从开始做四轴到现在，已经累计使用了三个月的时间，从开始的尝试用四元数法进行姿态检测，到接着使用的卡尔曼滤波算法，我们走过了很多弯路，我在从上周开始了对德国人四轴代码的研究和移植，发现德国人的代码的确有他的独到之处，改变了很多我对模型的想法，因为本人是第一次尝试着制作模型，因此感觉很多想法还是比较简单。经过了一周的时间，我将德国人的代码翻译并移植到了我目前的四轴上，并进行了调试，今天，专门请到了一个飞直升机的教练，对我们的四轴进行试飞，并与一个华科尔的四轴进行了现场比较，现在我们四轴的稳定性已经达到了商品四轴的程度。下面是我这一周时间内对德国人代码的一些理解：

德国人代码中的姿态检测算法：
首先，将陀螺仪和加速度及的测量值减常值误差，得到角速度和加速度，并对角速度进行积分，然后对陀螺仪积分和加速度计的数值进行融合。融合分为两部分，实时融合和长期融合，实时融合每一次算法周期都要执行，而长期融合没 256 个检测周期执行一次，（注意检测周期小于控制周期的 2ms）
实时融合：
1.将陀螺仪积分和加表滤波后的值做差；
2.按照情况对差值进行衰减，并作限幅处理；
3.将衰减值加入到角度中。
长期融合：
长期融合主要包括两个部分，一是对角速度的漂移进行估计（估计值是要在每一个控制周期都耦合到角度中的），二是对陀螺仪的常值误差（也就是陀螺仪中立点）进行实时的修正。
1.将陀螺仪积分的积分和加速度积分做差（PS：为什么这里要使用加表积分和陀螺仪积分的积分，因为在 256 个检测周期内，有一些加速度计的值含有有害的加速度分量，如果只使用一个时刻的加表值对陀螺仪漂移进行估计，显然估计值不会准确，使用多个周期的值进行叠加后做座平均处理，可以减小随机的有害加速度对估计陀螺仪漂移过程中所锁产生的影响）
2.将上面两个值进行衰减，得到估计的陀螺仪漂移
3.对使考虑了陀螺仪漂移和不考虑陀螺仪漂移得到的角度做差，如果这两个值较大，说明陀螺仪在前段时间内测到的角速率不够准确，需要对差值误差（也就是陀螺仪中立点）进行修正，修正幅度和差值有关
长期融合十分关键，如果不能对陀螺仪漂移做修正，则系统运行一段时间后，速率环的稳定性会降低。

下面比较一下德国四轴中姿态检测部分和卡尔曼滤波之间的关系：
卡尔曼滤波是一种线性系统的最优估计滤波方法。对于本系统而言，使用卡尔曼滤波的作用是通过对系统状态量的估计，和通过加速度计测量值对系统状态进行验证，从而得到该系统的最优状态量，并实时更新系统的各参数（矩阵），而最重要的一点，改滤波器能够对陀螺仪的常值漂移进行估计，从而保证速率环的正常运行，并在加速度计敏感到各种有害加速度的时候，使姿态检测更加准确。但是卡尔曼滤波器能否工作在最优状态很大程度上取决于系统模型的准确性，模型参数的标定和系统参数的选取。然而，仅仅通过上位机观测而得到最优工作参数是不显示的，因为参数的修改会导致整个系统中很多地方发生改变，很难保证几个值都恰好为最优解，这需要扎实的理论知识，大量的测量数据和系统的仿真，通过我对卡尔曼滤波器的使用，发现要想兼顾锁有的问题，还是有一定难度的。

而德国人的姿态检测部分是在尝试使用一种简单方法去解决复杂问题，他既没有使用传统的

四元数法进行姿态检测，也么有使用卡尔曼滤波。他的计算量不比最简单的卡尔曼滤程序波程序的计算量小，但与卡尔曼滤波相比，更加直观，易于理解，参数调节也更加方便。我个人理解，这个方法是在尝试着对卡尔曼滤波这一复杂相互耦合的多状态变量的线性系统状态估计过程进行了简单的解耦，从而将姿态的最优估计和陀螺仪漂移的最优估计分隔开，这样，就可以通过比较直观的观测手段对两个部分的参数进行调整，但是，这种方法的理论性肯定不如使用四元数法和卡尔曼滤波，在一些特殊的情况下还可能出现问题，但是，由于卡尔曼滤波器设计的难度，使用这种方法还是比较现实的。

控制算法：
德国人的控制算法的核心是对角速度做 PI 计算，P 的作用是使四轴能够产生对于外界干扰的抵抗力矩，I 的作用是让四轴产生一个与角度成正比的抵抗力。
如果只有 P 的作用，将四轴拿在手上就会发现，四轴能够抵抗外界的干扰力矩的作用，而且这个抵抗力非常快速，只要手妄图改变四轴的转速，四轴就会产生一个抵抗力矩，但是，如果用手将四轴扳过一个角度，则四轴无法自己回到水平的角度位置，这就需要 I 的调节作用。

对角速度做 I（积分）预算实际得到的就是角度，德国人四轴里面用的也是角度值，如果四轴有一个倾斜角度，那么四轴就会自己进行调整，直到四轴的倾角为零，它所产生的抵抗力是与角度成正比的，但是，如果只有 I 的作用，会使四轴迅速产生振荡，因此，必须将 P 和 I 结合起来一起使用，这时候基本上就会得到德国四轴的效果了。

在对角速度进行了 PI 调节之后，德国人将操纵杆的值融合到结果中去，并对得到的新的值有进行了一次 PI 计算，这个积分参数很小，使用这个积分的作用因为，四轴在有一个非常小的倾角的情况下产生的抵抗力矩很小，无法使四轴回到水平位置，这就会导致无论怎么手动调节微调，四轴都很难做到悬停，会不停得做水平漂移运动，这就必须不停的进行调整。

下面是我给德国四轴中飞控程序的一些注释：

```
void Piep(unsigned char Anzahl)
{
 while(Anzahl--)
 {
  if(MotorenEin) return; //auf keinen Fall im Flug!
  beeptime = 100;
  Delay_ms(250);
 }
}
```

//函数:SetNeutral 设定传感器发出参数的中立数值，因为有漂移所以要使其每次工作都要测量出来。

```
//###########################################################################
//  Nullwerte ermitteln
/*设置中立点*/
```

```
void SetNeutral(void)
//###########################################################################################
{
/*加速度计中立点*/
NeutralAccX = 0;
NeutralAccY = 0;
NeutralAccZ = 0;
/*陀螺仪中立点*/
AdNeutralNick = 0;
AdNeutralRoll = 0;
AdNeutralGier = 0;
   Parameter_AchsKopplung1 = 0;
   Parameter_AchsGegenKopplung1 = 0;。。。

/*这个地方我还没有弄得太明白，检测中立点的函数被调用了两次，但是第一次的数据好像
没有保存，只用到了

第二次的数据*/
/*记录中立点*/
CalibrierMittelwert();
   Delay_ms_Mess(100);
/*记录中立点*/
CalibrierMittelwert();
/*既然只使用了后一次的数据，为什么要进行两次记录中立点的函数*/
   if((EE_Parameter.GlobalConfig & CFG_HOEHENREGELUNG))
   {
    if((MessLuftdruck > 950) || (MessLuftdruck < 750)) SucheLuftruckOffset();//如果气压表输出
在

950 外 750 内，则设定气压初始的偏差。
   }

/*将量测值作为陀螺仪的中立点*/
AdNeutralNick= AdWertNick;
AdNeutralRoll= AdWertRoll;
AdNeutralGier= AdWertGier;

/*这两个参数在飞控程序中没有用到*/
StartNeutralRoll = AdNeutralRoll;
StartNeutralNick = AdNeutralNick;

   if(eeprom_read_byte(&EEPromArray[EEPROM_ADR_ACC_NICK]) > 4) //
   {
/*由于在函数 CalibrierMittelwert()中加速度计的输出乘以了 ACC_AMPLIFY，所以这里必须处
```

以 ACC_AMPLIFY，

在这段程序中，所有的对加速度计和陀螺仪的数值的衰减或者放大都是为了让
陀螺仪积分和加速度计数值在同样的角度偏差的情况下能基本匹配，如果不匹配，那么就谈不上用加速度计来补

偿陀螺仪积分了*/
```
    NeutralAccY = abs(Mittelwert_AccRoll) / ACC_AMPLIFY;
 NeutralAccX = abs(Mittelwert_AccNick) / ACC_AMPLIFY;
 NeutralAccZ = Aktuell_az;
  }
  else
  {
```
/*如果发现变化不大，则仍然储存上一次的*/
```
    NeutralAccX = (int)eeprom_read_byte(&EEPromArray[EEPROM_ADR_ACC_NICK]) * 256 + (int)

eeprom_read_byte(&EEPromArray[EEPROM_ADR_ACC_NICK+1]);
 NeutralAccY = (int)eeprom_read_byte(&EEPromArray[EEPROM_ADR_ACC_ROLL]) * 256 + (int)

eeprom_read_byte(&EEPromArray[EEPROM_ADR_ACC_ROLL+1]);
 NeutralAccZ = (int)eeprom_read_byte(&EEPromArray[EEPROM_ADR_ACC_Z]) * 256 + (int)

eeprom_read_byte(&EEPromArray[EEPROM_ADR_ACC_Z+1]);
  }
```
/*将所有数据清零，这里带 2 的变量都是加入了陀螺仪漂移补偿值之后得到的角度*/
```
Mess_IntegralNick = 0;
  Mess_IntegralNick2 = 0;
  Mess_IntegralRoll = 0;
  Mess_IntegralRoll2 = 0;
  Mess_Integral_Gier = 0;
  MesswertNick = 0;
  MesswertRoll = 0;
  MesswertGier = 0;
  StartLuftdruck = Luftdruck;
  HoeheD = 0;
  Mess_Integral_Hoch = 0;
  KompassStartwert = KompassValue;
  GPS_Neutral();
  beeptime = 50;  //
```
/*从 EEPROM 中读取陀螺仪积分到达 90°时候的值，并储存，当得到的姿态角度大于这个范围时，说明超过了 90°

，就要进行相应的处理*/

```
Umschlag180Nick = (long) EE_Parameter.WinkelUmschlagNick * 2500L;
Umschlag180Roll = (long) EE_Parameter.WinkelUmschlagRoll * 2500L;
  ExternHoehenValue = 0;
}

////////////////////////////////
//函数描述：求参数的平均数值
////////////////////////////////

//#############################################################################################
// Bearbeitet die Messwerte
void Mittelwert(void)
// 根据测量值 计算陀螺仪和加速度计数据
//#############################################################################################
{
  static signed long tmpl,tmpl2;
/*将陀螺仪数据减去常值误差，得到实际的叫速率的倍数*/
  MesswertGier = (signed int) AdNeutralGier - AdWertGier;
  MesswertRoll = (signed int) AdWertRoll - AdNeutralRoll;
  MesswertNick = (signed int) AdWertNick - AdNeutralNick;

//DebugOut.Analog[26] = MesswertNick;
DebugOut.Analog[28] = MesswertRoll;
//加速度传感器输出
/*加速度计数据滤波，ACC_AMPLIFY=12 得到的 Mittelwert_AccNick 是加速度计数值的 12 倍
*/
/*AdWertAccNick 为测量值*/
// Beschleunigungssensor  ++++++++++++++++++++++++++++++++++++++++++++++++++
Mittelwert_AccNick = ((long)Mittelwert_AccNick * 1 + ((ACC_AMPLIFY * (long)AdWertAccNick))) /


2L;//具有滤波功能的方法，用当前加速度和上次的加速度平均
Mittelwert_AccRoll = ((long)Mittelwert_AccRoll * 1 + ((ACC_AMPLIFY * (long)AdWertAccRoll))) / 2
L;
Mittelwert_AccHoch = ((long)Mittelwert_AccHoch * 1 + ((long)AdWertAccHoch)) / 2L;

/*计算加速度计的积分，加速度计对运动十分敏感，采用加速度计积分，可以减少瞬间的运
动加速度的影响*/
  IntegralAccNick += ACC_AMPLIFY * AdWertAccNick;
  IntegralAccRoll += ACC_AMPLIFY * AdWertAccRoll;
  IntegralAccZ   += Aktuell_az - NeutralAccZ;
// Gier  ++++++++++++++++++++++++++++++++++++++++++++++++++
/*偏航方向无法进行滤波，因此直接进行积分得到偏航角度*/
      Mess_Integral_Gier +=  MesswertGier;
```

```
        Mess_Integral_Gier2 += MesswertGier;
/*耦合项应该是另外两个陀螺仪对这个轴上陀螺仪的影响，当四轴在稳定姿态不为水平的时候，进行偏航运动时

候所进行的补偿*/
/*假设目前的俯仰角是 30°，而横滚角是 0°，这时候如保持俯仰和横滚轴没有任何运动，而将偏航轴转动 90°

，那么实际的俯仰角就会变为 0°，横滚角就会变为 30°
 但是，按照目前的算法，由于俯仰和横滚方向没有运动，因此就不会有陀螺仪的积分，俯仰和横滚角是不变的

，这就是采用陀螺仪直接积分测角度的不完善性，这时候
 使用加速度计对姿态进行修正能够起到作用，但是需要一段时间，使用下面的这段话，就是将偏航轴的运动耦

合在另外两个轴上，使姿态角度能够迅速收敛到真实值上*/
/*注：使用四元数法进行姿态结算可以避免出现这种问题，但这种方法需要有准确的陀螺仪和加表的数学模型，

四元数法还需要进行大量的矩阵计算，
 而且对四元数姿态进行加速度计的融合不太方便*/
    if(!Looping_Nick && !Looping_Roll && (EE_Parameter.GlobalConfig &

CFG_ACHSENKOPPLUNG_AKTIV))//不在俯仰滚转控制循环中
      {
        tmpl = Mess_IntegralNick / 4096L;
        tmpl *= MesswertGier;
        tmpl *= Parameter_AchsKopplung1;  //125
        tmpl /= 2048L;
        tmpl2 = Mess_IntegralRoll / 4096L;
        tmpl2 *= MesswertGier;
        tmpl2 *= Parameter_AchsKopplung1;
        tmpl2 /= 2048L;
      }
    else  tmpl = tmpl2 = 0;
// Roll  ++++++++++++++++++++++++++++++++++++++++++++++++
        MesswertRoll += tmpl;
        MesswertRoll += (tmpl2*Parameter_AchsGegenKopplung1)/512L;
        Mess_IntegralRoll2 += MesswertRoll;
        Mess_IntegralRoll +=  MesswertRoll - LageKorrekturRoll;
/*积分超过半圈的情况*/
        if(Mess_IntegralRoll > Umschlag180Roll)
        {
```

```
        Mess_IntegralRoll  = -(Umschlag180Roll - 10000L);
        Mess_IntegralRoll2 = Mess_IntegralRoll;
       }
       if(Mess_IntegralRoll <-Umschlag180Roll)//一样
       {
        Mess_IntegralRoll =  (Umschlag180Roll - 10000L);
        Mess_IntegralRoll2 = Mess_IntegralRoll;
       }
       if(AdWertRoll < 15)   MesswertRoll = -1000;
       if(AdWertRoll <  7)   MesswertRoll = -2000;
       if(PlatinenVersion == 10)
{
        if(AdWertRoll > 1010) MesswertRoll = +1000;
        if(AdWertRoll > 1017) MesswertRoll = +2000;
}
else
{
        if(AdWertRoll > 2020) MesswertRoll = +1000;
        if(AdWertRoll > 2034) MesswertRoll = +2000;
}
// Nick  ++++++++++++++++++++++++++++++++++++++++++++++++
       MesswertNick -= tmpl2;
       MesswertNick -= (tmpl*Parameter_AchsGegenKopplung1)/512L;
       Mess_IntegralNick2 += MesswertNick;
/*LageKorrekturNick 是通过加速度计积分和角速率积分的积分进行做差比较得到

的,*/
       Mess_IntegralNick  += MesswertNick - LageKorrekturNick;
       if(Mess_IntegralNick > Umschlag180Nick)
       {
        Mess_IntegralNick = -(Umschlag180Nick - 10000L);
        Mess_IntegralNick2 = Mess_IntegralNick;
       }
       if(Mess_IntegralNick <-Umschlag180Nick)
       {
        Mess_IntegralNick =  (Umschlag180Nick - 10000L);
        Mess_IntegralNick2 = Mess_IntegralNick;
       }
       if(AdWertNick < 15)   MesswertNick = -1000;
       if(AdWertNick <  7)   MesswertNick = -2000;
       if(PlatinenVersion == 10)
{
        if(AdWertNick > 1010) MesswertNick = +1000;
        if(AdWertNick > 1017) MesswertNick = +2000;
```

```
 }
 else
 {
        if(AdWertNick > 2020) MesswertNick = +1000;
        if(AdWertNick > 2034) MesswertNick = +2000;
 }
//++++++++++++++++++++++++++++++++++++++++++++++++++
// ADC einschalten
   ANALOG_ON; //重新开始模拟量的采集
//++++++++++++++++++++++++++++++++++++++++++++++++++

/*上一步计算完了积分之后，现在将积分赋值，因此后面使用的就将是 IntegralNick，
IntegralNick2 等数据了

*/
   Integral_Gier  = Mess_Integral_Gier;
   IntegralNick = Mess_IntegralNick;
   IntegralRoll = Mess_IntegralRoll;
   IntegralNick2 = Mess_IntegralNick2;
   IntegralRoll2 = Mess_IntegralRoll2;

/*这部分代码不执行，因为在 EEPROM 中 CFG_DREHRATEN_BEGRENZER 这一位为 0*/
  if(EE_Parameter.GlobalConfig & CFG_DREHRATEN_BEGRENZER && !Looping_Nick && !Looping_
Roll)
  {
   if(MesswertNick > 200)      MesswertNick += 4 * (MesswertNick - 200);
   else if(MesswertNick < -200) MesswertNick += 4 * (MesswertNick + 200);
   if(MesswertRoll > 200)      MesswertRoll += 4 * (MesswertRoll - 200);
   else if(MesswertRoll < -200) MesswertRoll += 4 * (MesswertRoll + 200);
  }
   if(Poti1 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI1]] + 110) Poti1++;
else if(Poti1 > PPM_in[EE_Parameter.Kanalbelegung[K_POTI1]] + 110 && Poti1) Poti1--;
   if(Poti2 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI2]] + 110) Poti2++; else if(Poti2 > PPM_i
n

[EE_Parameter.Kanalbelegung[K_POTI2]] + 110 && Poti2) Poti2--;
   if(Poti3 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI3]] + 110) Poti3++; else if(Poti3 > PPM_i
n

[EE_Parameter.Kanalbelegung[K_POTI3]] + 110 && Poti3) Poti3--;
   if(Poti4 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI4]] + 110) Poti4++; else if(Poti4 > PPM_i
n

[EE_Parameter.Kanalbelegung[K_POTI4]] + 110 && Poti4) Poti4--;
```

```
  if(Poti1 < 0) Poti1 = 0; else if(Poti1 > 255) Poti1 = 255;
  if(Poti2 < 0) Poti2 = 0; else if(Poti2 > 255) Poti2 = 255;
  if(Poti3 < 0) Poti3 = 0; else if(Poti3 > 255) Poti3 = 255;
  if(Poti4 < 0) Poti4 = 0; else if(Poti4 > 255) Poti4 = 255;
}


//函数：校正平均值

//############################################################################
// Messwerte beim Ermitteln der Nullage
/*确定零位*/
/*记录中立点*/
void CalibrierMittelwert(void)
//############################################################################
{
  // ADC auschalten, damit die Werte sich nicht w 鋒 rend der Berechnung 鋘 dern
ANALOG_OFF;
MesswertNick = AdWertNick;
MesswertRoll = AdWertRoll;
MesswertGier = AdWertGier;
/*要乘以 ACC_AMPLIFY 是为了进行数值的匹配*/
Mittelwert_AccNick = ACC_AMPLIFY * (long)AdWertAccNick;
Mittelwert_AccRoll = ACC_AMPLIFY * (long)AdWertAccRoll;
Mittelwert_AccHoch = (long)AdWertAccHoch;
  // ADC einschalten
   ANALOG_ON; //开模拟量
   if(Poti1 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI1]] + 110) Poti1++;
else if(Poti1 > PPM_in[EE_Parameter.Kanalbelegung[K_POTI1]] + 110 && Poti1) Poti1--;

if(Poti2 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI2]] + 110) Poti2++;
else if(Poti2 > PPM_in[EE_Parameter.Kanalbelegung[K_POTI2]] + 110 && Poti2) Poti2--;

if(Poti3 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI3]] + 110) Poti3++;
else if(Poti3 > PPM_in[EE_Parameter.Kanalbelegung[K_POTI3]] + 110 && Poti3) Poti3--;

   if(Poti4 < PPM_in[EE_Parameter.Kanalbelegung[K_POTI4]] + 110) Poti4++;
else if(Poti4 > PPM_in[EE_Parameter.Kanalbelegung[K_POTI4]] + 110 && Poti4) Poti4--;

   if(Poti1 < 0) Poti1 = 0;
else if(Poti1 > 255) Poti1 = 255;

   if(Poti2 < 0) Poti2 = 0;
else if(Poti2 > 255) Poti2 = 255;
```

```
  if(Poti3 < 0) Poti3 = 0;
else if(Poti3 > 255) Poti3 = 255;

  if(Poti4 < 0) Poti4 = 0;
else if(Poti4 > 255) Poti4 = 255;

/*这两个数据是在对陀螺仪积分区域进行的限制，如果超过这个范围，说明就超出了+-90°
的范围，则需要相应

的改变*/
Umschlag180Nick = (long) EE_Parameter.WinkelUmschlagNick * 2500L;
Umschlag180Roll = (long) EE_Parameter.WinkelUmschlagNick * 2500L;
}

//发送电机数据
//############################################################################
// Senden der Motorwerte per I2C-Bus
void SendMotorData(void)
//############################################################################
{
  if(MOTOR_OFF || !MotorenEin)//关机或未工作
    {
    Motor_Hinten = 0;
    Motor_Vorne = 0;
    Motor_Rechts = 0;
    Motor_Links = 0;//都置零
    if(MotorTest[0]) Motor_Vorne = MotorTest[0];
    if(MotorTest[1]) Motor_Hinten = MotorTest[1];
    if(MotorTest[2]) Motor_Links = MotorTest[2];
    if(MotorTest[3]) Motor_Rechts = MotorTest[3];//如果是试验就干。
    }

  DebugOut.Analog[12] = Motor_Vorne;
  DebugOut.Analog[13] = Motor_Hinten;
  DebugOut.Analog[14] = Motor_Links;
  DebugOut.Analog[15] = Motor_Rechts;

  //Start I2C Interrupt Mode
  twi_state = 0;
  motor = 0;
  i2c_start();
}
```

```
//函数：参数分配


//#############################################################################
// Tr 鋑 t ggf. das Poti als Parameter ein
void ParameterZuordnung(void)
//#############################################################################
{
//
/*这个宏定义的作用是：将 a 中的值赋给 b，并将 b 限制在 max 和 min 之间*/
 #define CHK_POTI(b,a,min,max) { if(a > 250) { if(a == 251) b = Poti1; else if(a == 252) b =

Poti2; else if(a == 253) b = Poti3; else if(a == 254) b = Poti4;} else b = a; if(b <= min) b =

min; else if(b >= max) b = max;}
 CHK_POTI(Parameter_MaxHoehe,EE_Parameter.MaxHoehe,0,255);
 CHK_POTI(Parameter_Luftdruck_D,EE_Parameter.Luftdruck_D,0,100);
 CHK_POTI(Parameter_Hoehe_P,EE_Parameter.Hoehe_P,0,100);
 CHK_POTI(Parameter_Hoehe_ACC_Wirkung,EE_Parameter.Hoehe_ACC_Wirkung,0,255);
 CHK_POTI(Parameter_KompassWirkung,EE_Parameter.KompassWirkung,0,255);
 CHK_POTI(Parameter_Gyro_P,EE_Parameter.Gyro_P,10,255);
 CHK_POTI(Parameter_Gyro_I,EE_Parameter.Gyro_I,0,255);
 CHK_POTI(Parameter_I_Faktor,EE_Parameter.I_Faktor,0,255);
 CHK_POTI(Parameter_UserParam1,EE_Parameter.UserParam1,0,255);
 CHK_POTI(Parameter_UserParam2,EE_Parameter.UserParam2,0,255);
 CHK_POTI(Parameter_UserParam3,EE_Parameter.UserParam3,0,255);
 CHK_POTI(Parameter_UserParam4,EE_Parameter.UserParam4,0,255);
 CHK_POTI(Parameter_UserParam5,EE_Parameter.UserParam5,0,255);
 CHK_POTI(Parameter_UserParam6,EE_Parameter.UserParam6,0,255);
 CHK_POTI(Parameter_UserParam7,EE_Parameter.UserParam7,0,255);
 CHK_POTI(Parameter_UserParam8,EE_Parameter.UserParam8,0,255);
 CHK_POTI(Parameter_ServoNickControl,EE_Parameter.ServoNickControl,0,255);
 CHK_POTI(Parameter_LoopGasLimit,EE_Parameter.LoopGasLimit,0,255);
 CHK_POTI(Parameter_AchsKopplung1,   EE_Parameter.AchsKopplung1,0,255);
 CHK_POTI(Parameter_AchsGegenKopplung1,EE_Parameter.AchsGegenKopplung1,0,255);
 CHK_POTI(Parameter_DynamicStability,EE_Parameter.DynamicStability,0,255);

 Ki = (float) Parameter_I_Faktor * 0.0001;
 MAX_GAS = EE_Parameter.Gas_Max;
 MIN_GAS = EE_Parameter.Gas_Min;
}

/*飞控核心*/
//#############################################################################
```

```
//
void MotorRegler(void)
//############################################################################################
{
 int motorwert,pd_ergebnis,h,tmp_int;//电机数值，PI 算法的计算数值
 int GierMischanteil,GasMischanteil;//偏航混合数值，油门混和数值
    static long SummeNick=0,SummeRoll=0;//俯仰积分总和，滚转积分总和
    static long sollGier = 0,tmp_long,tmp_long2;//标准偏航值，
    static long IntegralFehlerNick = 0;//俯仰误差积分
    static long IntegralFehlerRoll = 0;//滚转误差积分
   static unsigned int RcLostTimer;
   static unsigned char delay_neutral = 0;
   static unsigned char delay_einschalten = 0,delay_ausschalten = 0;//延迟接通，延迟关闭
   static unsigned int  modell_fliegt = 0;//飞机飞行时间
    static int hoehenregler = 0;//高度调节
    static char TimerWerteausgabe = 0;//时间数值
    static char NeueKompassRichtungMerken = 0;//罗盘方向调整中立值
    static long ausgleichNick, ausgleichRoll;//俯仰均衡，滚转均衡

/*根据测量值 计算陀螺仪和加速度计数据*/
Mittelwert();

   GRN_ON;//打开端口
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// Gaswert ermitteln//判断油门数值
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
   GasMischanteil = StickGas;
if(GasMischanteil < 0) GasMischanteil = 0;

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// Emfang schlecht//无线电故障，不好
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
if(SenderOkay < 100)
{
if(!PcZugriff)
{
if(BeepMuster == 0xffff)
        {
         beeptime = 15000;
         BeepMuster = 0x0c00;
```

```
        }
}
      if(RcLostTimer) RcLostTimer--;
      else
{
MotorenEin = 0;
Notlandung = 0;
}
      ROT_ON;
      if(modell_fliegt > 2000)
      {
GasMischanteil = EE_Parameter.NotGas;
        Notlandung = 1;
        PPM_in[EE_Parameter.Kanalbelegung[K_NICK]] = 0;
        PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] = 0;
        PPM_in[EE_Parameter.Kanalbelegung[K_GIER]] = 0;
}
      else
MotorenEin = 0;
} // end of if(SenderOkay < 100)
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// Emfang gut//
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
else if(SenderOkay > 140)
{
Notlandung = 0;
RcLostTimer = EE_Parameter.NotGasZeit * 50;
if(GasMischanteil > 40)
{
if(modell_fliegt < 0xffff)
modell_fliegt++;
}
if((modell_fliegt < 200) || (GasMischanteil < 40))
{
SummeNick = 0;
SummeRoll = 0;
Mess_Integral_Gier = 0;
Mess_Integral_Gier2 = 0;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// auf Nullwerte kalibrieren
```

```
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
if((PPM_in[EE_Parameter.Kanalbelegung[K_GAS]] > 80) && MotorenEin == 0)
{
if(PPM_in[EE_Parameter.Kanalbelegung[K_GIER]] > 75)
{
if(++delay_neutral > 200)
{
GRN_OFF;
MotorenEin = 0;
            delay_neutral = 0;
            modell_fliegt = 0;
            if(PPM_in[EE_Parameter.Kanalbelegung[K_NICK]] > 70 || abs(PPM_in

[EE_Parameter.Kanalbelegung[K_ROLL]]) > 70)
            {
unsigned char setting=1;
              if(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] > 70 && PPM_in

[EE_Parameter.Kanalbelegung[K_NICK]] < 70) setting = 1;
              if(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] > 70 && PPM_in

[EE_Parameter.Kanalbelegung[K_NICK]] > 70) setting = 2;
              if(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] < 70 && PPM_in

[EE_Parameter.Kanalbelegung[K_NICK]] > 70) setting = 3;
              if(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] <-70 && PPM_in

[EE_Parameter.Kanalbelegung[K_NICK]] > 70) setting = 4;
              if(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] <-70 && PPM_in

[EE_Parameter.Kanalbelegung[K_NICK]] < 70) setting = 5;
              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACTIVE_SET], setting);
            }
            if((EE_Parameter.GlobalConfig & CFG_HOEHENREGELUNG))  // H 鰄 enregelung

aktiviert?
            {
if((MessLuftdruck > 950) || (MessLuftdruck < 750))
SucheLuftruckOffset();
            }
            ReadParameterSet(GetActiveParamSetNumber(), (unsigned char *)

&EE_Parameter.Kanalbelegung[0], STRUCT_PARAM_LAENGE);
```

```
SetNeutral();
              Piep(GetActiveParamSetNumber());
            }
}
else if(PPM_in[EE_Parameter.Kanalbelegung[K_GIER]] < -75)
        {
if(++delay_neutral > 200)  // nicht sofort
            {
GRN_OFF;
              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACC_NICK],0xff); // Werte l        鰞
chen
              MotorenEin = 0;
              delay_neutral = 0;
              modell_fliegt = 0;
              SetNeutral();//设立中性点。
              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACC_NICK],NeutralAccX / 256); //

ACC-NeutralWerte speichern
              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACC_NICK+1],NeutralAccX % 256);
 //

ACC-NeutralWerte speichern
              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACC_ROLL],NeutralAccY / 256);
              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACC_ROLL+1],NeutralAccY % 256);

              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACC_Z],(int)NeutralAccZ / 256);
              eeprom_write_byte(&EEPromArray[EEPROM_ADR_ACC_Z+1],(int)NeutralAccZ % 256);

              Piep(GetActiveParamSetNumber());
            }
        }
        else
delay_neutral = 0;
} // end if of if(PPM_in[EE_Parameter.Kanalbelegung[K_GIER]] > 75)
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
// Gas ist unten
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
        if(PPM_in[EE_Parameter.Kanalbelegung[K_GAS]] < 35-120)
        {
// Starten
          if(PPM_in[EE_Parameter.Kanalbelegung[K_GIER]] < -75)
          {
```

```
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Einschalten
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
            if(++delay_einschalten > 200)
{
               delay_einschalten = 200;
               modell_fliegt = 1;
               MotorenEin = 1;
               sollGier = 0;
               Mess_Integral_Gier = 0;
               Mess_Integral_Gier2 = 0;
               Mess_IntegralNick = 0;
               Mess_IntegralRoll = 0;
               Mess_IntegralNick2 = IntegralNick;
               Mess_IntegralRoll2 = IntegralRoll;
               SummeNick = 0;
               SummeRoll = 0;
            }
         }
         else
delay_einschalten = 0;//没事，就让其延迟关闭为 0
         //Auf Neutralwerte setzen
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Auschalten
/*切换*/
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
         if(PPM_in[EE_Parameter.Kanalbelegung[K_GIER]] > 75)
         {
if(++delay_ausschalten > 200)  // nicht sofort
            {
            MotorenEin = 0;
            delay_ausschalten = 200;
            modell_fliegt = 0;
            }
         }
         else delay_ausschalten = 0;
      }
   } // end if of else if(SenderOkay > 140)

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
++++++++++++++++++
// neue Werte von der Funke
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
 if(!NewPpmData-- || Notlandung)
  {
    int tmp_int;
static int stick_nick,stick_roll;//俯仰杆，倾斜杆

    ParameterZuordnung();

/*新老数据滤波混合，这里改变的应该是期望角位置，必须知道 EE_Parameter.Stick_P 的数值才可以得到滤波效

果*/
    StickNick = (StickNick * 3 + PPM_in[EE_Parameter.Kanalbelegung[K_NICK]] *

EE_Parameter.Stick_P) / 4; //新数据和老数据混合起滤波作用

/* 将期望角位置的微分加入操纵杆变量上，这里必须知道 EE_Parameter.Kanalbelegung[K_ROLL]的求法，和

EE_Parameter.Stick_D 得数值*/
    StickNick += PPM_diff[EE_Parameter.Kanalbelegung[K_NICK]] * EE_Parameter.Stick_D;// 增加上微分

量，用于提高反应的快速性。

    StickRoll = (StickRoll * 3 + PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] *

EE_Parameter.Stick_P) / 4;
    StickRoll += PPM_diff[EE_Parameter.Kanalbelegung[K_ROLL]] * EE_Parameter.Stick_D;

    StickGier = -PPM_in[EE_Parameter.Kanalbelegung[K_GIER]];
    StickGas  = PPM_in[EE_Parameter.Kanalbelegung[K_GAS]] + 120;

/*用此记录历史上的最大给杆量，如果给杆量很小，则 Max 数值会不断减小，用于在后面给陀螺仪积分做补偿时

，对加速度计数据和陀螺仪积分的差值做衰减*/
   if(abs(PPM_in[EE_Parameter.Kanalbelegung[K_NICK]]) > MaxStickNick)
     MaxStickNick = abs(PPM_in[EE_Parameter.Kanalbelegung[K_NICK]]); else MaxStickNick--;
   if(abs(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]]) > MaxStickRoll)
     MaxStickRoll = abs(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]]); else MaxStickRoll--;
```

/*如果在降落过程中，则数据为 0，也就是说降落的过程中不需要衰减，降落时候的保持位置全部为 0，所以不需

要衰减*/
```
  if(Notlandung)
  {
  MaxStickNick = 0; MaxStickRoll = 0;
  }
```

/*可以认为是控制参数，前一个是陀螺仪的比例项（速率环参数） 后一个是陀螺仪积分即姿态角的比例（位置

环参数）*/
```
   GyroFaktor     = ((float) Parameter_Gyro_P + 10.0) / 256.0;
   IntegralFaktor = ((float) Parameter_Gyro_I) / 44000;

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Digitale Steuerung per DubWise
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#define KEY_VALUE (Parameter_UserParam1 * 4)  //(Poti3 * 8)//为了增加杆的输入的丰富性，提供了扩展
```

的杆的描述，对最终杆的描述更加丰富。
```
if(DubWiseKeys[1])
beeptime = 10;
if(DubWiseKeys[1] & DUB_KEY_UP)
tmp_int = KEY_VALUE;
else if(DubWiseKeys[1] & DUB_KEY_DOWN)
tmp_int = -KEY_VALUE;
else
tmp_int = 0;

ExternStickNick = (ExternStickNick * 7 + tmp_int) / 8;

if(DubWiseKeys[1] & DUB_KEY_LEFT)
tmp_int = KEY_VALUE;
else if(DubWiseKeys[1] & DUB_KEY_RIGHT)
tmp_int = -KEY_VALUE;
else
tmp_int = 0;

ExternStickRoll = (ExternStickRoll * 7 + tmp_int) / 8;
```

```
if(DubWiseKeys[0] & 8)
ExternStickGier = 50;
else if(DubWiseKeys[0] & 4)
ExternStickGier =-50;
else
ExternStickGier = 0;

if(DubWiseKeys[0] & 2)
ExternHoehenValue++;

if(DubWiseKeys[0] & 16)
ExternHoehenValue--;

StickNick += ExternStickNick / 8;
StickRoll += ExternStickRoll / 8;
StickGier += ExternStickGier;
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++
//+ Analoge Steuerung per Seriell
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++
if(ExternControl.Config & 0x01 && Parameter_UserParam1 > 128)//同上，具有扩展功能的控制
输入
{
StickNick += (int) ExternControl.Nick * (int) EE_Parameter.Stick_P;
StickRoll += (int) ExternControl.Roll * (int) EE_Parameter.Stick_P;
StickGier += ExternControl.Gier;
   ExternHoehenValue =  (int) ExternControl.Hight * (int)EE_Parameter.Hoehe_Verstaerkung;
   if(ExternControl.Gas < StickGas) StickGas = ExternControl.Gas;
}

/*陀螺仪积分比例为零，应该是 Looping 的情况？ */
if(EE_Parameter.GlobalConfig & CFG_HEADING_HOLD)
IntegralFaktor =  0;
if(GyroFaktor < 0) GyroFaktor = 0;
if(IntegralFaktor < 0) IntegralFaktor = 0;


// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// Looping?//这里是在空中转圈的情况
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
++++++++++++++++++
 if((PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] > EE_Parameter.LoopThreshold) &&

EE_Parameter.LoopConfig & CFG_LOOP_LINKS)  Looping_Links = 1;
 else
 {
  {
   if((PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] < (EE_Parameter.LoopThreshold -

EE_Parameter.LoopHysterese))) Looping_Links = 0;
  }
 }
 if((PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] < -EE_Parameter.LoopThreshold) &&

EE_Parameter.LoopConfig & CFG_LOOP_RECHTS) Looping_Rechts = 1;
  else
 {
 if(Looping_Rechts) // Hysterese
  {
   if(PPM_in[EE_Parameter.Kanalbelegung[K_ROLL]] > -(EE_Parameter.LoopThreshold -

EE_Parameter.LoopHysterese)) Looping_Rechts = 0;
  }
 }

 if((PPM_in[EE_Parameter.Kanalbelegung[K_NICK]] > EE_Parameter.LoopThreshold) &&

EE_Parameter.LoopConfig & CFG_LOOP_OBEN) Looping_Oben = 1;
 else
 {
  if(Looping_Oben)  // Hysterese
  {
   if((PPM_in[EE_Parameter.Kanalbelegung[K_NICK]] < (EE_Parameter.LoopThreshold -

EE_Parameter.LoopHysterese))) Looping_Oben = 0;
  }
 }
 if((PPM_in[EE_Parameter.Kanalbelegung[K_NICK]] < -EE_Parameter.LoopThreshold) &&

EE_Parameter.LoopConfig & CFG_LOOP_UNTEN) Looping_Unten = 1;
  else
 {
  if(Looping_Unten) // Hysterese
  {
```

```
    if(PPM_in[EE_Parameter.Kanalbelegung[K_NICK]] > -(EE_Parameter.LoopThreshold -

EE_Parameter.LoopHysterese)) Looping_Unten = 0;
    }
  }
/*不应该出现轴都是 Looping 的情况*/
  if(Looping_Links || Looping_Rechts)   Looping_Roll = 1; else Looping_Roll = 0;
  if(Looping_Oben  || Looping_Unten) {Looping_Nick = 1; Looping_Roll = 0; Looping_Links = 0;

Looping_Rechts = 0;} else Looping_Nick = 0;
} // end if of if(!NewPpmData-- || Notlandung)


if(Looping_Roll) beeptime = 100;
if(Looping_Roll || Looping_Nick)
{
if(GasMischanteil > EE_Parameter.LoopGasLimit) GasMischanteil = EE_Parameter.LoopGasLimit;
}

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// Bei Empfangsausfall im Flug
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
if(Notlandung)
{
/*如果出现紧急降落，则将三个期望位置全部置零，即让飞行器向最稳定的方向调整，同时
改变控制参

数，并且不让飞行器处在空中打转的状态*/
StickGier = 0;
StickNick = 0;
StickRoll = 0;
GyroFaktor  = 0.1;
IntegralFaktor = 0.005;
Looping_Roll = 0;
   Looping_Nick = 0;
}


// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// Integrale auf ACC-Signal abgleichen//加速度信号的积分校准
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
++++++++++++++++++
#define ABGLEICH_ANZAHL 256L

/*计算陀螺仪积分的积分，为了和加速度计的积分做比较，进行角速率的补偿和陀螺仪中立点的修正*/
 MittelIntegralNick  += IntegralNick;
 MittelIntegralRoll  += IntegralRoll;
 MittelIntegralNick2 += IntegralNick2;
 MittelIntegralRoll2 += IntegralRoll2;
/*在空中打转过程中，让所有的积分项都为零，因为机动过程会产生很大的误差，因此需要尽快结束其控制，然

后自动调平。 */
 if(Looping_Nick || Looping_Roll)
 {
   IntegralAccNick = 0;
   IntegralAccRoll = 0;
   MittelIntegralNick = 0;
   MittelIntegralRoll = 0;
   MittelIntegralNick2 = 0;
   MittelIntegralRoll2 = 0;
   Mess_IntegralNick2 = Mess_IntegralNick;
   Mess_IntegralRoll2 = Mess_IntegralRoll;
   ZaehlMessungen = 0;
   LageKorrekturNick = 0;
   LageKorrekturRoll = 0;
 }

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
if(!Looping_Nick && !Looping_Roll)
{
long tmp_long, tmp_long2;

/*使用加速度计的值去补偿陀螺仪的积分，这里必须知道 EE_Parameter.GyroAccFaktor 参数，才能够知道补偿了

多少*/
/*其中 IntegralNick 应该是陀螺仪积分
Mittelwert_AccNick = ((long)Mittelwert_AccNick * 1 + ((ACC_AMPLIFY * (long)AdWertAccNick))) /
 2L;

是滤波后的加速度，用当前加速度和上次的加速度平均 */
tmp_long = (long)(IntegralNick / EE_Parameter.GyroAccFaktor - (long)
```

```
Mittelwert_AccNick);//
tmp_long2 = (long)(IntegralRoll / EE_Parameter.GyroAccFaktor - (long)

Mittelwert_AccRoll); //
tmp_long /= 16;
tmp_long2 /= 16;
```

/*如果历史最大摇杆的量比较大，则说明在前段时间内飞行器的姿态可能不为 0，这就导致加速度计的输出受到

有害加速度的影响，因此必须加速度计和陀螺仪积分差值的基础上做一次衰减*/

```
if((MaxStickNick > 15) || (MaxStickRoll > 15))
{
tmp_long  /= 3;
tmp_long2 /= 3;
}
```

/*当偏航轴的操纵杆输入较大时候，则说明这时候偏航轴有一个角速度，为了消除有害加速度的影响，必须对这

两个数值再做一次衰减*/

```
if(abs(PPM_in[EE_Parameter.Kanalbelegung[K_GIER]]) > 25)
{
tmp_long  /= 3;
tmp_long2 /= 3;
}
```

/*做一个限制，补偿值必须在一定的范围内。将补偿的范围限制在+-32*/

```
#define AUSGLEICH 32
if(tmp_long >  AUSGLEICH)
tmp_long  = AUSGLEICH;
if(tmp_long < -AUSGLEICH)
tmp_long  =-AUSGLEICH;
if(tmp_long2 > AUSGLEICH)
tmp_long2 = AUSGLEICH;
if(tmp_long2 <-AUSGLEICH)
tmp_long2 =-AUSGLEICH;
```

/*将补偿值考虑进去，这时候 Mess_IntegralNick 补偿了，Mess_IntegralNick2 没有补偿，因为在后面还要用到

```
*/
Mess_IntegralNick -= tmp_long;
```

```
Mess_IntegralRoll -= tmp_long2;
} // end if of if(!Looping_Nick && !Looping_Roll)
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++

/*当 >ABGLEICH_ANZAHL(256)时候 说明测量了 256 次航向*/
/*变量 ZaehlMessungen 是在 AD 检测的函数中改变的，也就是说，下面这个 if 语句是每 256
个检测周期计算一次，

而不是控制周期，检测周期要高于控制周期*/
if(ZaehlMessungen >= ABGLEICH_ANZAHL)//关于时间积累的处理过程
{
static int cnt = 0;
static char last_n_p,last_n_n,last_r_p,last_r_n;
static long MittelIntegralNick_Alt,MittelIntegralRoll_Alt;
if(!Looping_Nick && !Looping_Roll)
{
MittelIntegralNick  /= ABGLEICH_ANZAHL;
MittelIntegralRoll  /= ABGLEICH_ANZAHL;
/*计算加速度计积分的作用，在不运动时候，xy 加速度计的积分应该是 0，所以 xy 积

分而 z 不积分*/
IntegralAccNick = (EE_Parameter.GyroAccFaktor * IntegralAccNick) /

ABGLEICH_ANZAHL;
IntegralAccRoll = (EE_Parameter.GyroAccFaktor * IntegralAccRoll) /

ABGLEICH_ANZAHL;
IntegralAccZ    = IntegralAccZ / ABGLEICH_ANZAHL;
#define MAX_I 0//(Poti2/10)
// Nick +++++++++++++++++++++++++++++++++++++++++++++++++
/*不考虑补偿的陀螺仪积分的积分-加速度计积分/平衡项*/
IntegralFehlerNick = (long)(MittelIntegralNick - (long)IntegralAccNick);
ausgleichNick = IntegralFehlerNick / EE_Parameter.GyroAccAbgleich;
// Roll ++++++++++++++++++++++++++++++++++++++++++++++++
IntegralFehlerRoll = (long)(MittelIntegralRoll - (long)IntegralAccRoll);
ausgleichRoll = IntegralFehlerRoll / EE_Parameter.GyroAccAbgleich;

LageKorrekturNick = ausgleichNick / ABGLEICH_ANZAHL;
LageKorrekturRoll = ausgleichRoll / ABGLEICH_ANZAHL;

if((MaxStickNick > 15) || (MaxStickRoll > 15) || (abs(PPM_in

[EE_Parameter.Kanalbelegung[K_GIER]]) > 25))
```

```
{
/*这个参数在后面的程序中还要进行修正，修正后的值加入到陀螺仪的积分

中，可以认为这个参数是系统对于陀螺仪漂移的估计*/
LageKorrekturNick /= 2;
LageKorrekturRoll /= 2;
}

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Gyro-Drift ermitteln//陀螺漂移的确定
/*对陀螺仪漂移的估计过程*/
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
/*前面 MittelIntegralNick 已经用过了，因此这里使用 MittelIntegralNick2*/
MittelIntegralNick2 /= ABGLEICH_ANZAHL;
MittelIntegralRoll2 /= ABGLEICH_ANZAHL;
/*有校正和没有校正的陀螺仪积分做差，即陀螺仪的漂移*/
/*IntegralNick2 是没有校正的陀螺仪积分 IntegralNick 是有校正的陀螺仪积分 这里的校正指的是使用加速度

计积分进行的校正*/
tmp_long  = IntegralNick2 - IntegralNick;
tmp_long2 = IntegralRoll2 - IntegralRoll;
   //DebugOut.Analog[25] = MittelIntegralRoll2 / 26;

/*将差值加入到 Mess_IntegralNick2 和 Mess_IntegralRoll2 中 这时 Mess_IntegralNick2 和 Mess_IntegralRoll2

被使用*/
IntegralFehlerNick = tmp_long;
IntegralFehlerRoll = tmp_long2;
/*下面两个公式的作用就是让 Mess_IntegralNick2=Mess_IntegralNick，

Mess_IntegralRoll2=Mess_IntegralRoll 为下一个计算周期做准备*/
Mess_IntegralNick2 -= IntegralFehlerNick;
Mess_IntegralRoll2 -= IntegralFehlerRoll;

//    IntegralFehlerNick = (IntegralFehlerNick * 1 + tmp_long) / 2;
//    IntegralFehlerRoll = (IntegralFehlerRoll * 1 + tmp_long2) / 2;


DebugOut.Analog[17] = IntegralAccNick / 26;
DebugOut.Analog[18] = IntegralAccRoll / 26;
```

```
DebugOut.Analog[19] = IntegralFehlerNick;// / 26;
DebugOut.Analog[20] = IntegralFehlerRoll;// / 26;
DebugOut.Analog[21] = MittelIntegralNick / 26;
DebugOut.Analog[22] = MittelIntegralRoll / 26;
//DebugOut.Analog[28] = ausgleichNick;
DebugOut.Analog[29] = ausgleichRoll;
DebugOut.Analog[30] = LageKorrekturRoll * 10;

#define FEHLER_LIMIT  (ABGLEICH_ANZAHL * 4)
#define FEHLER_LIMIT2 (ABGLEICH_ANZAHL * 16)
#define BEWEGUNGS_LIMIT 20000


// Nick ++++++++++++++++++++++++++++++++++++++++++++++++++

/*以下部分就是对 LageKorrekturNick 的修正和对陀螺仪常值误差的修正*/
cnt = 1;// + labs(IntegralFehlerNick) / 4096;
      if(labs(MittelIntegralNick_Alt - MittelIntegralNick) < BEWEGUNGS_LIMIT)
      {
if(IntegralFehlerNick >  FEHLER_LIMIT2)
{
/*必须连续两次的误差都很大，才能进入下面的 if 语句*/
if(last_n_p)
{
/*连续两次误差较大时，对陀螺仪漂移进行补偿*/
/*最后改变了 LageKorrekturNick 的值*/
cnt += labs(IntegralFehlerNick) / FEHLER_LIMIT2;
ausgleichNick = IntegralFehlerNick / 8;
if(ausgleichNick > 5000)
ausgleichNick = 5000;
LageKorrekturNick += ausgleichNick / ABGLEICH_ANZAHL;
}
else
last_n_p = 1;
}
else
last_n_p = 0;

if(IntegralFehlerNick < -FEHLER_LIMIT2)
{
if(last_n_n)
{
cnt += labs(IntegralFehlerNick) / FEHLER_LIMIT2;
ausgleichNick = IntegralFehlerNick / 8;
```

```
if(ausgleichNick < -5000) ausgleichNick = -5000;
LageKorrekturNick += ausgleichNick / ABGLEICH_ANZAHL;
}
else
last_n_n = 1;
}
else
last_n_n = 0;
}
else cnt = 0;

if(cnt > EE_Parameter.Driftkomp)
cnt = EE_Parameter.Driftkomp;
/*在飞行器飞行的过程中，如果发现陀螺仪的中立点发生变化，则仍然进行修正*/
/*误差过大时候，改变陀螺仪的常值误差，每次最多改变 EE_Parameter.Driftkomp*/
if(IntegralFehlerNick >  FEHLER_LIMIT)
AdNeutralNick += cnt;

if(IntegralFehlerNick < -FEHLER_LIMIT)
AdNeutralNick -= cnt;

// Roll ++++++++++++++++++++++++++++++++++++++++++++++++++
cnt = 1;// + labs(IntegralFehlerNick) / 4096;

ausgleichRoll = 0;

if(labs(MittelIntegralRoll_Alt - MittelIntegralRoll) < BEWEGUNGS_LIMIT)
   {
if(IntegralFehlerRoll >  FEHLER_LIMIT2)
{
if(last_r_p)
{
cnt += labs(IntegralFehlerRoll) / FEHLER_LIMIT2;
ausgleichRoll = IntegralFehlerRoll / 8;
if(ausgleichRoll > 5000)
ausgleichRoll = 5000;
LageKorrekturRoll += ausgleichRoll / ABGLEICH_ANZAHL;
}
else
  last_r_p = 1;
}
else
last_r_p = 0;
      if(IntegralFehlerRoll < -FEHLER_LIMIT2)
```

```
    {
if(last_r_n)
{
cnt += labs(IntegralFehlerRoll) / FEHLER_LIMIT2;
ausgleichRoll = IntegralFehlerRoll / 8;
if(ausgleichRoll < -5000) ausgleichRoll = -5000;
LageKorrekturRoll += ausgleichRoll / ABGLEICH_ANZAHL;
}
else
last_r_n = 1;
    }
else
last_r_n = 0;
}
else
{
cnt = 0;
}

if(cnt > EE_Parameter.Driftkomp) cnt = EE_Parameter.Driftkomp;
  if(IntegralFehlerRoll >  FEHLER_LIMIT)
AdNeutralRoll += cnt;
  if(IntegralFehlerRoll < -FEHLER_LIMIT)
AdNeutralRoll -= cnt;

DebugOut.Analog[27] = ausgleichRoll;
DebugOut.Analog[23] = AdNeutralNick;//10*(AdNeutralNick - StartNeutralNick);
DebugOut.Analog[24] = 10*(AdNeutralRoll - StartNeutralRoll);
} // 整个的融合过程结束
else
{
LageKorrekturRoll = 0;
LageKorrekturNick = 0;
}

/*如果 IntegralFaktor 为零，也就是没有使用陀螺仪积分对电机输出进行修正，则不使用
LageKorrekturRoll，

也就是不进行陀螺仪漂移的补偿*/
/*在 Heading_hold 标志位被置位的情况下*/
if(!IntegralFaktor)
{
LageKorrekturRoll = 0;
LageKorrekturNick = 0;
```

```
} // z.B. bei HH

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
/*将上一次的值储存下来*/
  MittelIntegralNick_Alt = MittelIntegralNick;
  MittelIntegralRoll_Alt = MittelIntegralRoll;
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
/*数据清零 加速度计积分每一次都进行清零*/
   IntegralAccNick = 0;
   IntegralAccRoll = 0;
   IntegralAccZ = 0;
   MittelIntegralNick = 0;
   MittelIntegralRoll = 0;
   MittelIntegralNick2 = 0;
   MittelIntegralRoll2 = 0;
   ZaehlMessungen = 0;
} //end if of if(ZaehlMessungen >= ABGLEICH_ANZAHL)
//DebugOut.Analog[31] = StickRoll / (26*IntegralFaktor);

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
//  Gieren//偏航
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
   if(abs(StickGier) > 20) // war 35
   {
if(!(EE_Parameter.GlobalConfig & CFG_KOMPASS_FIX))
NeueKompassRichtungMerken = 1;
   }
   tmp_int  = (long) EE_Parameter.Gier_P * ((long)StickGier * abs(StickGier)) / 512L; // expo  y

= ax + bx?
   tmp_int += (EE_Parameter.Gier_P * StickGier) / 4;
   sollGier = tmp_int;
/*如果没有这句话 那么偏航轴的期望角度将一直等于  0  度 那么如果需要调整偏航轴的角
度 就必须一直

不断的进行偏航的修正 加上这句话后 就不用一直修正了*/
   Mess_Integral_Gier -= tmp_int;
   if(Mess_Integral_Gier > 50000) Mess_Integral_Gier = 50000;  // begrenzen 约束和限制
   if(Mess_Integral_Gier <-50000) Mess_Integral_Gier =-50000;

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
```

```
// Kompass
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
  if(KompassValue && (EE_Parameter.GlobalConfig & CFG_KOMPASS_AKTIV))
   {
    int w,v;
    static int SignalSchlecht = 0;
    w = abs(IntegralNick /512);
    v = abs(IntegralRoll /512);
    if(v > w) w = v;
    if(w < 25 && NeueKompassRichtungMerken && !SignalSchlecht)
     {
      KompassStartwert = KompassValue;
      NeueKompassRichtungMerken = 0;
     }
    w = (w * Parameter_KompassWirkung) / 64;
    w = Parameter_KompassWirkung - w;
    if(w > 0)
     {
      if(!SignalSchlecht) Mess_Integral_Gier += (KompassRichtung * w) / 32;
      if(SignalSchlecht) SignalSchlecht--;
     }
     else SignalSchlecht = 500;
   }
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
// Debugwerte zuordnen
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
 if(!TimerWerteausgabe--)
  {
   TimerWerteausgabe = 24;
   DebugOut.Analog[0] = IntegralNick / EE_Parameter.GyroAccFaktor;
   DebugOut.Analog[1] = IntegralRoll / EE_Parameter.GyroAccFaktor;
   DebugOut.Analog[2] = Mittelwert_AccNick;
   DebugOut.Analog[3] = Mittelwert_AccRoll;
   DebugOut.Analog[4] = MesswertGier;
   DebugOut.Analog[5] = HoehenWert;
   DebugOut.Analog[6] =(Mess_Integral_Hoch / 512);
   DebugOut.Analog[8] = KompassValue;
   DebugOut.Analog[9] = UBat;
```

```
  DebugOut.Analog[10] = SenderOkay;
  DebugOut.Analog[16] = Mittelwert_AccHoch;

/*   DebugOut.Analog[16] = motor_rx[0];
  DebugOut.Analog[17] = motor_rx[1];
  DebugOut.Analog[18] = motor_rx[2];
  DebugOut.Analog[19] = motor_rx[3];
  DebugOut.Analog[20] = motor_rx[0] + motor_rx[1] + motor_rx[2] + motor_rx[3];
  DebugOut.Analog[20] /= 14;
  DebugOut.Analog[21] = motor_rx[4];
  DebugOut.Analog[22] = motor_rx[5];
  DebugOut.Analog[23] = motor_rx[6];
  DebugOut.Analog[24] = motor_rx[7];
  DebugOut.Analog[25] = motor_rx[4] + motor_rx[5] + motor_rx[6] + motor_rx[7];
*/
//   DebugOut.Analog[9] = MesswertNick;
//   DebugOut.Analog[9] = SollHoehe;
//   DebugOut.Analog[10] = Mess_Integral_Gier / 128;
//   DebugOut.Analog[11] = KompassStartwert;
//   DebugOut.Analog[10] = Parameter_Gyro_I;
//   DebugOut.Analog[10] = EE_Parameter.Gyro_I;
//   DebugOut.Analog[9] = KompassRichtung;
//   DebugOut.Analog[10] = GasMischanteil;
//   DebugOut.Analog[3] = HoeheD * 32;
//   DebugOut.Analog[4] = hoehenregler;
 }

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
// Drehgeschwindigkeit und -winkel zu einem Istwert zusammenfassen//角速度和角度变化的
归纳部分
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
//DebugOut.Analog[26] = MesswertNick;
//DebugOut.Analog[28] = MesswertRoll;

/*对角度做 PD，也就是对角速率做了 PI*/
  if(Looping_Nick) MesswertNick = MesswertNick * GyroFaktor;
  else          MesswertNick = IntegralNick * IntegralFaktor + MesswertNick * GyroFaktor;
  if(Looping_Roll) MesswertRoll = MesswertRoll * GyroFaktor;
  else          MesswertRoll = IntegralRoll * IntegralFaktor + MesswertRoll * GyroFaktor;
  MesswertGier = MesswertGier * (2 * GyroFaktor) + Integral_Gier * IntegralFaktor / 2;

DebugOut.Analog[25] = IntegralRoll * IntegralFaktor;
```

```
DebugOut.Analog[31] = StickRoll;// / (26*IntegralFaktor);
DebugOut.Analog[28] = MesswertRoll;

/*对控制器输出进行幅度限制*/
  #define MAX_SENSOR  2048
  if(MesswertNick >  MAX_SENSOR) MesswertNick =  MAX_SENSOR;
  if(MesswertNick < -MAX_SENSOR) MesswertNick = -MAX_SENSOR;
  if(MesswertRoll >  MAX_SENSOR) MesswertRoll =  MAX_SENSOR;
  if(MesswertRoll < -MAX_SENSOR) MesswertRoll = -MAX_SENSOR;
  if(MesswertGier >  MAX_SENSOR) MesswertGier =  MAX_SENSOR;
  if(MesswertGier < -MAX_SENSOR) MesswertGier = -MAX_SENSOR;

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
// H 鰄 enregelung
// Die H 鰄 enregelung schw 鋍 ht lediglich das Gas ab, erh 鰄 t es allerdings nicht
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
//OCR0B = 180 - (Poti1 + 120) / 4;
//DruckOffsetSetting = OCR0B;
if((EE_Parameter.GlobalConfig & CFG_HOEHENREGELUNG))
{
int tmp_int;
  if(EE_Parameter.GlobalConfig & CFG_HOEHEN_SCHALTER)
  {
if(Parameter_MaxHoehe < 50)
{
SollHoehe = HoehenWert - 20;
HoehenReglerAktiv = 0;
}
else
HoehenReglerAktiv = 1;
}
  else
  {
SollHoehe = ((int) ExternHoehenValue + (int) Parameter_MaxHoehe) * (int)

EE_Parameter.Hoehe_Verstaerkung - 20;
HoehenReglerAktiv = 1;
  }

  if(Notlandung)
SollHoehe = 0;
  h = HoehenWert;
```

```
   if((h > SollHoehe) && HoehenReglerAktiv)
   {
h = ((h - SollHoehe) * (int) Parameter_Hoehe_P) / 16;
h = GasMischanteil - h;
h -= (HoeheD * Parameter_Luftdruck_D)/8;
tmp_int = ((Mess_Integral_Hoch / 512) * (signed long) Parameter_Hoehe_ACC_Wirkung)

/ 32;
if(tmp_int > 50)
tmp_int = 50;
else if(tmp_int < -50) tmp_int = -50;
h -= tmp_int;
hoehenregler = (hoehenregler*15 + h) / 16;
if(hoehenregler < EE_Parameter.Hoehe_MinGas)
{
if(GasMischanteil >= EE_Parameter.Hoehe_MinGas) hoehenregler =

EE_Parameter.Hoehe_MinGas;
if(GasMischanteil < EE_Parameter.Hoehe_MinGas) hoehenregler =

GasMischanteil;
}
if(hoehenregler > GasMischanteil) hoehenregler = GasMischanteil;
GasMischanteil = hoehenregler;
}
} // 高度调节器工作完成

 if(GasMischanteil > MAX_GAS - 20)
 GasMischanteil = MAX_GAS - 20;
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
// + Mischer und PI-Regler 在 PI 控制器下的混合数值
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
DebugOut.Analog[7] = GasMischanteil;
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
// Gier-Anteil//偏航部分
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
#define MUL_G  1.0
   GierMischanteil = MesswertGier - sollGier;
// GierMischanteil = 0;
```

```
/*对偏航数值进行限制，尽量避免最后计算出的四个电机的转速小于 0*/
   if(GierMischanteil > (GasMischanteil / 2)) GierMischanteil = GasMischanteil / 2;
   if(GierMischanteil < -(GasMischanteil / 2)) GierMischanteil = -(GasMischanteil / 2);
   if(GierMischanteil > ((MAX_GAS - GasMischanteil))) GierMischanteil = ((MAX_GAS -

GasMischanteil));
   if(GierMischanteil < -((MAX_GAS - GasMischanteil))) GierMischanteil = -((MAX_GAS -

GasMischanteil));
/*油门本身如果太小了，就限制偏航为 0*/
   if(GasMischanteil < 20) GierMischanteil = 0;//
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
// Nick-Achse 俯仰轴
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
/*PI 调节器*/
/*这个控制算法实际上是位置环为 PI 控制，速率环为 P 控制*/
   DiffNick = MesswertNick - (StickNick - GPS_Nick);
   if(IntegralFaktor) SummeNick += IntegralNick * IntegralFaktor - (StickNick - GPS_Nick);
   else  SummeNick += DiffNick;
   if(SummeNick >  16000) SummeNick =  16000;
   if(SummeNick < -16000) SummeNick = -16000;
   pd_ergebnis = DiffNick + Ki * SummeNick; //PD 控制结果为比例+积分控制
   // Motor Vorn
   tmp_int = (long)((long)Parameter_DynamicStability * (long)(GasMischanteil + abs

(GierMischanteil)/2)) / 64;
   if(pd_ergebnis >  tmp_int) pd_ergebnis =  tmp_int; //如果控制器输出太大，则要限制幅度
   if(pd_ergebnis < -tmp_int) pd_ergebnis = -tmp_int;

/*前后两个电机的实际输出*/
   motorwert = GasMischanteil + pd_ergebnis + GierMischanteil;
/*对电机数值进行限幅*/
if ((motorwert < 0))
motorwert = 0;
else if(motorwert > MAX_GAS)
motorwert = MAX_GAS;
if (motorwert < MIN_GAS)
motorwert = MIN_GAS;
Motor_Vorne = motorwert;
   // Motor Heck

motorwert = GasMischanteil - pd_ergebnis + GierMischanteil;
```

```
if ((motorwert < 0))
motorwert = 0;
else if(motorwert > MAX_GAS)
motorwert = MAX_GAS;
if (motorwert < MIN_GAS)
motorwert = MIN_GAS;
Motor_Hinten = motorwert;
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
// Roll-Achse 横滚轴
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++
DiffRoll = MesswertRoll - (StickRoll  - GPS_Roll); // Differenz bestimmen
   if(IntegralFaktor) SummeRoll += IntegralRoll * IntegralFaktor - (StickRoll  - GPS_Roll);// I-

Anteil bei Winkelregelung
   else         SummeRoll += DiffRoll;  // I-Anteil bei HH
   if(SummeRoll >  16000) SummeRoll =  16000;
   if(SummeRoll < -16000) SummeRoll = -16000;
   pd_ergebnis = DiffRoll + Ki * SummeRoll; // PI-Regler f 黵 Roll
   tmp_int = (long)((long)Parameter_DynamicStability * (long)(GasMischanteil + abs

(GierMischanteil)/2)) / 64;
   if(pd_ergebnis >  tmp_int) pd_ergebnis =  tmp_int;
   if(pd_ergebnis < -tmp_int) pd_ergebnis = -tmp_int;
   // Motor Links
   motorwert = GasMischanteil + pd_ergebnis - GierMischanteil;
#define GRENZE Poti1

if ((motorwert < 0)) motorwert = 0;
else if(motorwert > MAX_GAS) motorwert = MAX_GAS;
if (motorwert < MIN_GAS)          motorwert = MIN_GAS;
   Motor_Links = motorwert;
   // Motor Rechts
motorwert = GasMischanteil - pd_ergebnis - GierMischanteil;

if ((motorwert < 0)) motorwert = 0;
else if(motorwert > MAX_GAS)    motorwert = MAX_GAS;
if (motorwert < MIN_GAS)          motorwert = MIN_GAS;
   Motor_Rechts = motorwert;
  // ++++++++++++++++++++++++++++++++++++++++++++++
}
```