```
/**
  ******************************************************************************
  * @file    tim3_pwm_rc_servo_32ch/main.c
  * @author  Yasuo Kawachi
  * @version V1.0.0
  * @date    04/15/2009
  * @brief   Main program body
  ******************************************************************************
  * @copy
  *
  * Copyright 2008-2009 Yasuo Kawachi All rights reserved.
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted provided that the following conditions are met:
  *  1. Redistributions of source code must retain the above copyright notice,
  *  this list of conditions and the following disclaimer.
  *  2. Redistributions in binary form must reproduce the above copyright notice,
  *  this list of conditions and the following disclaimer in the documentation
  *  and/or other materials provided with the distribution.
  *
  * THIS SOFTWARE IS PROVIDED BY YASUO KAWACHI "AS IS" AND ANY EXPRESS OR IMPLIE  D
  * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
  * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
  * EVENT SHALL YASUO KAWACHI OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  * This software way contain the part of STMicroelectronics firmware.
  * Below notice is applied to the part, but the above BSD license is not.
  *
  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS
  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE
  * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY
  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING
  * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE
  * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
  *
  * COPYRIGHT 2009 STMicroelectronics
  */

/* Includes ------------------------------------------------------------------*/
#include "stm32f10x.h"
#include "platform_config.h"
#include "main.h"
#include "com_config.h"
#include "delay.h"
#include "lcd_function.h"

__IO int8_t RxData;
__IO int16_t Duty = 1350;
__IO int16_t Position[24];
__IO uint16_t Decoder_Select = 0;

int16_t motion[20][19];
int16_t pose[8][19];
int8_t a=0;                                //motion行数
```

```
int8_t b=0;                                //motion繰返し開始行(0～)
int8_t c=0;                                //motion終了開始行(0～)
int8_t d=0;                                //motion繰返し回数
int8_t m=0;                                //action番号
int mode;                                //mode番号
uint32_t speed=100;                        //motion apeed
int servo;                                //servo番号
int16_t level=0;                        //servo数値
int8_t O=1,P=1,Q=1,R=1,S=1,T=1,U=1;
//                                                                                                                               actionﾓｰﾄﾞ用ﾃﾞｰﾀ
//                        右脚 右足 右腰 右踵 右肩 右腕 右肘 右手 左脚 左足 左腰 左踵 左肩 左腕 左肘 左手 左右 前後   頭 右指       1行目の5つの数値は設定用ﾃﾞｰﾀ

int16_t motion1[5][20]= {{   5,   1,   3,   2, 100,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //aton1 Stand
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //b 繰返し開始行
                              {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //c 終了開始行
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t motion2[10][20]={{  10,   2,   8,   3, 500,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //action2 Squat
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         { -60,  60,   0,   0,   0, 300,   0, 600,  60, -60,   0,   0,   0, 300, 200,-100,-100, 100,   0,   0}, //b
                         { -60,  60,   0,   0,   0, 300,   0, 600,  60, -60,   0,   0,   0, 300, 200,-100,-100, 100,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0, 100,   0,   0,   0,   0,   0,   0,   0,-100,   0,   0,   0,   0},
                         { -60,  60,   0,   0,   0,-300,-200, 100,  60, -60,   0,   0,   0,-300,   0,-600, 100,-100,   0,   0},
                         { -60,  60,   0,   0,   0,-300,-200, 100,  60, -60,   0,   0,   0,-300,   0,-600, 100,-100,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0, 100,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //c
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t motion3[10][20]={{  10,   2,   8,   5, 350,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //action3 Swing
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0, 150, 150,-100,   0,-200,   0,   0,   0, 150, 150,   0,   0,-200,   0,-100, 100,   0,   0}, //b
                         {   0,   0, 150, 150,   0,   0,   0,   0,   0,   0, 150, 150,   0,   0,   0,   0,-100, 100,   0,   0},
                         {   0,   0, 150, 150,   0,   0,   0,   0,   0,   0, 150, 150,   0,   0,   0,   0,-100,   0,   0,   0},
                         {   0,   0,-150,-150,   0,   0, 200,   0,   0,   0,-150,-150,   0,   0, 200,   0, 100,-100,   0,   0},
                         {   0,   0,-150,-150,   0,   0,   0,   0,   0,   0,-150,-150,   0,   0,   0,   0, 100,-100,   0,   0},
                         {   0,   0,-150,-150,   0,   0,   0,   0,   0,   0,-150,-150,   0,   0,   0,   0, 100,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //c
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t motion4[14][20]={{  14,   3,  11,   5, 180,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //action4 Ashibumi
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         { -60,  60, 100, 100,   0,   0,   0,   0,  60, -60, 100, 100, 150,   0,   0,   0,-150,   0,   0,   0},
                          {-300, 300, 100, 100,-100, 300,   0,   0,  60, -60, 100, 100, 150, 300,   0,   0,-250,   0,   0,   0}, //b
                         { -60,  60, 100, 100,-100, 300,   0,   0,  60, -60, 100, 100, 150, 300,   0,   0,-250,   0,   0,   0},
                         { -60,  60, 100, 100,-150,   0,   0,   0,  60, -60, 100, 100, 100,   0,   0,   0,-150,   0,   0,   0},
                         { -60,  60,-100,-100,-150,   0,   0,   0,  60, -60,-100,-100, 100,   0,   0,   0, 150,   0,   0,   0},
                         { -60,  60,-200,-200,-150,-300,   0,   0, 300,-300,-200,-200, 100,-300,   0,   0, 250,   0,   0,   0},
                         { -60,  60,-200,-200,-150,-300,   0,   0,  60, -60,-200,-200, 100,-300,   0,   0, 250,   0,   0,   0},
                         { -60,  60,-100,-100,-100,   0,   0,   0,  60, -60,-100,-100, 150,   0,   0,   0, 150,   0,   0,   0},
                         { -60,  60, 100, 100,-100,   0,   0,   0,  60, -60, 100, 100, 150,   0,   0,   0,-150,   0,   0,   0},
                         { -60,  60, 100, 100,-100,   0,   0,   0,  60, -60, 100, 100, 100,   0,   0,   0,-150,   0,   0,   0}, //c
                         { -60,  60,-100,-100,   0,   0,   0,   0,  60, -60,-100,-100,   0,   0,   0,   0, 100,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t motion5[14][20]={{  14,   3,  11,   5, 220,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //action5 Walk
                          {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {-170, 170, 100, 100,-100,   0,   0,   0, 170,-170, 100, 100, 100,   0,   0,   0,-150,   0,   0,   0},
                         {-350, 350, 100, 100,-100, 300,   0,   0, 170,-170, 100, 100, 400, 300,   0,-300,-200,   0,   0,   0}, //b
                         {-460, 200, 100, 100,-100, 300,   0,   0, 170,-170, 100, 100, 400, 300,   0,-300,-200,-100,   0,   0},
                         {-240,  20, 100, 100,-100,   0,   0,   0,  20,-240, 100, 100, 100,   0,   0,   0,-150,   0,   0,   0},
                         {-240,  20,-100,-100,-100,   0,   0,   0,  20,-240,-100,-100, 100,   0,   0,   0, 150,   0,   0,   0},
                         {-170, 170,-200,-200,-400,-300,   0, 300, 350,-350,-200,-200, 100,-300,   0,   0, 200,   0,   0,   0},
                         {-170, 170,-200,-200,-400,-300,   0, 300, 460,-200,-200,-200, 100,-300,   0,   0, 200,-100,   0,   0},
                         { -20, 240,-100,-100,-100,   0,   0,   0, 240, -20,-100,-100, 100,   0,   0,   0, 150,   0,   0,   0},
                         { -20, 240, 100, 100,-100,   0,   0,   0, 240, -20, 100, 100, 100,   0,   0,   0,-150,   0,   0,   0},
                         {-170, 170, 100, 100,-100,   0,   0,   0, 240, -20, 100, 100, 100,   0,   0,   0,-150,-200,   0,   0}, //c
                         {-170, 170,-100,-100,   0,   0,   0,   0, 170, 170,-100,-100,   0,   0,   0,   0, 100,-100,   0,   0},

                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t trim[20]=        { -50, -30, -60,   0,   0,  40, -40, -20,  30, -60, -10,  50, -20,  30,   0,  50,   0,   0,   0,   0}; //ﾄﾘﾑ調整用ﾃﾞｰﾀ

//                        右脚 右足 右腰 右踵 右肩 右腕 右肘 右手 左脚 左足 左腰 左踵 左肩 左腕 左肘 左手 左右 前後   頭 右指    //poseﾓｰﾄﾞ用ﾃﾞｰﾀ
int16_t pose1[8][20]=        {{   8,   1,   5,   1, 300,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //pose1
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //b
                         {   0,   0,   0,   0,-100,-500,   0,   0,   0,   0,   0,   0, 100, 500,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,-100,-600,-600,   0,   0,   0,   0,   0, 100, 600, 600,   0,   0, 100,   0,   0},
                         {   0,   0,   0,   0,-100,-700,-600, 200,   0,   0,   0,   0, 100, 700, 600,-200,   0, 200,   0,   0},
                         {   0,   0,   0,   0,-100,-500,-600,   0,   0,   0,   0,   0, 100, 500, 600,   0,   0, 100,   0,   0}, //c
                         {   0,   0,   0,   0,   0,-500,   0,   0,   0,   0,   0,   0,   0, 500,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t pose2[12][20]=  {{  12,   1,   8,   1, 300,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //pose2 balance
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //b
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,-100,   0,   0,   0},
                         {   0,   0,  50,  50,   0,   0,   0,   0,   0,   0,  50,  50,   0,   0,   0,   0,-200,   0,   0,   0},
                         {   0,   0, 100, 100,   0,   0,   0,   0,   0,   0, 100, 100,   0,   0,   0,   0,-300,   0,   0,   0},
                         {   0,   0, 200, 200,   0,   0,   0,   0,   0,   0, 200, 200,   0,   0,   0,   0,-400,   0,   0,   0},
                         {-150, 150, 200, 200,   0,   0,   0,   0,   0,   0, 200, 200,   0,   0,   0,   0,-500,   0,   0,   0},
                         {-350, 350, 300, 300,   0,   0,   0,   0,   0,   0, 300, 300,   0,   0,   0,   0,-600,   0,   0,   0},
                         {-200, 200, 100, 100,   0,   0,   0,   0,   0,   0, 100, 100,   0,   0,   0,   0,-400,   0,   0,   0}, //c
                         {-100, 100,  50,  50,   0,   0,   0,   0,   0,   0,  50,  50,   0,   0,   0,   0,-200,   0,   0,   0},
                         { -50,  50,  50,  50,   0,   0,   0,   0,   0,   0,  50,  50,   0,   0,   0,   0,-100,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t pose3[12][20]=  {{  12,   1,   8,   1, 300,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //pose3 balance2
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //b
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 100,   0,   0,   0},
                         {   0,   0, -50, -50,   0,   0,   0,   0,   0,   0, -50, -50,   0,   0,   0,   0, 200,   0,   0,   0},
                         {   0,   0,-100,-100,   0,   0,   0,   0,   0,   0,-100,-100,   0,   0,   0,   0, 300,   0,   0,   0},
                         {   0,   0,-200,-200,   0,   0,   0,   0,   0,   0,-200,-200,   0,   0,   0,   0, 400,   0,   0,   0},
                         {   0,   0,-200,-200,   0,   0,   0,   0, 150,-150,-200,-200,   0,   0,   0,   0, 500,   0,   0,   0},
                         {   0,   0,-300,-300,   0,   0,   0,   0, 350,-350,-300,-300,   0,   0,   0,   0, 600,   0,   0,   0},
                         {   0,   0,-100,-100,   0,   0,   0,   0, 200,-200,-100,-100,   0,   0,   0,   0, 400,   0,   0,   0}, //c
                         {   0,   0, -50, -50,   0,   0,   0,   0, 100,-100, -50, -50,   0,   0,   0,   0, 200,   0,   0,   0},
                         {   0,   0, -50, -50,   0,   0,   0,   0,  50, -50, -50, -50,   0,   0,   0,   0, 100,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t pose4[8][20]=   {{   8,   1,   6,   1, 300,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //pose4
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //b
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //c
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t pose5[8][20]=   {{   8,   1,   6,   1, 300,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //pose5
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //b
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}, //c
                         {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}};

int16_t test1[20]=       {   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0}; //testﾓｰﾄﾞ用ﾃﾞｰﾀ

const int8_t Welcome_Message[] ="\r\nHello Cortex-M3/STM32 World!\r\n"
                                "Expand your creativity and enjoy making.\r\n\r\n"
                                "32 servos swings continuasly.\r\n\r\n";

/* Private function prototypes -----------------------------------------------*/
void GPIO_Configuration(void);
void NVIC_Configuration(void);
void Set_Select(uint16_t Decoder_Select);
void TIM3_Configuration(void);
void action(void);
void posing(void);
void test();
void test_action();

/* Private functions ---------------------------------------------------------*/
/**
  * @brief  Main program.
  * @param  None
  * @retval : None
  */

int main(void)                                                        //以下ﾒｲﾝﾌﾟﾛｸﾞﾗﾑ
{
    void BoardInit();
    NVIC_Configuration();
    COM_Configuration();
    TIM3_Configuration();
    GPIO_Configuration();
    LCD_Config();

    cprintf(Welcome_Message);
    delay_ms(100);

    TIM_SetCompare1(TIM3, NEUTRAL);
    TIM_SetCompare2(TIM3, NEUTRAL);
    TIM_SetCompare3(TIM3, NEUTRAL);
    TIM_SetCompare4(TIM3, NEUTRAL);

    mode = 0;                                                        //基本姿勢（直立）
    m = 1;
    action();

    while(1)                                                         //ﾓｰﾄﾞ・action・ｽﾋﾟｰﾄﾞ　選択ﾙｰﾌﾟ
    {
        lcd_xy(1,1);
        lcd_puts("** ROBO STM32 **");
        lcd_xy(1,2);
        lcd_puts(" WELCOM !!      ");
        delay_ms(1000);

        while(1)                                                    //mode選択ﾙｰﾌﾟ　0:action　1:pose　2:ｽﾄｯﾌﾟﾓｰｼｮﾝ　3:test
        {
            O=1;P=1;Q=1;R=1;T=1;
            lcd_xy(1,1);
            lcd_puts("select Mode ?   ");
            lcd_xy(1,2);
            lcd_puts("Mo=             ");
            lcd_xy(4,2);
            lcd_dataout(mode);

            while(1)                                                    //送信機からの指示読取りﾙｰﾌﾟ
            {
                O = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_8);
                P = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_11);
                S = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14);

                if(O == 0)
                {
                        mode=mode+1;                                    //mode up
                    if(mode>3) mode=3;
                    lcd_xy(4,2);                                    //LCD表示座標
                    lcd_dataout(mode);
                    delay_ms(300);
                }

                if(P == 0)
                {
                    mode=mode-1;                                    //mode down
                    if(mode<0) mode=0;
                    lcd_xy(4,2);
                    lcd_dataout(mode);
                    delay_ms(300);
                }

                if(S == 0)                                          //mode決定
                {
                    break;
                }
            }

            if(mode == 3)
            {
                test();                                             //testﾓｰﾄﾞｻﾌﾞﾙｰﾁﾝへ
                lcd_xy(1,1);                                        //ｻﾌﾞﾙｰﾁﾝから戻ったときの表示
                lcd_puts("END!            ");
                break;
            }

            lcd_xy(1,1);                                            //action番号選択の画面表示
            lcd_puts("select Action ? ");
            lcd_xy(6,2);
            lcd_puts("Ac=");
            lcd_xy(9,2);
            lcd_dataout(m);

            while(1)                                                    //action選択ﾙｰﾌﾟ
            {
                Q = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12);      //送信機からの指示読取り
                R = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_13);
                S = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14);
                if(Q == 0)
                {
                        m=m-1;                                          //action番号 down
                    if(m<1) m=1;
                     lcd_xy(9,2);
                    lcd_dataout(m);
                    delay_ms(300);
                }

                if(R == 0)
                {
                        m=m+1;                                          //action番号 up
                    if(m>5) m=5;
                     lcd_xy(9,2);
                    lcd_dataout(m);
                    delay_ms(300);
                }

                if(S == 0)                                          //決定
                {
                    break;
                }
            }

            if(mode==0 || mode==2)                                  //actionﾓｰﾄﾞとｽﾄｯﾌﾟﾓｰｼｮﾝﾓｰﾄﾞのとき
            {
                if(m==1)  speed=motion1[0][4];                      //action別にspeed の読み取り
                if(m==2)  speed=motion2[0][4];
                 if(m==3)  speed=motion3[0][4];
                if(m==4)  speed=motion4[0][4];
                if(m==5)  speed=motion5[0][4];
            }

            if(mode==1)                                             //poseﾓｰﾄﾞ　のとき
            {
                if(m==1)  speed=pose1[0][4];
                        if(m==2)  speed=pose2[0][4];
                         if(m==3)  speed=pose3[0][4];
                        if(m==4)  speed=pose4[0][4];
                        if(m==5)  speed=pose5[0][4];
            }

            while(1)                                               //speed設定ﾙｰﾌﾟ
            {
                lcd_xy(1,1);
                lcd_puts("select Speed ?  ");
                lcd_xy(11,2);
                lcd_puts("Sp=   ");
                lcd_xy(14,2);
                lcd_dataout(speed);

                while(1)
                {
                    O = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_8);
                        P = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_11);
                        S = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14);
                    T = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_15);

                    if(O == 0)
                        {
                            speed=speed+10;
                        if(speed>500) speed=500;
                        lcd_xy(14,2);
                        lcd_puts("   ");
                        lcd_xy(14,2);
                        lcd_dataout(speed);
                        delay_ms(300);
                    }

                    if(P == 0)
                        {
                            speed=speed-10;
                               if(speed<50) speed=50;
                        lcd_xy(14,2);
                        lcd_puts("   ");
                        lcd_xy(14,2);
                        lcd_dataout(speed);
                        delay_ms(300);
                        }

                    if(S == 0)  break;
                    if(T == 0)  break;
                }
                if(T == 0)  break;

                lcd_xy(1,1);
                lcd_puts("push E-key ?    ");

                while(1)
                {
                        S = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14);
                    T = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_15);

                    if(S == 0)
                        {
                        lcd_xy(1,1);
                        lcd_puts("Action !!       ");
                        action();
                        lcd_xy(1,1);
                        lcd_puts("push E- or M-key");
                    }

                    if(T == 0) break;
                }
            }
        }
    }
}                                                                    //mainﾙｰﾁﾝ終了

void action()                                                        //action用ｻﾌﾞﾙｰﾁﾝ
{
    if(mode==0 || mode==2)
    {
        switch(m)
            {
            case 1:                                                  //action設定用ﾃﾞｰﾀの読み取り
                a=motion1[0][0];                                     //action行数
                b=motion1[0][1];                                     //actionn繰返し開始行(0～)
                c=motion1[0][2];                                     //action終了開始行(0～)
                d=motion1[0][3];                                     //action繰返し回数
```

```
                  for(int i=0;i<a;i++)
                {
                        for(int j=0;j<19;j++)
                    {
                        motion[i][j]=motion1[i][j];                  //motionﾃﾞｰﾀの読み込み
                    }
                }
                break;

            case 2:
                a=motion2[0][0];
                b=motion2[0][1];
                c=motion2[0][2];
                d=motion2[0][3];

                for(int i=0;i<a;i++)
                {
                    for(int j=0;j<19;j++)
                    {
                            motion[i][j]=motion2[i][j];
                    }
                }
                break;

            case 3:
                a=motion3[0][0];
                b=motion3[0][1];
                c=motion3[0][2];
                d=motion3[0][3];

                for(int i=0;i<a;i++)
                {
                    for(int j=0;j<19;j++)
                    {
                             motion[i][j]=motion3[i][j];
                    }
                }
                break;

            case 4:
                a=motion4[0][0];
                b=motion4[0][1];
                c=motion4[0][2];
                d=motion4[0][3];

                for(int i=0;i<a;i++)
                {
                    for(int j=0;j<19;j++)
                    {
                            motion[i][j]=motion4[i][j];
                    }
                }
                break;

            case 5:
                a=motion5[0][0];
                b=motion5[0][1];
                c=motion5[0][2];
                d=motion5[0][3];
                for(int i=0;i<a;i++)
                {
                    for(int j=0;j<19;j++)
                    {
                            motion[i][j]=motion5[i][j];
                    }
                }
                break;

            default:
                    break;
        }
    }

    if(mode==1)
    {
        switch(m)
            {
            case 1:
                a=pose1[0][0];
                b=pose1[0][1];
                c=pose1[0][2];
                d=pose1[0][3];

                for(int i=0;i<a;i++)
                    {
                        for(int j=0;j<19;j++)
                    {
                        motion[i][j]=pose1[i][j];
                    }
                    }
                    break;

            case 2:
                a=pose2[0][0];
                b=pose2[0][1];
                c=pose2[0][2];
                d=pose2[0][3];

                for(int i=0;i<a;i++)
                    {
                    for(int j=0;j<19;j++)
                    {
                            motion[i][j]=pose2[i][j];
                    }
                    }
                    break;

            case 3:
                a=pose3[0][0];
                b=pose3[0][1];
                c=pose3[0][2];
                d=pose3[0][3];

                for(int i=0;i<a;i++)
                    {
                    for(int j=0;j<19;j++)
                    {
                             motion[i][j]=pose3[i][j];
                    }
                    }
                    break;

            case 4:
                a=pose4[0][0];
                b=pose4[0][1];
                c=pose4[0][2];
                d=pose4[0][3];

                for(int i=0;i<a;i++)
                    {
                    for(int j=0;j<19;j++)
                    {
                            motion[i][j]=pose4[i][j];
                    }
                    }
                   break;

            case 5:
                a=pose5[0][0];
                 b=pose5[0][1];
                c=pose5[0][2];
                d=pose5[0][3];
                    for(int i=0;i<a;i++)
                    {
                    for(int j=0;j<19;j++)
                    {
                             motion[i][j]=pose5[i][j];
                    }
                    }
                    break;

            default:
                    break;
        }
    }

    for(int i=1;i<b;i++)                            //繰返し初回目  b:actionﾃﾞｰﾀ行数
    {
            Position[0]  = motion[i][0]  +trim[0] ; //右脚
        Position[1]  = motion[i][1]  +trim[1] ; //右足
        Position[2]  = motion[i][2]  +trim[2] ; //右腰
        Position[3]  = motion[i][3]  +trim[3] ; //右踵

        Position[4]  = motion[i][4]  +trim[4] ; //右肩
        Position[5]  = motion[i][5]  +trim[5] ; //右腕
        Position[6]  = motion[i][6]  +trim[6] ; //右肘
        Position[7]  = motion[i][7]  +trim[7] ; //右手

        Position[8]  = motion[i][8]  +trim[8] ; //左脚
        Position[9]  = motion[i][9]  +trim[9] ; //左足
        Position[10] = motion[i][10] +trim[10]; //左腰
        Position[11] = motion[i][11] +trim[11]; //左踵

        Position[12] = motion[i][12] +trim[12]; //左肩
        Position[13] = motion[i][13] +trim[13]; //左腕
        Position[14] = motion[i][14] +trim[14]; //左肘
        Position[15] = motion[i][15] +trim[15]; //左手
        Position[16] = motion[i][16] +trim[16]; //左右
        Position[17] = motion[i][17] +trim[17]; //前後
        Position[18] = motion[i][18] +trim[18]; //頭
          Position[19] = motion[i][19] +trim[19]; //右指

        delay_ms(speed);
    }

    for(int j=0;j<d;j++)                       //2回目以降  d:繰返し回数
    {
            for(int i=b;i<c;i++)                   //繰返し開始行  motion行数
        {
            Position[0]  = motion[i][0]  +trim[0] ; //右脚
            Position[1]  = motion[i][1]  +trim[1] ; //右足
            Position[2]  = motion[i][2]  +trim[2] ; //右腰
            Position[3]  = motion[i][3]  +trim[3] ; //右踵

            Position[4]  = motion[i][4]  +trim[4] ; //右肩
            Position[5]  = motion[i][5]  +trim[5] ; //右腕
            Position[6]  = motion[i][6]  +trim[6] ; //右肘
            Position[7]  = motion[i][7]  +trim[7] ; //右手

            Position[8]  = motion[i][8]  +trim[8] ; //左脚
            Position[9]  = motion[i][9]  +trim[9] ; //左足
            Position[10] = motion[i][10] +trim[10]; //左腰
            Position[11] = motion[i][11] +trim[11]; //左踵

            Position[12] = motion[i][12] +trim[12]; //左肩
            Position[13] = motion[i][13] +trim[13]; //左腕
            Position[14] = motion[i][14] +trim[14]; //左肘
            Position[15] = motion[i][15] +trim[15]; //左手

            Position[16] = motion[i][16] +trim[16]; //左右
            Position[17] = motion[i][17] +trim[17]; //前後
            Position[18] = motion[i][18] +trim[18]; //頭
              Position[19] = motion[i][19] +trim[19]; //右指

            delay_ms(speed);

            if(mode==2)                                                //ｽﾄｯﾌﾟﾓｰｼｮﾝﾓｰﾄﾞ
            {
                lcd_xy(1,1);
                lcd_puts("pose! push E-key");

                while(1)
                {
                    S = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14);
                    T = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_15);

                    if(S == 0)
                    {
                        lcd_xy(1,1);
                        lcd_puts("next pose!      ");
                        break;
                    }

                    if(T == 0)  break;
                }
            }
            if(T == 0)  break;
        }

        if(T == 0)  break;
    }

    if(mode==1)                                                            //poseﾓｰﾄﾞ
    {
        lcd_xy(1,1);
               lcd_puts("pose! push E-key");

        while(1)
        {
            S = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14);
            T = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_15);

            if(S == 0) break;
            if(T == 0) break;
        }
    }

    for(int i=c;i<a;i++)                                                  //終了部分 c:終了開始行
    {
            Position[0]  = motion[i][0]  +trim[0] ; //右脚
        Position[1]  = motion[i][1]  +trim[1] ; //右足
        Position[2]  = motion[i][2]  +trim[2] ; //右腰
        Position[3]  = motion[i][3]  +trim[3] ; //右踵

        Position[4]  = motion[i][4]  +trim[4] ; //右肩
        Position[5]  = motion[i][5]  +trim[5] ; //右腕
        Position[6]  = motion[i][6]  +trim[6] ; //右肘
        Position[7]  = motion[i][7]  +trim[7] ; //右手

        Position[8]  = motion[i][8]  +trim[8] ; //左脚
        Position[9]  = motion[i][9]  +trim[9] ; //左足
        Position[10] = motion[i][10] +trim[10]; //左腰
        Position[11] = motion[i][11] +trim[11]; //左踵

        Position[12] = motion[i][12] +trim[12]; //左肩
        Position[13] = motion[i][13] +trim[13]; //左腕
        Position[14] = motion[i][14] +trim[14]; //左肘
        Position[15] = motion[i][15] +trim[15]; //左手

        Position[16] = motion[i][16] +trim[16]; //左右
        Position[17] = motion[i][17] +trim[17]; //前後
        Position[18] = motion[i][18] +trim[18]; //頭
          Position[19] = motion[i][19] +trim[19]; //右指

        delay_ms(speed);
    }
}

void test()                                                              //testﾓｰﾄﾞｻﾌﾞﾙｰﾁﾝ
{
    servo=0;
    level=0;
    lcd_xy(1,1);
    lcd_puts("set servo ?     ");
    lcd_xy(1,2);
    lcd_puts("servo=");
    lcd_xy(7,2);
    lcd_dataout(servo);
    lcd_xy(10,2);
    lcd_puts("A=");
    lcd_xy(12,2);

    if(test1[servo]<0) lcd_puts("-");
    else               lcd_puts(" ");
    lcd_dataout(abs(test1[servo]));

    while(1)                                                             //action番号選択
    {
        O = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_8);
        P = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_11);
        Q = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12);
        R = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_13);
        S = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14);
        T = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_15);
        U = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_9);

        if(P == 0)
        {
            servo=servo-1;                                          //servo番号 down
            if(servo<0) servo=19;

            lcd_xy(7,2);
            lcd_puts("  ");
            lcd_xy(7,2);
            lcd_dataout(servo);
            lcd_xy(12,2);
            lcd_puts("    ");
            lcd_xy(12,2);

            if(test1[servo]<0) lcd_puts("-");
            else               lcd_puts(" ");

            lcd_dataout(abs(test1[servo]));
            delay_ms(300);
        }

        if(O == 0)
        {
            servo=servo+1;                                         //servo番号up
            if(servo>19) servo=0;

            lcd_xy(7,2);
            lcd_puts("  ");
            lcd_xy(7,2);
            lcd_dataout(servo);
            lcd_xy(12,2);
            lcd_puts("    ");
            lcd_xy(12,2);

            if(test1[servo]<0) lcd_puts("-");                     //servo角度ﾏｲﾅｽ表示
            else               lcd_puts(" ");

            lcd_dataout(abs(test1[servo]));                       //servo角度の絶対値表示
            delay_ms(300);
        }
        if(Q == 0)
        {
            test1[servo]=test1[servo]+100;                        //servo角度100up
            if(test1[servo]>600) test1[servo]=600;

            lcd_xy(12,2);
            lcd_puts("    ");
            lcd_xy(12,2);

            if(test1[servo]<0) lcd_puts("-");
            else               lcd_puts(" ");

            lcd_dataout(abs(test1[servo]));
            delay_ms(300);
        }

        if(R == 0)
        {
                   test1[servo]=test1[servo]-100;                        //servo角度100down
            if(level<-600) level=-600;

            lcd_xy(12,2);
            lcd_puts("    ");
            lcd_xy(12,2);

            if(test1[servo]<0) lcd_puts("-");
            else               lcd_puts(" ");
            lcd_dataout(abs(test1[servo]));
            delay_ms(300);
        }

        if(U == 0)
        {
            test1[servo]=test1[servo]+10;                        //servo角度10up
            if(level>500) level=500;

            lcd_xy(12,2);
            lcd_puts("    ");
            lcd_xy(12,2);
            if(test1[servo]<0) lcd_puts("-");
            else               lcd_puts(" ");
            lcd_dataout(abs(test1[servo]));
            delay_ms(300);
        }

        if(S == 0)
        {
//          test1[servo]=level;
            test_action();
        }

        if(T == 0)
        {
            break;
        }
    }

    servo=0;
    level=0;
    for(servo=0;servo<20;servo++)                                //servoﾃﾞｰﾀをｸﾘｱ
    {
        test1[servo]=0;
    }
    test_action();
}

void test_action()
{
    Position[0]  = test1[0]  +trim[0] ; //右脚
    Position[1]  = test1[1]  +trim[1] ; //右足
    Position[2]  = test1[2]  +trim[2] ; //右腰
    Position[3]  = test1[3]  +trim[3] ; //右踵

    Position[4]  = test1[4]  +trim[4] ; //右肩
    Position[5]  = test1[5]  +trim[5] ; //右腕
    Position[6]  = test1[6]  +trim[6] ; //右肘
    Position[7]  = test1[7]  +trim[7] ; //右手

    Position[8]  = test1[8]  +trim[8] ; //左脚
    Position[9]  = test1[9]  +trim[9] ; //左足
    Position[10] = test1[10] +trim[10]; //左腰
    Position[11] = test1[11] +trim[11]; //左踵

    Position[12] = test1[12] +trim[12]; //左肩
    Position[13] = test1[13] +trim[13]; //左腕
    Position[14] = test1[14] +trim[14]; //左肘
    Position[15] = test1[15] +trim[15]; //左手

    Position[16] = test1[16] +trim[16]; //左右
    Position[17] = test1[17] +trim[17]; //前後
    Position[18] = test1[18] +trim[18]; //頭
    Position[19] = test1[19] +trim[19]; //右指
}

/**
  * @brief  Configure the nested vectored interrupt controller.
  * @param  None
  * @retval None
*/

void NVIC_Configuration(void)
{
          NVIC_InitTypeDef NVIC_InitStructure;
          /* Enable the TIM3 Interrupt */
         NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
          NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
          NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
          NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
          NVIC_Init(&NVIC_InitStructure);
}

/**
  * @brief  Configure the GPIO Pins.
  * @param  None
  * @retval : None
*/

void GPIO_Configuration(void)
{
```

```
          GPIO_InitTypeDef GPIO_InitStructure;

          //Supply APB2 Clock
          RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC , ENABLE);
          GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2;
          GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
          GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
          GPIO_Init(GPIOC, &GPIO_InitStructure);

        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB , ENABLE);
        GPIO_InitStructure.GPIO_Pin =GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11|GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15;
          GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
          GPIO_Init(GPIOB, &GPIO_InitStructure);
}

/**
  * @brief  Configure TIM3
  * @param  None
  * @retval : None
*/

void TIM3_Configuration(void)
{
          GPIO_InitTypeDef GPIO_InitStructure;
          TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
          TIM_OCInitTypeDef  TIM_OCInitStructure;

          //Supply APB1 Clock
          RCC_APB1PeriphClockCmd(TIM3_RCC , ENABLE);
          //Supply APB2 Clock
          RCC_APB2PeriphClockCmd(TIM3_CH12_GPIO_RCC | TIM3_CH34_GPIO_RCC , ENABLE);

          /* GPIO Configuration:TIM3 Channel1 as alternate function push-pull */
          GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
          GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
          GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

          GPIO_Init(TIM3_CH12_PORT, &GPIO_InitStructure);
          GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9 ;
          GPIO_Init(TIM3_CH34_PORT, &GPIO_InitStructure);

        FullRemap_TIM3_Configuration();

          /* ---------------------------------------------------------------------------
    TIM3 Configuration: Output Compare Toggle Mode:
    TIM3CLK = 72 MHz, Prescaler = 60, TIM3 counter clock = 1.2MHz
          ----------------------------------------------------------------------------*/

          /* Time base configuration */
          // 2500us cycle
         TIM_TimeBaseStructure.TIM_Period = 2609;//2649;//2249;//17999;//2249;//PWM_CYCLE;
         TIM_TimeBaseStructure.TIM_Prescaler =22; //26; //79;
         TIM_TimeBaseStructure.TIM_ClockDivision = 0;
          TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
          TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

          /* Output Compare Toggle Mode configuration: Channel1 */
          TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;//PWM2;
          TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
          TIM_OCInitStructure.TIM_Pulse = 0;
          TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
          TIM_OC1Init(TIM3, &TIM_OCInitStructure);
          TIM_OC1PreloadConfig(TIM3, TIM_OCPreload_Disable);

          /* Output Compare Toggle Mode configuration: Channel2 */
          TIM_OCInitStructure.TIM_Pulse = 0;
          TIM_OC2Init(TIM3, &TIM_OCInitStructure);
          TIM_OC2PreloadConfig(TIM3, TIM_OCPreload_Disable);

          /* Output Compare Toggle Mode configuration: Channel3 */
          TIM_OCInitStructure.TIM_Pulse = 0;
          TIM_OC3Init(TIM3, &TIM_OCInitStructure);
          TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Disable);

          /* Output Compare Toggle Mode configuration: Channel4 */
          TIM_OCInitStructure.TIM_Pulse = 0;
          TIM_OC4Init(TIM3, &TIM_OCInitStructure);
          TIM_OC4PreloadConfig(TIM3, TIM_OCPreload_Disable);


          /* TIM IT enable */
          TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);

          /* TIM enable counter */
          TIM_Cmd(TIM3, ENABLE);
}

/**
  * @brief  Set select pin of decoder
  * @param  None
  * @retval None
*/

void Set_Select(uint16_t Decoder_Select)
{
          GPIO_Write(GPIOC, (GPIO_ReadOutputData(GPIOC) & 0x1111111111111000) | Decoder_Select);
}
```