/\* ###################################################################

\*\* Filename : main.c

\*\* Project : Test\_code

\*\* Processor : MK20DX128VLH5

\*\* Version : Driver 01.01

\*\* Compiler : GNU C Compiler

\*\* Date/Time : 2019-10-24, 14:50, # CodeGen: 0

\*\* Abstract :

\*\* Main module.

\*\* This module contains user's application code.

\*\* Settings :

\*\* Contents :

\*\* No public methods

\*\*

\*\* ###################################################################\*/

/\*!

\*\* @file main.c

\*\* @version 01.01

\*\* @brief

\*\* Main module.

\*\* This module contains user's application code.

\*/

/\*!

\*\* @addtogroup main\_module main module documentation

\*\* @{

\*/

/\* MODULE main \*/

/\* Including needed modules to compile this module/procedure \*/

#include "Cpu.h"

#include "Events.h"

#include "Term1.h"

#include "Inhr1.h"

#include "ASerialLdd1.h"

#include "PWMD4X.h"

#include "PwmLdd1.h"

#include "TU1.h"

#include "PWMA1Y.h"

#include "PwmLdd2.h"

#include "PWMC3Z.h"

#include "PwmLdd3.h"

#include "BitE1X.h"

#include "BitIoLdd1.h"

#include "BitE0Y.h"

#include "BitIoLdd2.h"

#include "BitA5Z.h"

#include "BitIoLdd3.h"

#include "SLEEP.h"

#include "BitIoLdd4.h"

#include "PWMSpindle.h"

#include "PwmLdd4.h"

#include "BitD0M0.h"

#include "BitIoLdd5.h"

#include "BitC11M1.h"

#include "BitIoLdd6.h"

#include "BitC5M2.h"

#include "BitIoLdd7.h"

#include "Timer.h"

#include "RealTimeLdd1.h"

#include "TU2.h"

#include "BitRes.h"

#include "BitIoLdd8.h"

/\* Including shared modules, which are used for whole project \*/

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

/\* User includes (#include below this line is not maintained by Processor Expert) \*/

/\*lint -save -e970 Disable MISRA rule (6.3) checking. \*/

int main(void)

/\*lint -restore Enable MISRA rule (6.3) checking. \*/

{

/\* Write your local variable definition here \*/

char c; // User input

char buf [100]; // Collective user input

char type; // Type of command (percentage of movement range or millimeters)

char dir; // Direction of movement

int moveXL; // Closer to booleans than integers

int moveYF;

int moveZU;

int moveXR;

int moveYB;

int moveZD;

int time; // Time in milliseconds

int i; // Used for the buffer

double wait\_time; // Used for calculating how long to move for

double positionX; // Tracks position of spindle relative to initial position

double positionY;

double positionZ;

int dist; // Distance to move in millimeters

// the following functions move the spindle in the desired direction

void left(void){

PWMD4X\_SetRatio8(150);

BitE1X\_ClrVal();

}

void right(void){

PWMD4X\_SetRatio8(250);

BitE1X\_SetVal();

}

void forwards(void){

PWMA1Y\_SetRatio8(250);

BitE0Y\_ClrVal();

}

void back(void){

PWMA1Y\_SetRatio8(250);

BitE0Y\_SetVal();

}

void down(void){

PWMC3Z\_SetRatio8(250);

BitA5Z\_ClrVal();

}

void up(void){

PWMC3Z\_SetRatio8(250);

BitA5Z\_SetVal();

}

// Start and stop the spindle

void spindle\_start(void){

PWMSpindle\_SetRatio8(60);

}

void spindle\_stop(void){

PWMSpindle\_SetRatio8(0);

}

//Stops all movement

void stop(void){

PWMC3Z\_SetRatio8(0);

PWMA1Y\_SetRatio8(0);

PWMD4X\_SetRatio8(0);

PWMSpindle\_SetRatio8(0);

}

//Sends a given string

void send\_string(const char \*str)

{

int len, n;

len = strlen(str); // returns the number of chars in str

for (n = 0; n < len; n++) {

// send this character

Term1\_SendChar(str[n]);

}

}

/\*\*\* Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! \*\*\*/

PE\_low\_level\_init();

/\*\*\* End of Processor Expert internal initialization. \*\*\*/

/\* Write your code here \*/

/\* For example: for(;;) { } \*/

SLEEP\_SetVal(); // Sleep must be high for DRV8825 to operate, //same goes for Res

BitRes\_SetVal();

BitD0M0\_ClrVal(); // These three bits set the mode, Mode0 is used //here

BitC11M1\_ClrVal();

BitC5M2\_ClrVal();

PWMC3Z\_SetRatio8(0); // These three operate the movement of the spindle

PWMA1Y\_SetRatio8(0);

PWMD4X\_SetRatio8(0);

positionX = 0.00;

positionY = 0.00;

positionZ = 0.00;

send\_string("\r\n");

time = 0;

i = 0;

dist = 0.0;

Timer\_Enable();

send\_string("-----------------------------------------------");

send\_string("\r\n Manual:\r\n");

send\_string(" A = Left D = Right\r\n W = Forwards S = Back\r\n z = Down Z = Up\r\n");

send\_string(" H = reset position R = set position\r\n s = stop\r\n");

send\_string(" Automatic:\r\n Control syntax is:\r\n \"number type direction\"\r\n");

send\_string(" Directions are:\r\n l = Left r = Right\r\n f = Forwards");

send\_string(" b = Back\r\n u = Up d = Down\r\n");

send\_string(" j = Forward-Right k = Forwards-Left\r\n n = Back-right m = Back-Left\r\n");

send\_string(" Types are:\r\n p = Percentage m = Millimeters\r\n Example:\r\n '45 p l' moves 45% left\r\n");

send\_string("-----------------------------------------------");

send\_string("\r\n");

for (;;){

if (Term1\_KeyPressed() == 1){

Term1\_ReadChar(&c);

if (c == 'D'){

moveXL = 1;

left();

Timer\_Reset();

send\_string("Left\r\n");

}

else if (c == 'A'){

moveXR = 1;

right();

Timer\_Reset();

send\_string("Right\r\n");

}

else if (c == 'S'){

moveYF = 1;

forwards();

Timer\_Reset();

send\_string("Forwards\r\n");

}

else if (c == 'W'){

moveYB = 1;

back();

Timer\_Reset();

send\_string("Back\r\n");

}

else if (c == 'z'){

moveZD = 1;

down();

Timer\_Reset();

send\_string("Down\r\n");

}

else if (c == 'Z'){

moveZU = 1;

up();

Timer\_Reset();

send\_string("Up\r\n");

}

else if (c == 'g'){

spindle\_start();

send\_string("Go\r\n");

}

else if (c == 'G'){

spindle\_stop();

send\_string("End\r\n");

}

else if (c == 'H'){ // Resets the position of the spindle to home

if (positionX > 0){

right();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* positionX; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX - (time \* 0.0065); //0.0065 mm/ms

send\_string("X = 0\r\n");

}

else if (positionX < 0){

left();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = -153.846 \* positionX; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX + (time \* 0.0065); //0.0065 mm/ms

send\_string("X = 0\r\n");

}

if (positionY > 0){

back();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* positionY; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionY = positionY - (time \* 0.0065); //0.0065 mm/ms

send\_string("Y = 0\r\n");

}

else if (positionY < 0){

forwards();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = -153.846 \* positionY; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionY = positionY + (time \* 0.0065); //0.0065 mm/ms

send\_string("Y = 0\r\n");

}

if (positionZ > 0){

up();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* positionZ; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionZ = positionZ - (time \* 0.0065); //0.0065 mm/ms

send\_string("Z = 0\r\n");

}

else if (positionZ < 0){

down();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = -153.846 \* positionZ; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionZ = positionZ + (time \* 0.0065); //0.0065 mm/ms

send\_string("Z = 0\r\n");

}

}

else if (c == 'R'){ //Sets the current position as the initial position

positionX = 0;

positionY = 0;

positionZ = 0;

send\_string("Reset Position\r\n");

}

else if (c == 's'){

stop();

if (moveXL == 1){

Timer\_GetTimeMS(&time);

positionX = positionX + (time \* 0.0065); //0.0065 mm/ms

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

moveXL = 0;

}

if (moveXR == 1){

Timer\_GetTimeMS(&time);

positionX = positionX - (time \* 0.0065);

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

moveXR = 0;

}

if (moveYF == 1){

Timer\_GetTimeMS(&time);

positionY = positionY + (time \* 0.0065);

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

moveYF = 0;

}

if (moveYB == 1){

Timer\_GetTimeMS(&time);

positionY = positionY - (time \* 0.0065);

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

moveYB = 0;

}

if (moveZD == 1){

Timer\_GetTimeMS(&time);

positionZ = positionZ + (time \* 0.0065);

send\_string("Z = ");

Term1\_SendFloatNum(positionZ);

send\_string("\r\n");

moveZD = 0;

}

if (moveZU == 1){

Timer\_GetTimeMS(&time);

positionZ = positionZ - (time \* 0.0065);

send\_string("Z = ");

Term1\_SendFloatNum(positionZ);

send\_string("\r\n");

moveZU = 0;

}

send\_string("Stop\r\n");

}

else if (c == '\r') // Enter key

{

send\_string("\r\n");

buf[i] = 0;

i = 0;

time = 0;

if (sscanf(buf, "%i %c %c", &dist, &type, &dir) == 3){ // This is the command syntax

if ((type == 'p') || (type == 'm')) //These are the two types (percentage and millimeter)

{

if(dir == 'l')

{

if (type == 'm')

{

left();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX + (time \* 0.0065);;

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

}

else

{

left();

Timer\_Reset();

dist = (double)dist \* 1.8;

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX + (time \* 0.0065);;

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

}

}

else if(dir == 'r')

{

if (type == 'm')

{

right();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX - (time \* 0.0065);;

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

}

else

{

right();

Timer\_Reset();

dist = (double)dist \* 1.8;

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX - (time \* 0.0065);;

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

}

}

else if(dir == 'f')

{

if (type == 'm')

{

forwards();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionY = positionY + (time \* 0.0065);

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

else

{

forwards();

Timer\_Reset();

//100/100.0 = 1

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionY = positionY + (time \* 0.0065);

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

}

else if(dir == 'b')

{

if (type == 'm')

{

back();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //1538.46 ms/cm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionY = positionY - dist;

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

else

{

back();

Timer\_Reset();

//100/100.0 = 1

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionY = positionY - (time \* 0.0065);

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

}

else if(dir == 'd')

{

if (type == 'm')

{

down();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionZ = positionZ + (time \* 0.0065);

send\_string("Z = ");

Term1\_SendFloatNum(positionZ);

send\_string("\r\n");

}

else

{

down();

Timer\_Reset();

dist = (double)dist \* 4.5;

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionZ = positionZ + (time \* 0.0065);

send\_string("Z = ");

Term1\_SendFloatNum(positionZ);

send\_string("\r\n");

}

}

else if(dir == 'u')

{

if (type == 'm')

{

up();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionZ = positionZ - (time \* 0.0065);

send\_string("Z = ");

Term1\_SendFloatNum(positionZ);

send\_string("\r\n");

}

else

{

up();

Timer\_Reset();

(double)dist \* 4.5;

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionZ = positionZ - (time \* 0.0065);

send\_string("Z = ");

Term1\_SendFloatNum(positionZ);

send\_string("\r\n");

}

}

else if (dir == 'h'){

if (type == 'm'){

left();

forwards();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 76.923 \* dist; //153.846 \* 0.5 = 76.923 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

back();

time = 0;

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 76.923 \* dist; //153.846 \* 0.5 = 76.923 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

back();

time = 0;

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

right();

time = 0;

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

forwards();

time = 0;

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 153.846 \* dist; //153.846 ms/mm

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

}

else{

send\_string("House does not support that type\r\n");

}

}

else if(dir == 'k')

{

if (type == 'm')

{

left();

forwards();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 108.786 \* dist; //153.846 ms/mm \* 1/sqroot(2) = 108.786

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX + (time \* 0.0065);

positionY = positionY + (time \* 0.0065);

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

else

{

send\_string("Type P is not available for diagonals\r\n");

}

}

else if (dir == 'j')

{

if (type == 'm'){

right();

forwards();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 108.786 \* dist; //153.846 ms/mm \* 1/sqroot(2) = 108.786

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX - (time \* 0.0065);

positionY = positionY + (time \* 0.0065);

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

else

{

send\_string("Type P is not available for diagonals\r\n");

}

}

else if (dir == 'n')

{

if (type == 'm'){

right();

back();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 108.786 \* dist; //153.846 ms/mm \* 1/sqroot(2) = 108.786

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX - (time \* 0.0065);

positionY = positionY - (time \* 0.0065);

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

else

{

send\_string("Type P is not available for diagonals\r\n");

}

}

else if (dir == 'm')

{

if (type == 'm'){

left();

back();

Timer\_Reset();

Timer\_GetTimeMS(&time);

wait\_time = 108.786 \* dist; //153.846 ms/mm \* 1/sqroot(2) = 108.786

while(time < wait\_time)

{

Timer\_GetTimeMS(&time);

if (Term1\_KeyPressed() == 1){

break;

}

}

stop();

positionX = positionX + (time \* 0.0065);

positionY = positionY - (time \* 0.0065);

send\_string("X = ");

Term1\_SendFloatNum(positionX);

send\_string("\r\n");

send\_string("Y = ");

Term1\_SendFloatNum(positionY);

send\_string("\r\n");

}

else

{

send\_string("Type P is not available for diagonals\r\n");

}

}

else{

send\_string("I don't recognise that direction\r\n");

}

}

else

{

send\_string("I don't recognise that type\r\n");

}

}

else{

send\_string("I don't recognise that command\r\n");

}

}

else{

buf[i] = c;

Term1\_SendChar(c);

i = i + 1;

}

}

}

//for end

/\*\*\* Don't write any code pass this line, or it will be deleted during code generation. \*\*\*/

/\*\*\* RTOS startup code. Macro PEX\_RTOS\_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! \*\*\*/

#ifdef PEX\_RTOS\_START

PEX\_RTOS\_START(); /\* Startup of the selected RTOS. Macro is defined by the RTOS component. \*/

#endif

/\*\*\* End of RTOS startup code. \*\*\*/

/\*\*\* Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! \*\*\*/

for(;;){}

/\*\*\* Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! \*\*\*/

} /\*\*\* End of main routine. DO NOT MODIFY THIS TEXT!!! \*\*\*/

/\* END main \*/

/\*!

\*\* @}

\*/

/\*

\*\* ###################################################################

\*\*

\*\* This file was created by Processor Expert 10.5 [05.21]

\*\* for the Freescale Kinetis series of microcontrollers.

\*\*

\*\* ###################################################################

\*/