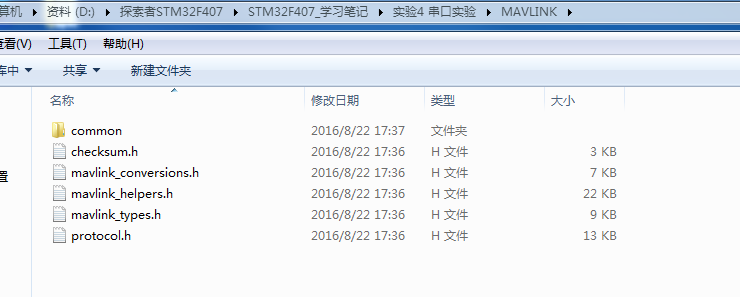
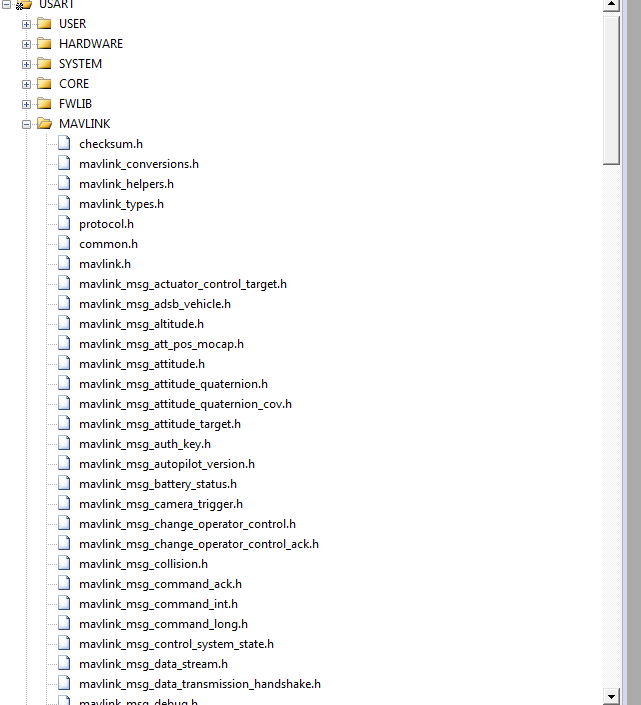
**2开始真正的移植工作**

a.在串口实验里新建一个MAVLINK文件夹，并将刚才生成的文件拷贝过来，如图



 打开工程新建分组，将所有拷贝过来的文件和头文件目录添加好，具体这里就不讲了,添加好后如图



 b.在main函数中添加#include "mavlink.h"如下



 然后编译

"..\OBJ\USART.axf" - 30 Error(s), 17 Warning(s).

一个一个解决。

**错误一：**

..\MAVLINK\common\../mavlink\_types.h(53): error:  #20: identifier "pack" is undefined

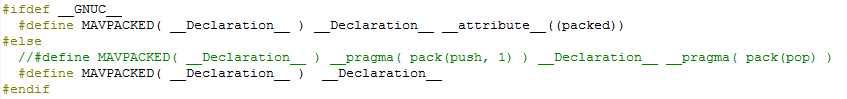
**解决方法：**

将mavlink\_types.h中

* 1. #define MAVPACKED( \_\_Declaration\_\_ ) \_\_pragma( pack(push, 1) ) \_\_Declaration\_\_ \_\_pragma( pack(pop) )

改为

* 1. #define MAVPACKED( \_\_Declaration\_\_ ) \_\_Declaration\_\_



这里不使用对齐字节了，直接用也是一样的。

**编译后：**

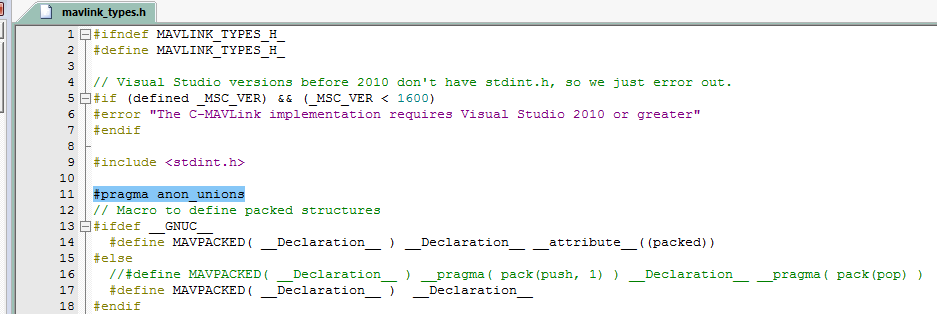
"..\OBJ\USART.axf" - 30 Error(s), 7 Warning(s).

**错误二：**

..\MAVLINK\common\../mavlink\_types.h(54): error:  #3092: anonymous unions are only supported in --gnu mode, or when enabled with #pragma anon\_unions

**解决方法：**

根据提示信息在mavlink\_types.h的前面加入#pragma anon\_unions



**编译后：**

"..\OBJ\USART.axf" - 30 Error(s), 8 Warning(s).

**错误三：**

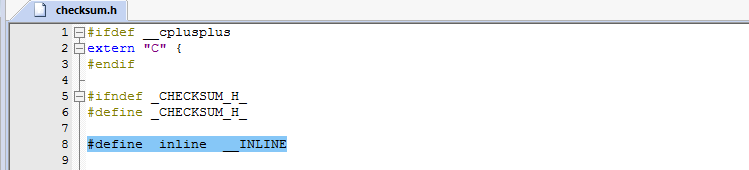
..\MAVLINK\common\../checksum.h(34): warning:  #260-D: explicit type is missing ("int" assumed)

包括后面很多错误都有共同点，指向有inline的行，这里是因为mdk中无法识别inline

**解决方法：**

在checksum.h加入一行代码定义inline

* 1. #define inline \_\_INLINE



**编译后：**

"..\OBJ\USART.axf" - 4 Error(s), 195 Warning(s).

**错误四：**

..\MAVLINK\common\../mavlink\_conversions.h(167): error:  #268: declaration may not appear after executable statement in block

**解决方法：**

指向定义的变量，将变量的定义放到函数的最前端（四个错误都是这种错误）

以前mavlink\_conversions.h的代码

* 1. MAVLINK\_HELPER void mavlink\_dcm\_to\_quaternion(const float dcm[3][3], float quaternion[4])
  2. {
  3. float tr = dcm[0][0] + dcm[1][1] + dcm[2][2];
  4. if (tr > 0.0f) {
  5. float s = sqrtf(tr + 1.0f);
  6. quaternion[0] = s \* 0.5f;
  7. s = 0.5f / s;
  8. quaternion[1] = (dcm[2][1] - dcm[1][2]) \* s;
  9. quaternion[2] = (dcm[0][2] - dcm[2][0]) \* s;
  10. quaternion[3] = (dcm[1][0] - dcm[0][1]) \* s;
  11. } else {
  12. /\* Find maximum diagonal element in dcm
  13. \* store index in dcm\_i \*/
  14. int dcm\_i = 0;
  15. int i;
  16. for (i = 1; i < 3; i++) {
  17. if (dcm[i][i] > dcm[dcm\_i][dcm\_i]) {
  18. dcm\_i = i;
  19. }
  20. }
  21. int dcm\_j = (dcm\_i + 1) % 3;
  22. int dcm\_k = (dcm\_i + 2) % 3;
  23. float s = sqrtf((dcm[dcm\_i][dcm\_i] - dcm[dcm\_j][dcm\_j] -
  24. dcm[dcm\_k][dcm\_k]) + 1.0f);
  25. quaternion[dcm\_i + 1] = s \* 0.5f;
  26. s = 0.5f / s;
  27. quaternion[dcm\_j + 1] = (dcm[dcm\_i][dcm\_j] + dcm[dcm\_j][dcm\_i]) \* s;
  28. quaternion[dcm\_k + 1] = (dcm[dcm\_k][dcm\_i] + dcm[dcm\_i][dcm\_k]) \* s;
  29. quaternion[0] = (dcm[dcm\_k][dcm\_j] - dcm[dcm\_j][dcm\_k]) \* s;
  30. }
  31. }

更改后mavlink\_conversions.h的代码

* 1. MAVLINK\_HELPER void mavlink\_dcm\_to\_quaternion(const float dcm[3][3], float quaternion[4])
  2. {
  3. int dcm\_j,dcm\_k;
  4. float s;
  5. float tr = dcm[0][0] + dcm[1][1] + dcm[2][2];
  6. if (tr > 0.0f) {
  7. float s = sqrtf(tr + 1.0f);
  8. quaternion[0] = s \* 0.5f;
  9. s = 0.5f / s;
  10. quaternion[1] = (dcm[2][1] - dcm[1][2]) \* s;
  11. quaternion[2] = (dcm[0][2] - dcm[2][0]) \* s;
  12. quaternion[3] = (dcm[1][0] - dcm[0][1]) \* s;
  13. } else {
  14. /\* Find maximum diagonal element in dcm
  15. \* store index in dcm\_i \*/
  16. int dcm\_i = 0;
  17. int i;
  18. for (i = 1; i < 3; i++) {
  19. if (dcm[i][i] > dcm[dcm\_i][dcm\_i]) {
  20. dcm\_i = i;
  21. }
  22. }
  23. dcm\_j = (dcm\_i + 1) % 3;
  24. dcm\_k = (dcm\_i + 2) % 3;
  25. s = sqrtf((dcm[dcm\_i][dcm\_i] - dcm[dcm\_j][dcm\_j] -
  26. dcm[dcm\_k][dcm\_k]) + 1.0f);
  27. quaternion[dcm\_i + 1] = s \* 0.5f;
  28. s = 0.5f / s;
  29. quaternion[dcm\_j + 1] = (dcm[dcm\_i][dcm\_j] + dcm[dcm\_j][dcm\_i]) \* s;
  30. quaternion[dcm\_k + 1] = (dcm[dcm\_k][dcm\_i] + dcm[dcm\_i][dcm\_k]) \* s;
  31. quaternion[0] = (dcm[dcm\_k][dcm\_j] - dcm[dcm\_j][dcm\_k]) \* s;
  32. }
  33. }

以前mavlink\_helpers.h的代码

* 1. MAVLINK\_HELPER uint16\_t mavlink\_msg\_to\_send\_buffer(uint8\_t \*buffer, const mavlink\_message\_t \*msg)
  2. {
  3. memcpy(buffer, (const uint8\_t \*)&msg->magic, MAVLINK\_NUM\_HEADER\_BYTES + (uint16\_t)msg->len);
  4. uint8\_t \*ck = buffer + (MAVLINK\_NUM\_HEADER\_BYTES + (uint16\_t)msg->len);
  5. ck[0] = (uint8\_t)(msg->checksum & 0xFF);
  6. ck[1] = (uint8\_t)(msg->checksum >> 8);
  7. return MAVLINK\_NUM\_NON\_PAYLOAD\_BYTES + (uint16\_t)msg->len;
  8. }

更改后mavlink\_helpers.h的代码

* 1. MAVLINK\_HELPER uint16\_t mavlink\_msg\_to\_send\_buffer(uint8\_t \*buffer, const mavlink\_message\_t \*msg)
  2. {
  3. uint8\_t \*ck;
  4. memcpy(buffer, (const uint8\_t \*)&msg->magic, MAVLINK\_NUM\_HEADER\_BYTES + (uint16\_t)msg->len);
  5. ck = buffer + (MAVLINK\_NUM\_HEADER\_BYTES + (uint16\_t)msg->len);
  6. ck[0] = (uint8\_t)(msg->checksum & 0xFF);
  7. ck[1] = (uint8\_t)(msg->checksum >> 8);
  8. return MAVLINK\_NUM\_NON\_PAYLOAD\_BYTES + (uint16\_t)msg->len;
  9. }

**编译后：**

"..\OBJ\USART.axf" - 0 Error(s), 195 Warning(s).

**解决所有警告的方法：**

将protocol.h里面的旧代码

* 1. #define \_MAV\_RETURN\_char(msg, wire\_offset) (const char)\_MAV\_PAYLOAD(msg)[wire\_offset]
  2. #define \_MAV\_RETURN\_int8\_t(msg, wire\_offset) (const int8\_t)\_MAV\_PAYLOAD(msg)[wire\_offset]
  3. #define \_MAV\_RETURN\_uint8\_t(msg, wire\_offset) (const uint8\_t)\_MAV\_PAYLOAD(msg)[wire\_offset]

去掉const，改为

* 1. #define \_MAV\_RETURN\_char(msg, wire\_offset) (const char)\_MAV\_PAYLOAD(msg)[wire\_offset]
  2. #define \_MAV\_RETURN\_int8\_t(msg, wire\_offset) (int8\_t)\_MAV\_PAYLOAD(msg)[wire\_offset]
  3. #define \_MAV\_RETURN\_uint8\_t(msg, wire\_offset) (uint8\_t)\_MAV\_PAYLOAD(msg)[wire\_offset]

**编译后：**

"..\OBJ\USART.axf" - 0 Error(s), 0 Warning(s).