1. #include "vxWorks.h"
2. #include "msgQLib.h"
3. /\* defines \*/
4. #define MAX\_MSGS (10)
5. #define MAX\_MSG\_LEN (100)
6. MSG\_Q\_ID myMsgQId;
7. int a;
9. //任务3仅仅创建消息队列
10. task3(void)
11. {
12. /\* create message queue \*/
13. if ((myMsgQId = msgQCreate (MAX\_MSGS, MAX\_MSG\_LEN, MSG\_Q\_PRIORITY)) == NULL)
14. return (ERROR);
15. a=100;
16. }
17. //任务2完成接收，并显示
18. task2 (void)
19. {
20. **char** msgBuf[MAX\_MSG\_LEN];
21. /\* get message from queue; if necessary wait until msg is available \*/
22. if (**msgQReceive**(myMsgQId, msgBuf, MAX\_MSG\_LEN, **WAIT\_FOREVER**) == ERROR)
23. return (ERROR);
24. /\* display message \*/
25. **a=200;**
26. logMsg ("Message from task 1:\n%s\n", msgBuf, 0,0,0,0,0);
27. }
28. //任务1里延时5秒输出
29. #define MESSAGE "Greetings from Task 1"
30. task1 (void)
31. {
32. taskDelay (sysClkRateGet()\*5);//延时5秒
33. **a=300;**
34. if (**msgQSend** (myMsgQId, MESSAGE, MAX\_MSG\_LEN, **WAIT\_FOREVER**, MSG\_PRI\_NORMAL) == ERROR)
35. return (ERROR);
36. /\* send a normal priority message, blocking if queue is full
37. if (msgQSend (myMsgQId, MESSAGE, sizeof (MESSAGE), WAIT\_FOREVER,
38. MSG\_PRI\_NORMAL) == ERROR)
39. return (ERROR);
40. \*/
41. }
42. **void TestMsgQ(void)**
43. {
44. taskSpawn("t3",100,0,0x20000,(FUNCPTR)task3,0,0,0,0,0,0,0,0,0,0);
45. taskSpawn("t2",100,0,0x20000,(FUNCPTR)task2,0,0,0,0,0,0,0,0,0,0);
46. taskSpawn("t1",100,0,0x20000,(FUNCPTR)task1,0,0,0,0,0,0,0,0,0,0);
47. }