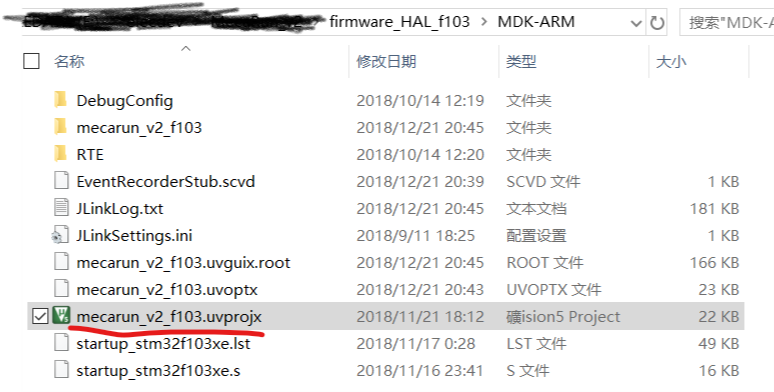
PID在线调试教程

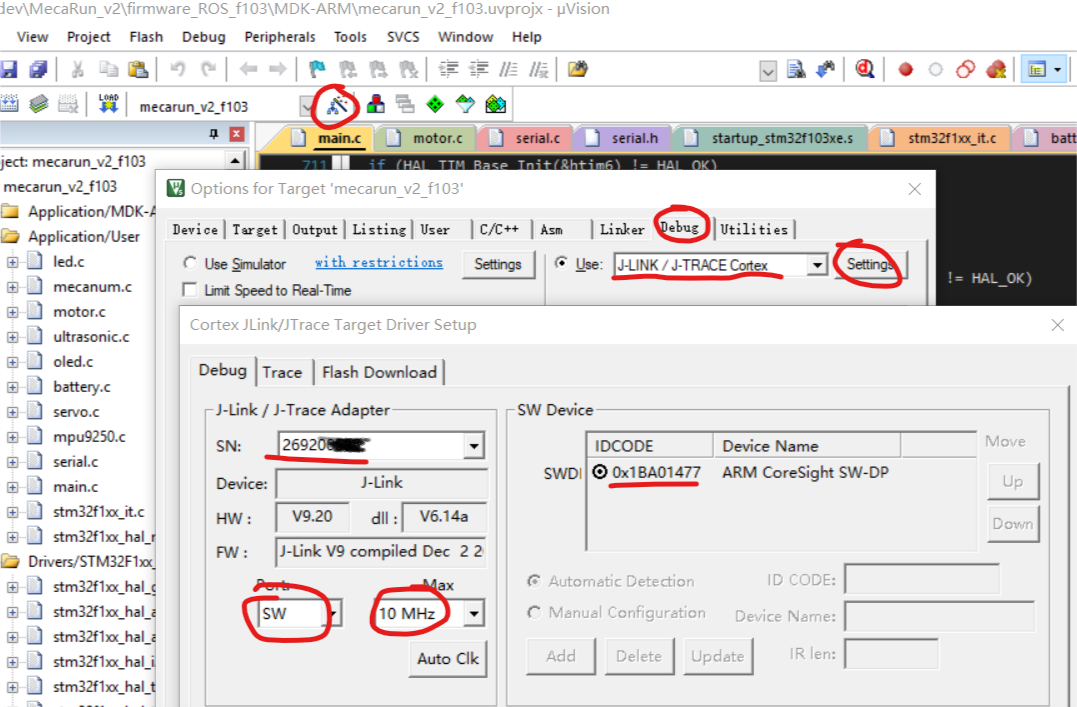
Pid调试上位机正在开发中，现在先用调试器的在线调试功能调！

1. 用mdk（keil）5打开工程



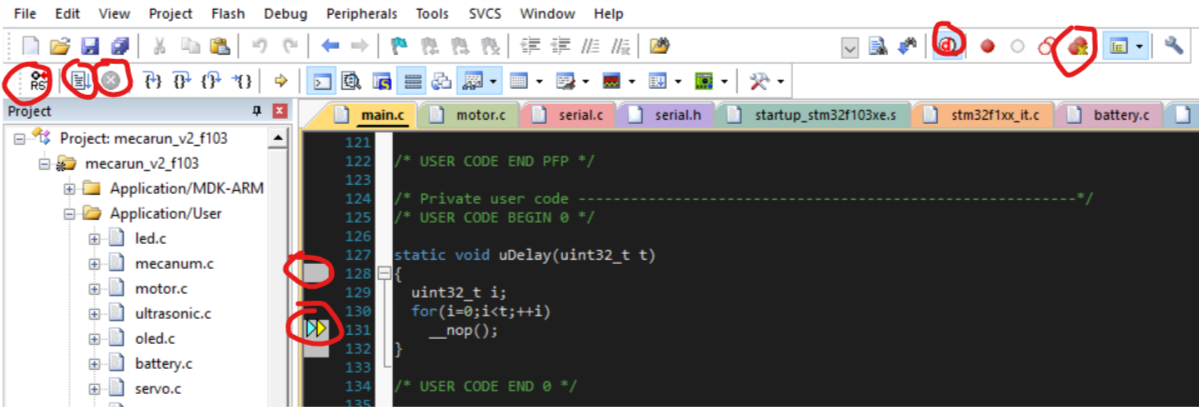
路径图所示

1. 首先进行调试器设置



在这之前，请先安装好调试器的驱动，在windows设备管理器应该能看到调试器而且没有感叹号。进入mdk，点击工程设置按钮，弹出来的窗口选择debug选项卡，根据你使用的是stlink、jlink还是daplink选择相应的调试器，选择后连接好调试器，以及接好调试器与主板之间的swdio，swclk以及gnd线，主板上电。这时候点击setting，弹出来的窗口中应该能看到调试器的sn号，然后选择sw模式，时钟频率可以选择到6m以上（stm32最高支持6m，但是stlink只能到4m），然后右边swdevice处应该能看见一个id号，这是cpuio。到这一步了说明硬件连接没有问题了。

1. 点击或者按快捷键Ctrl-F5进入在线调试



上面圈出来的常用按钮从左到右分别是复位（reset），运行（run），暂停（halt），进入/退出调试模式，取消所有断点

在代码编辑器的右侧，点击灰色的区域可以打断点或者取消断点，蓝色箭头表示当前程序运行到哪一行（pc指针），黄色表示汇编查看器里选中的汇编指令。

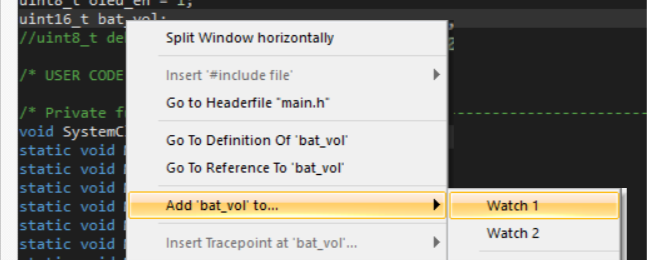
1. 调试tips

按复位前请先暂停（halt），有时候直接reset无效。

断点是表示程序执行这一行之前停下来，打断点最好在halt或者reset状态下打，否则有时候打不上去

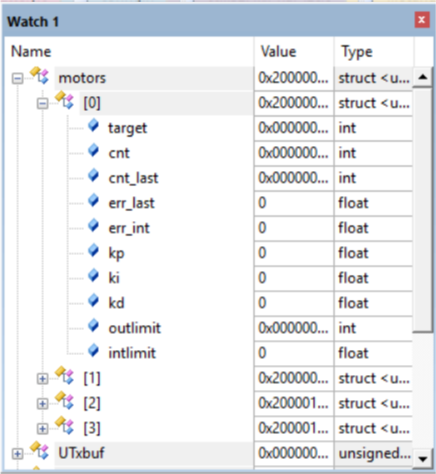
局部变量查看及修改请halt之后再查看，全局变量以及静态（static）变量可以全速运行的时候随时查看以及修改。

1. 观察变量



选择一个变量，按右键，选择Add “变量名”to 选Watch 1和 Watch 2

可以将变量添加进去观察列表中



我们利用此功能来调试pid

先说说数据结构与算法：

pid结构体

Target为目标速度

cnt为当前编码器读数，在计算pid前需先调用Get\_cnt(motors);从定时器寄存器同步编码器数据到motors结构体中

cnt\_last为上一次计算pid时cnt的值，两者相减可以得到当前时期电机转过的角度，也可以计算车走过的距离和电机转速

err\_last为上次误差，用于计算微分项

err\_int为误差积分

kp、ki、kd为pid的系数

outlimit为输出限幅，防止超出-1000~+1000的范围

intlimit为积分限幅，防止积分项过大

每个电机一个这样的结构体，在本工程中为motors[4]数组，0123分别对应左前、左后、右前、右后

pid的计算在motor.c中

第48行开始

|  |
| --- |
| {  time\_curr = HAL\_GetTick();//获取当前tick  if(time\_curr >= time\_next)//2ms计算一次，为了简便，pid三者采样时间取一样，调节系数即可  {  time\_len = (time\_curr - time\_next + ctrl->pid\_update\_period )\*1000.0 + (SYSTICK\_PRELOAD - SysTick->VAL)/SYSCLK\_MHZ;//计算该采样周期时长，1us 精度  for(i=0;i<4;++i)//计算每个电机的pid  {  currerr[i]=(float)mt[i].target\*3.08-(float)(mt[i].cnt-mt[i].cnt\_last)/time\_len\*1000000; //3.08这个系数是根据测试时用的电机在7.4v下100%占空比，空载，每秒钟编码器有3080个脉冲得到的  mt[i].err\_int += currerr[i];//计算积分项  mt[i].err\_int = (mt[i].err\_int > mt[i].intlimit)?mt[i].intlimit:mt[i].err\_int;  mt[i].err\_int = (mt[i].err\_int < -mt[i].intlimit)?(-mt[i].intlimit):mt[i].err\_int;//积分项限幅  mt[i].cnt\_last = mt[i].cnt;//储存编码器读数    #if MOTOR\_REVERSE == 0//根据编码器和电机转向的不同分两种情况  speed[i]=currerr[i]\*(mt[i].kp)+mt[i].err\_int\*(mt[i].ki)+(currerr[i]-mt[i].err\_last)\*(mt[i].kd);//计算pid输出    #else  speed[i]=-(currerr[i]\*(mt[i].kp)+mt[i].err\_int\*(mt[i].ki)+(currerr[i]-mt[i].err\_last)\*(mt[i].kd));  #endif    mt[i].err\_last = currerr[i];//储存误差  speed[i] = (speed[i] > mt[i].outlimit)?mt[i].outlimit:speed[i];  speed[i] = (speed[i] < -mt[i].outlimit)?(-mt[i].outlimit):speed[i];//输出限幅  }  ctrl->motor\_update = 1;//置位标志位，进行下一步计算  time\_next = time\_curr + ctrl->pid\_update\_period;//更新下一次执行时间  }  else  ctrl->motor\_update = 0;  } |

若需要观察电机实际转速，请取消注释掉56行

currspeedwatch[i]=(float)(mt[i].cnt-mt[i].cnt\_last)/(time\_curr-time\_next+ctrl->pid\_update\_period)\*1000.0/3.08;

以及24行int32\_t currspeedwatch[4];即可用watch功能观察及修改变量

1. pid调节tips

本工程中，我们定义了电机逆时针旋转为正转，第一步请先确认电机转向以及编码器计数方向。可以先禁用pid，在mian。C第187行注释Motor\_PID\_Enable(&common\_mt\_ctrl,motors);即可取消pid使能。

编译后进入调试模式，点击run全速运行，利用watch功能修改motors结构体，给其中一个电机的target设置1000，确认电机转向，若方向相反修改config.h中的MOTOR\_REVERSE，取值为0或1

接下来确认编码器方向，电机逆时针旋转编码器计数应该是增加的，若相反请修改config.h中的ENCODER\_REVERSE，取值为0或1

确认方向后可以选一只电机调节pid，调整完成之后所有电机一般可以共用一套参数。

先打一个断点在main.c的第188行，即执行完Motor\_PID\_Enable(&common\_mt\_ctrl,motors);，这时候修改pid系数，先调kp，将ki和kd设置为0，kp可以参考原先的参数上下调整一下，然后设置目标转速（-1000~+1000），点击运行，观察currspeedwatch是否能在不太大的范围内波动，err\_int会不会到限幅值。调得电机能够大致以某个速度转动（会有速度波动甚至暂停下来也正常）的时候可以开始调ki，ki调整范围较小，找到一个差不多的数值就ok，目前程序没有启用微分控制。

附：pid调节口诀

参数整定找最佳，从小到大顺序查

先是比例后积分，最后再把微分加

曲线振荡很频繁，比例度盘要放大

曲线漂浮绕大湾，比例度盘往小扳

曲线偏离回复慢，积分时间往下降

曲线波动周期长，积分时间再加长

曲线振荡频率快，先把微分降下来

动差大来波动慢。微分时间应加长

理想曲线两个波，前高后低4比1

一看二调多分析，调节质量不会低