```c
#include "flight_control.h"
#include "rc.h"
#include <math.h>

float pid_x_integ1 = 0;
float pid_y_integ1 = 0;
float pid_z_integ1 = 0;
float pid_x_integ2 = 0;
float pid_y_integ2 = 0;
float pid_z_integ2 = 0;
float pid_x_pre_error2 = 0;
float pid_y_pre_error2 = 0;
float pid_z_pre_error2 = 0;
float pid_x_pre_deriv = 0;
float pid_y_pre_deriv = 0;

extern int16_t gTHR;
int16_t motor_thr;
float dt_recip;


float rp_rctrl_Fa[] = RP_RCTRL_FA;
float rp_rctrl_Ga[] = RP_RCTRL_GA;
float rp_rctrl_Ha[] = RP_RCTRL_HA;
float rp_rctrl_Aod[] = RP_RCTRL_AOD;
float rp_rctrl_Bod[] = RP_RCTRL_BOD;
float rp_rctrl_Cod[] = RP_RCTRL_COD;
float egx_integ = 0;    // gxの追従偏差の積分値
float egy_integ = 0;    // gyの追従偏差の積分値
float dmxe      = 0;            // MXの推定値
float dmye      = 0;            // MYの推定値


void PIDControlInit(P_PI_PIDControlTypeDef *pid)
{
  pid->ts = PID_SAMPLING_TIME;

  pid->x_kp1 = PITCH_PID_KP1;
  pid->x_ki1 = PITCH_PID_KI1;
  pid->x_i1_limit = PITCH_PID_I1_LIMIT;
  pid->x_kp2 = PITCH_PID_KP2;
  pid->x_ki2 = PITCH_PID_KI2;
  pid->x_kd2 = PITCH_PID_KD2;
  pid->x_i2_limit = PITCH_PID_I2_LIMIT;
  pid->x_s1 = 0;
  pid->x_s2 = 0;

  pid->y_kp1 = ROLL_PID_KP1;
  pid->y_ki1 = ROLL_PID_KI1;
  pid->y_i1_limit = ROLL_PID_I1_LIMIT;
```

追加：状態フィードバック制御で用いる変数の宣言および初期化

```
  pid->y_kp2 = ROLL_PID_KP2;
  pid->y_ki2 = ROLL_PID_KI2;
  pid->y_kd2 = ROLL_PID_KD2;
  pid->y_i2_limit = ROLL_PID_I2_LIMIT;
  pid->y_s1 = 0;
  pid->y_s2 = 0;

  pid->z_kp1 = YAW_PID_KP1;
  pid->z_ki1 = YAW_PID_KI1;
  pid->z_i1_limit = YAW_PID_I1_LIMIT;
  pid->z_kp2 = YAW_PID_KP2;
  pid->z_ki2 = YAW_PID_KI2;
  pid->z_kd2 = YAW_PID_KD2;
  pid->z_i2_limit = YAW_PID_I2_LIMIT;
  pid->z_s1 = 0;
  pid->z_s2 = 0;
}

void FlightControlPID(EulerAngleTypeDef *euler_rc, EulerAngleTypeDef *euler_ahrs, Gyro_Rad *gyro_rad, AHRS_State_TypeDef *ahrs, P_PI_PIDControlTypeDef *pid, MotorControlTypeDef *motor_pwm)
{
  float error, deriv;

  if(gTHR<MIN_THR)
  {
    pid_x_integ1 = 0;
    pid_y_integ1 = 0;
    pid_z_integ1 = 0;
    pid_x_integ2 = 0;
    pid_y_integ2 = 0;
    pid_z_integ2 = 0;
  }


  //x-axis pid
  error = euler_rc->thx - euler_ahrs->thx;
  pid_x_integ1 += error*pid->ts;
  if(pid_x_integ1 > pid->x_i1_limit)
    pid_x_integ1 = pid->x_i1_limit;
  else if(pid_x_integ1 < -pid->x_i1_limit)
    pid_x_integ1 = -pid->x_i1_limit;
  pid->x_s1 =  pid->x_kp1*error + pid->x_ki1*pid_x_integ1;

  error = euler_rc->thx - gyro_rad->gx;
  pid_x_integ2 += error*pid->ts;
  if(pid_x_integ2 > pid->x_i2_limit)
    pid_x_integ2 = pid->x_i2_limit;
  else if(pid_x_integ2 < -pid->x_i2_limit)
    pid_x_integ2 = -pid->x_i2_limit;
```

```
deriv = error - pid_x_pre_error2;
pid_x_pre_error2 = error;
pid->x_s2 = pid->x_kp2*error + pid->x_ki2*pid_x_integ2 + pid->x_kd2*deriv;

if(pid->x_s2 > MAX_ADJ_AMOUNT)  pid->x_s2 = MAX_ADJ_AMOUNT;
if(pid->x_s2 < -MAX_ADJ_AMOUNT)  pid->x_s2 = -MAX_ADJ_AMOUNT;


//y-axis pid
error = euler_rc->thy - euler_ahrs->thy;
pid_y_integ1 += error*pid->ts;
if(pid_y_integ1 > pid->y_i1_limit)
  pid_y_integ1 = pid->y_i1_limit;
else if(pid_y_integ1 < -pid->y_i1_limit)
  pid_y_integ1 = -pid->y_i1_limit;
pid->y_s1 =  pid->y_kp1*error + pid->y_ki1*pid_y_integ1;

error = euler_rc->thy - gyro_rad->gy;
pid_y_integ2 += error*pid->ts;
if(pid_y_integ2 > pid->y_i2_limit)
  pid_y_integ2 = pid->y_i2_limit;
else if(pid_y_integ2 < -pid->y_i2_limit)
  pid_y_integ2 = -pid->y_i2_limit;
deriv = error - pid_y_pre_error2;
pid_y_pre_error2 = error;
pid->y_s2 = pid->y_kp2*error + pid->y_ki2*pid_y_integ2 + pid->y_kd2*deriv;

if(pid->y_s2 > MAX_ADJ_AMOUNT)  pid->y_s2 = MAX_ADJ_AMOUNT;
if(pid->y_s2 < -MAX_ADJ_AMOUNT)  pid->y_s2 = -MAX_ADJ_AMOUNT;


//z-axis pid
error = euler_rc->thz - gyro_rad->gz;
pid_z_integ2 += error*pid->ts;
if(pid_z_integ2 > pid->z_i2_limit)
  pid_z_integ2 = pid->z_i2_limit;
else if(pid_z_integ2 < -pid->z_i2_limit)
  pid_z_integ2 = -pid->z_i2_limit;
deriv = error - pid_z_pre_error2;
pid_z_pre_error2 = error;
pid->z_s2 = pid->z_kp2*error + pid->z_ki2*pid_y_integ2 + pid->z_kd2*deriv;

if(pid->z_s2 > MAX_ADJ_AMOUNT)  pid->z_s2 = MAX_ADJ_AMOUNT;
if(pid->z_s2 < -MAX_ADJ_AMOUNT)  pid->z_s2 = -MAX_ADJ_AMOUNT;

#ifdef MOTOR_DC

  motor_thr = 0.33333f*gTHR + 633.333f;           //Devo7E >> 630 to 1700

#endif
```

```

  #ifdef MOTOR_ESC

    //motor_thr = 0.28f*gTHR + 750.0f;                  //TGY-i6 remocon and external ESC STEVAL-SC001V1
    //motor_thr = 0.28f*gTHR + 850.0f;                  //TGY-i6 remocon and external ESC Afro12A
    motor_thr = 0.32f*gTHR + 900.0f;                    //TGY-i6 remocon and external ESC Afro12A

  #endif


  motor_pwm->motor1_pwm = motor_thr - pid->x_s2 - pid->y_s2 + pid->z_s2 + MOTOR_OFF1;
  motor_pwm->motor2_pwm = motor_thr + pid->x_s2 - pid->y_s2 - pid->z_s2 + MOTOR_OFF2;
  motor_pwm->motor3_pwm = motor_thr + pid->x_s2 + pid->y_s2 + pid->z_s2 + MOTOR_OFF3;
  motor_pwm->motor4_pwm = motor_thr - pid->x_s2 + pid->y_s2 - pid->z_s2 + MOTOR_OFF4;


}

void FlightControlPID_OuterLoop(EulerAngleTypeDef *euler_rc, EulerAngleTypeDef *euler_ahrs, AHRS_State_TypeDef *ahrs, P_PI_PIDControlTypeDef *pid)
{
  float error;

  if(gTHR<MIN_THR)
  {
    pid_x_integ1 = 0;
    pid_y_integ1 = 0;
    pid_z_integ1 = 0;
  }

  //x-axis pid
  error = euler_rc->thx - euler_ahrs->thx;
  pid_x_integ1 += error*pid->ts;
  if(pid_x_integ1 > pid->x_i1_limit)
    pid_x_integ1 = pid->x_i1_limit;
  else if(pid_x_integ1 < -pid->x_i1_limit)
    pid_x_integ1 = -pid->x_i1_limit;
  pid->x_s1 =  pid->x_kp1*error + pid->x_ki1*pid_x_integ1;

  //y-axis pid
  error = euler_rc->thy - euler_ahrs->thy;
  pid_y_integ1 += error*pid->ts;
  if(pid_y_integ1 > pid->y_i1_limit)
    pid_y_integ1 = pid->y_i1_limit;
  else if(pid_y_integ1 < -pid->y_i1_limit)
    pid_y_integ1 = -pid->y_i1_limit;
  pid->y_s1 =  pid->y_kp1*error + pid->y_ki1*pid_y_integ1;
```

```
195
196   //z-axis pid
197   error = euler_rc->thz - euler_ahrs->thz;
198   pid_z_integ1 += error*pid->ts;
199   if(pid_z_integ1 > pid->z_i1_limit)
200     pid_z_integ1 = pid->z_i1_limit;
201   else if(pid_z_integ1 < -pid->z_i1_limit)
202     pid_z_integ1 = -pid->z_i1_limit;
203   pid->z_s1 =  pid->z_kp1*error + pid->z_ki1*pid_z_integ1;
204 }
205
206 /* start: add */
207 /*
208  * TYP_CTRL: 制御則の種類
209  * 0 = ST 純正（デフォルト），1 = 積分型最適サーボ
210  */
211 #define TYP_CTRL        0
212 /* end: add */
```

追加：FlightControlPID_innerLoop() 関数内で使用する制御則を変更できるようにする定数．1の場合，状態フィードバック制御となる

```
213
214 void   FlightControlPID_innerLoop(EulerAngleTypeDef   *euler_rc,   Gyro_Rad   *gyro_rad, AHRS_State_TypeDef *ahrs, P_PI_PIDControlTypeDef *pid, MotorControlTypeDef *motor_pwm)
215 {
216   float error, deriv;
```

インナ・ループ角速度制御

```
217 /* start: add */
218 #if TYP_CTRL == 1
219   float u1_F, u2_F;
220   float u1_G, u2_G;
221   float u1_H, u2_H;
222   float mx;                         // モーメント MX 推定値
223   float my;                         // モーメント MY 推定値
224   float egx;              // gx の追従偏差
225   float egy;              // gy の追従偏差
226   float x_s2_l, y_s2_l; // x_s2, y_s2 の値（状態推定器用）
227   float dmxe_next;        // dmxe の 1 サンプル更新後の値
228   float dmye_next;        // dmye の 1 サンプル更新後の値
229 #endif  /* TYP_CTRL */
230 /* end: add */
```

追加：状態フィードバック制御で使用する変数を宣言する

```
231
232   if(gTHR<MIN_THR)
233   {
234     pid_x_integ2 = 0;
235     pid_y_integ2 = 0;
236     pid_z_integ2 = 0;
237   }
238
239   dt_recip = 1/pid->ts;
240
241 /* start: add */
242 #if TYP_CTRL == 1
243
```

ここから300行目まで追加：
状態フィードバック制御を実行する

```
244   if(gTHR<MIN_THR)
245   {
246     egx_integ = 0;
247     egy_integ = 0;
248     dmxe = 0;
249     dmye = 0;
250   }
251
252   //XY Axis
253   mx = rp_rctrl_Cod[0] * dmxe + rp_rctrl_Cod[1] * dmye;
254   my = rp_rctrl_Cod[2] * dmxe + rp_rctrl_Cod[3] * dmye;
255   // 注：状態推定器の状態変数dmxe，dmyeの1ステップ更新は本関数の最後で行う。
256   u1_F = rp_rctrl_Fa[0] * mx + rp_rctrl_Fa[1] * my + rp_rctrl_Fa[2] * gyro_rad->gx +
  rp_rctrl_Fa[3] * gyro_rad->gy;
257   u2_F = rp_rctrl_Fa[4] * mx + rp_rctrl_Fa[5] * my + rp_rctrl_Fa[6] * gyro_rad->gx +
  rp_rctrl_Fa[7] * gyro_rad->gy;
258   egx = pid->x_s1 - gyro_rad->gx;
259   egy = pid->y_s1 - gyro_rad->gy;
260   egx_integ += egx * pid->ts;
261   if(egx_integ > EGX_I_LIMIT)
262     egx_integ = EGX_I_LIMIT;
263   else if(egx_integ < -EGX_I_LIMIT)
264     egx_integ = -EGX_I_LIMIT;
265   egy_integ += egy * pid->ts;
266   if(egy_integ > EGY_I_LIMIT)
267     egy_integ = EGY_I_LIMIT;
268   else if(egy_integ < -EGY_I_LIMIT)
269     egy_integ = -EGY_I_LIMIT;
270   u1_G = rp_rctrl_Ga[0] * egx_integ + rp_rctrl_Ga[1] * egy_integ;
271   u2_G = rp_rctrl_Ga[2] * egx_integ + rp_rctrl_Ga[3] * egy_integ;
272   u1_H = rp_rctrl_Ha[0] * pid->x_s1 + rp_rctrl_Ha[1] * pid->y_s1;
273   u2_H = rp_rctrl_Ha[2] * pid->x_s1 + rp_rctrl_Ha[3] * pid->y_s1;
274   pid->x_s2 = u1_F + u1_G + u1_H;
275   pid->y_s2 = u2_F + u2_G + u2_H;
276
277   if(pid->x_s2 > MAX_ADJ_AMOUNT)  pid->x_s2 = MAX_ADJ_AMOUNT;
278   if(pid->x_s2 < -MAX_ADJ_AMOUNT)  pid->x_s2 = -MAX_ADJ_AMOUNT;
279
280   if(pid->y_s2 > MAX_ADJ_AMOUNT)  pid->y_s2 = MAX_ADJ_AMOUNT;
281   if(pid->y_s2 < -MAX_ADJ_AMOUNT)  pid->y_s2 = -MAX_ADJ_AMOUNT;
282
283   x_s2_l = pid->x_s2;
284   if (x_s2_l > X_S2_LIMIT_0)
285     x_s2_l = X_S2_LIMIT_0;
286   else if (x_s2_l < -X_S2_LIMIT_0)
287     x_s2_l = -X_S2_LIMIT_0;
288   y_s2_l = pid->y_s2;
289   if (y_s2_l > Y_S2_LIMIT_0)
290     y_s2_l = Y_S2_LIMIT_0;
291   else if (y_s2_l < -Y_S2_LIMIT_0)
```

スロットルを下げているとき（着陸中など）は追従偏差の積分値と状態推定器の変数を初期化する

X軸（ピッチ）角速度の制御とY軸（ロール）角速度の制御を同時に行う

状態推定器の出力方程式

状態フィードバック部

追従偏差を求める

追従偏差の積分を求める（X軸）

追従偏差の積分値が過大にならないようリミッタをかけて飽和させる（X軸）

Y軸についても同様に，追従偏差の積分を求める

積分制御部

フィードフォワード制御部

制御出力

制御出力に対するリミッタ

状態推定器の状態方程式（～298行目）

```
292      y_s2_l = -Y_S2_LIMIT_0;
293    dmxe_next = rp_rctrl_Aod[0] * dmxe + rp_rctrl_Aod[1] * dmye +
294      rp_rctrl_Bod[0] * x_s2_l + rp_rctrl_Bod[1] * y_s2_l;
295    dmye_next = rp_rctrl_Aod[2] * dmxe + rp_rctrl_Aod[3] * dmye +
296      rp_rctrl_Bod[2] * x_s2_l + rp_rctrl_Bod[3] * y_s2_l;
297    dmxe = dmxe_next;
298    dmye = dmye_next;
299
300  #endif  /* TYP_CTRL */
301  /* end: add */
302
303  /* start: delete */
304  #if TYP_CTRL == 0
305    //X Axis
306    error = pid->x_s1 - gyro_rad->gx;
307    pid_x_integ2 += error*pid->ts;
308    if(pid_x_integ2 > pid->x_i2_limit)
309      pid_x_integ2 = pid->x_i2_limit;
310    else if(pid_x_integ2 < -pid->x_i2_limit)
311      pid_x_integ2 = -pid->x_i2_limit;
312    deriv = (error - pid_x_pre_error2)*dt_recip;
313    pid_x_pre_error2 = error;
314    deriv = pid_x_pre_deriv + (deriv - pid_x_pre_deriv)*D_FILTER_COFF;
315    pid_x_pre_deriv = deriv;
316    pid->x_s2 = pid->x_kp2*error + pid->x_ki2*pid_x_integ2 + pid->x_kd2*deriv;
317
318    if(pid->x_s2 > MAX_ADJ_AMOUNT)  pid->x_s2 = MAX_ADJ_AMOUNT;
319    if(pid->x_s2 < -MAX_ADJ_AMOUNT)  pid->x_s2 = -MAX_ADJ_AMOUNT;
320
321    //Y Axis
322    error = pid->y_s1 - gyro_rad->gy;
323    pid_y_integ2 += error*pid->ts;
324    if(pid_y_integ2 > pid->y_i2_limit)
325      pid_y_integ2 = pid->y_i2_limit;
326    else if(pid_y_integ2 < -pid->y_i2_limit)
327      pid_y_integ2 = -pid->y_i2_limit;
328    deriv = (error - pid_y_pre_error2)*dt_recip;
329    pid_y_pre_error2 = error;
330    deriv = pid_y_pre_deriv + (deriv - pid_y_pre_deriv)*D_FILTER_COFF;
331    pid_y_pre_deriv = deriv;
332    pid->y_s2 = pid->y_kp2*error + pid->y_ki2*pid_y_integ2 + pid->y_kd2*deriv;
333
334    if(pid->y_s2 > MAX_ADJ_AMOUNT)  pid->y_s2 = MAX_ADJ_AMOUNT;
335    if(pid->y_s2 < -MAX_ADJ_AMOUNT)  pid->y_s2 = -MAX_ADJ_AMOUNT;
336  #endif  /* TYP_CTRL */
337  /* end: delete */
338
339    //Z Axis
340    error = pid->z_s1 - gyro_rad->gz;
341    pid_z_integ2 += error*pid->ts;
```

元の制御則…PID制御による角速度制御（～336行目）

```
  if(pid_z_integ2 > pid->z_i2_limit)
    pid_z_integ2 = pid->z_i2_limit;
  else if(pid_z_integ2 < -pid->z_i2_limit)
    pid_z_integ2 = -pid->z_i2_limit;
  deriv = (error - pid_z_pre_error2)*dt_recip;
  pid_z_pre_error2 = error;
  pid->z_s2 = pid->z_kp2*error + pid->z_ki2*pid_z_integ2 + pid->z_kd2*deriv;

  if(pid->z_s2 > MAX_ADJ_AMOUNT_YAW)  pid->z_s2 = MAX_ADJ_AMOUNT_YAW;
  if(pid->z_s2 < -MAX_ADJ_AMOUNT_YAW)  pid->z_s2 = -MAX_ADJ_AMOUNT_YAW;


#ifdef MOTOR_DC

  motor_thr = 0.33333f*gTHR + 633.333f;           //Remocon Devo7E >> 630 to 1700

#endif

#ifdef MOTOR_ESC

  //motor_thr = 0.28f*gTHR + 750.0f;                   //TGY-i6 remocon and external ESC STEVAL-ESC001V1
  //motor_thr = 0.28f*gTHR + 850.0f;                   //TGY-i6 remocon and external ESC Afro12A
  motor_thr = 0.32f*gTHR + 900.0f;                     //TGY-i6 remocon and external ESC Afro12A


#endif

  motor_pwm->motor1_pwm = motor_thr - pid->x_s2 - pid->y_s2 + pid->z_s2 + MOTOR_OFF1;
  motor_pwm->motor2_pwm = motor_thr + pid->x_s2 - pid->y_s2 - pid->z_s2 + MOTOR_OFF2;
  motor_pwm->motor3_pwm = motor_thr + pid->x_s2 + pid->y_s2 + pid->z_s2 + MOTOR_OFF3;
  motor_pwm->motor4_pwm = motor_thr - pid->x_s2 + pid->y_s2 - pid->z_s2 + MOTOR_OFF4;

}

void PIDOuterLoopFrameTrans(P_PI_PIDControlTypeDef *pid, EulerAngleTypeDef *euler_ahrs)
{
  float cosx;

  cosx = cos(euler_ahrs->thx);
  pid->y_s1 = cosx*pid->y_s1;

}

```