

#include <stdlib.h>

#include "pssc\_cal\_lib.h"

#include "pssc\_cfg\_lib.h"

#include "pssc\_com\_lib.h"

#include "pssc\_logger.h"

#ifdef \_WIN32

#pragma comment(lib, "pssc\_cal\_lib.lib")

#pragma comment(lib, "pssc\_cfg\_lib.lib")

#pragma comment(lib, "pssc\_com\_lib.lib")

#pragma comment(lib, "pssc\_logger.lib")

#endif

#ifdef \_WIN32

static char \*port = "COM7"; // com port for windows

#else

static char \*port = "ttyUSB0"; // com port for linux

#endif

#define CAL\_POINTS 4

#define VER\_POINTS 1

// temperature values for calibration

static double temperature[] = {0, 0, 100.0, 100.0};

static double expected\_temperature[] = {-32000, -32000, 32000, 32000};

// pressure values for calibration

static double pressure[] = {0.0, 100.0, 0.0, 100.0};

static double expected\_pressure[] = {-16000, 16000, -16000, 16000};

// temperature values for verification

static double temperature\_ver[] = {100.0};

// pressure values for verification

static double pressure\_ver[] = {0.0};

static double expected\_pressure\_ver[] = {-16000};

int main(void) {

int32\_t error;

com\_t handle = NULL;

printf("\nStart PSSC calibration cycle\n");

printf("\nRead chip settings from file\n");

cfg\_t cfg\_file; // Configuration loaded from file

cfg\_t cfg\_chip; // Configuration get from chip

pssc\_cfg\_gbl\_init(&cfg\_file, PSSC\_703\_11);

pssc\_cfg\_gbl\_load\_from\_file(&cfg\_file, "configuration.hex");

printf("\nConnect to device\n");

char info[256];

// set logging level (to stdout)

// pssc\_logger\_set\_debug\_level(PSSC\_LOGGER\_INFO);

pssc\_logger\_set\_debug\_level(PSSC\_LOGGER\_SILENT); // default

if ((error = pssc\_com\_connect(&handle, port, AUTO, IF\_AUTO, info))) {

printf("Error: Could not connect to device (#%d)!\n", error);

pssc\_com\_get\_error\_message(error, info);

printf("\tError message: %s\n", info);

return 1;

}

else {

printf("\tConnection: %s\n", info);

}

uint32\_t number\_of\_sockets = 1;

uint32\_t numner\_of\_devices = 0;

pssc\_com\_get\_num\_sockets(handle, &number\_of\_sockets);

int32\_t \*addresses = malloc(sizeof(uint32\_t) \* number\_of\_sockets);

if ((error = pssc\_com\_scan\_devices(handle, &numner\_of\_devices, addresses))) {

printf("Error: Scan devices error (#%d)!\n", error);

pssc\_com\_get\_error\_message(error, info);

printf("\tError message: %s\n", info);

return 1;

} else {

printf("\tScan ok.\n");

}

if (addresses[0] == -1) {

printf("Error: Scan devices succeeded but no device found in socket.\n");

return 1;

} else {

// not necessary because last device is selected after scan devices and single boards only have one socket, but for completeness selection is made explicitly

pssc\_com\_select\_device(handle, 0);

}

printf("\nIdentify device\n");

if ((error = pssc\_com\_cfg\_read(handle, &cfg\_chip))) {

printf("Error: Could not read configuration from device (#%d)!\n", error);

return 1;

}

int32\_t version = pssc\_cfg\_gbl\_device\_type\_get(&cfg\_chip);

uint32\_t serial = pssc\_cfg\_gbl\_serial\_get(&cfg\_chip);

uint16\_t user\_serial = pssc\_cfg\_gbl\_user\_get(&cfg\_chip);

printf("\tChip: version=%d, ", version);

printf("serial number=%08x, ", serial);

printf("user defined serial number=%04x\n", user\_serial);

printf("Copy factory information from chip to configuration from file\n");

pssc\_cfg\_gbl\_device\_info\_transfer(&cfg\_file, &cfg\_chip);

printf("\nStop measuring\n");

if ((error = pssc\_com\_device\_measurement\_stop(handle))) {

printf("Error: Could not write stop command (#%d)!", error);

return 1;

}

printf("\nWrite initial settings from file to device\n");

if ((error = pssc\_com\_cfg\_write(handle, &cfg\_file))) {

printf("Error: Could not write configuration (#%d)!", error);

return 1;

}

printf("\nStart measuring\n");

if ((error = pssc\_com\_device\_measurement\_start(handle))) {

printf("Error: Could not write start command (#%d)!", error);

return 1;

}

printf("\nRun calibration\n");

double measured\_temperature\_t[CAL\_POINTS];

double measured\_temperature\_tc[CAL\_POINTS];

double measured\_pressure[CAL\_POINTS];

unsigned int steps;

for (steps = 0; steps < CAL\_POINTS; steps++) {

printf("\nStep %d: set up temperature %.1f dgC and pressure %.1f psi\n", steps + 1, temperature[steps], pressure[steps]);

printf("Press <enter> for calibration step\n"); getchar();

printf("Measure ... ");

int16\_t adc\_s, adc\_t, adc\_tc;

error = pssc\_com\_read\_adc(handle, &adc\_s, &adc\_t, &adc\_tc);

if (error != 0) {

measured\_temperature\_t[steps] = 0;

measured\_temperature\_tc[steps] = 0;

measured\_pressure[steps] = 0;

printf("failed!\n");

}

else {

measured\_temperature\_t[steps] = (double)((int16\_t)adc\_t);

measured\_temperature\_tc[steps] = (double)((int16\_t)adc\_tc);

measured\_pressure[steps] = (double)((int16\_t)adc\_s);

printf("done\n");

}

}

printf("Results:\n");

for (steps = 0; steps < CAL\_POINTS; steps++) {

printf("\tStep %d: T=%+.1f dgC, P=%+.1f psi, T\_meas=%g, TC\_meas=%g, S\_meas=%g\n", steps, temperature[steps], pressure[steps], measured\_temperature\_t[steps], measured\_temperature\_tc[steps], measured\_pressure[steps]);

}

printf("\nCalculate polynomial coefficients\n");

fit\_info t\_fit;

fit\_info s\_fit;

double t\_err\_at\_point[CAL\_POINTS];

double s\_err\_at\_point[CAL\_POINTS];

if ((error = pssc\_cal\_calibration(&cfg\_file,

T\_AUTO\_ORDER,

S\_AUTO\_ORDER,

CAL\_POINTS,

measured\_temperature\_t,

measured\_temperature\_tc,

measured\_pressure,

expected\_temperature,

expected\_pressure,

&t\_fit,

t\_err\_at\_point,

&s\_fit,

s\_err\_at\_point

))) {

printf("Pressure calibration failed (#%d)!\n", error);

}

printf("t\_fit.err\_max = %g\n", t\_fit.err\_max );

printf("t\_fit.err\_rms = %f\n", t\_fit.err\_rms );

printf("t\_fit.used\_adc\_t\_range = %g\n", t\_fit.used\_adc\_t\_range );

printf("t\_fit.used\_adc\_s\_range = %g\n", t\_fit.used\_adc\_s\_range );

printf("t\_fit.used\_adc\_ts\_area = %g\n", t\_fit.used\_adc\_ts\_area );

printf("t\_fit.sample\_density = %g\n", t\_fit.sample\_density );

int i;

for (i = 0; i < CAL\_POINTS; i++) {

printf("t\_err\_at\_point[%i] = %g\n", i, t\_err\_at\_point[i]);

}

printf("s\_fit.err\_max = %g\n", s\_fit.err\_max );

printf("s\_fit.err\_rms = %f\n", s\_fit.err\_rms );

printf("s\_fit.used\_adc\_t\_range = %g\n", s\_fit.used\_adc\_t\_range );

printf("s\_fit.used\_adc\_s\_range = %g\n", s\_fit.used\_adc\_s\_range );

printf("s\_fit.used\_adc\_ts\_area = %g\n", s\_fit.used\_adc\_ts\_area );

printf("s\_fit.sample\_density = %g\n", s\_fit.sample\_density );

for (i = 0; i < CAL\_POINTS; i++) {

printf("s\_err\_at\_point[%i] = %g\n", i, s\_err\_at\_point[i]);

}

printf("\nStop measuring\n");

if ((error = pssc\_com\_device\_measurement\_stop(handle))) {

printf("Error: Could not write stop command (#%d)!", error);

return 1;

}

printf("\nWrite settings with calibration\n");

if ((error = pssc\_com\_cfg\_write(handle, &cfg\_file))) {

printf("Error: Could not write configuration (#%d)!", error);

return 1;

}

printf("\nStart measuring\n");

if ((error = pssc\_com\_device\_measurement\_start(handle))) {

printf("Error: Could not write start command (#%d)!", error);

return 1;

}

double measured\_pressure\_ver[VER\_POINTS];

for (steps = 0; steps < VER\_POINTS; steps++) {

printf("\nStep %d: set up temperature %.1f dgC and pressure %.1f psi\n", steps + 1, temperature\_ver[steps], pressure\_ver[steps]);

printf("Press <enter> for verification step\n"); getchar();

printf("Measure ... ");

uint16\_t status;

int16\_t dsp\_s, dsp\_t;

error = pssc\_com\_read\_spt(handle, &status, &dsp\_s, &dsp\_t);

if (error != 0) {

measured\_pressure\_ver[steps] = 0;

printf("failed!\n");

} else {

measured\_pressure\_ver[steps] = (double)((int16\_t)dsp\_s);

printf("done\n");

}

}

printf("\nVerify\n");

cal\_verification veri;

veri.min = 0;

veri.max = 0;

veri.avg = 0;

veri.rms = 0;

pssc\_cal\_verification(VER\_POINTS, measured\_pressure\_ver, expected\_pressure\_ver, &veri);

printf("veri.min = %g\n", veri.min); /\*\*< minimal error \*/

printf("veri.max = %g\n", veri.max); /\*\*< maximal error \*/

printf("veri.avg = %g\n", veri.avg); /\*\*< average error \*/

printf("veri.rms = %g\n", veri.rms); /\*\*< root mean square error \*/

printf("\nStop measuring\n");

if ((error = pssc\_com\_device\_measurement\_stop(handle))) {

printf("Error: Could not write stop command (#%d)!", error);

return 1;

}

char selection;

printf("Write NVM an make a second verification round with programmed NVM? write and (v)erify / (w)rite only / do (n)ot write: ");

do selection = getchar(); while ((selection != 'v') && (selection != 'w') && (selection != 'n'));

getchar();

if ((selection == 'v') || (selection == 'w')) {

printf("Program settings with calibration and reconnect\n");

if ((error = pssc\_com\_cfg\_nvm\_write(handle, &cfg\_file))) {

printf("Error: Could not program configuration (#%d)!", error);

return 1;

}

if (selection == 'v') {

if ((error = pssc\_com\_scan\_socket(handle, 0, &addresses[0]))) {

printf("Error: Scan socket error (#%d)!\n", error);

pssc\_com\_get\_error\_message(error, info);

printf("\tError message: %s\n", info);

return 1;

}

else {

printf("\tScan ok.\n");

}

if (addresses[0] == -1) {

printf("Error: Scan socket succeeded but no device found.\n");

return 1;

}

printf("\nStart measuring\n");

if ((error = pssc\_com\_device\_measurement\_start(handle))) {

printf("Error: Could not write start command (#%d)!", error);

return 1;

}

double measured\_pressure\_ver[VER\_POINTS];

for (steps = 0; steps < VER\_POINTS; steps++) {

printf("\nStep %d: set up temperature %.1f dgC and pressure %.1f psi\n", steps + 1, temperature\_ver[steps], pressure\_ver[steps]);

printf("Press <enter> for verification step\n"); getchar();

printf("Measure ... ");

uint16\_t status;

int16\_t dsp\_s, dsp\_t;

error = pssc\_com\_read\_spt(handle, &status, &dsp\_s, &dsp\_t);

if (error != 0) {

measured\_pressure\_ver[steps] = 0;

printf("failed!\n");

}

else {

measured\_pressure\_ver[steps] = (double)((int16\_t)dsp\_s);

printf("done\n");

}

}

printf("\nVerify\n");

cal\_verification veri;

veri.min = 0;

veri.max = 0;

veri.avg = 0;

veri.rms = 0;

pssc\_cal\_verification(VER\_POINTS, measured\_pressure\_ver, expected\_pressure\_ver, &veri);

printf("veri.min = %g\n", veri.min); /\*\*< minimal error \*/

printf("veri.max = %g\n", veri.max); /\*\*< maximal error \*/

printf("veri.avg = %g\n", veri.avg); /\*\*< average error \*/

printf("veri.rms = %g\n", veri.rms); /\*\*< root mean square error \*/

printf("\nStop measuring\n");

if ((error = pssc\_com\_device\_measurement\_stop(handle))) {

printf("Error: Could not write stop command (#%d)!", error);

return 1;

}

}

}

printf("\nDisconnect from chip\n");

pssc\_com\_disconnect(&handle);

return 0;

}

//! [single chip calibration]