**电子科技大学**



**嵌入式系统实验**

**实验四**

**学号：202222080625**

**姓名：乔翱**

**学院：计算机科学与工程学院**

1. **实验名称**

一种汽车电子控制器（ECU）嵌入式软件的综合开发需求

1. **实验时间**

2023年5月30日

**三、实验内容和目的**

1、综合之前学过的知识，完成一个嵌入式软件的综合开发。

2、完成实验四的要求。

**四、实验设计**

系统的任务总共分为初始化相关任务、周期任务和同步相关任务。

初始化任务：包括ini\_LLD、HLS\_ini、HLS\_inisyn、HLS\_ini2、ini\_LLD2等任务，完成各种初始化操作；StartOS执行后，按下列顺序执行初始化任务：ini\_LLD、HLS\_ini、HLS\_inisyn、HLS\_ini2、ini\_LLD2。其中ini\_LLD为是自动启动的，而其他初始化任务则由其上一个初始化任务启动。初始化任务设置为不可被抢占。

周期任务：有50ms任务、100ms任务、200ms任务、1000ms任务（可假设1ms为10tick）。根据任务的周期设置其优先级，周期越短优先级越高。各周期任务可通过alarm被激活，并具有可被抢占的性质。

同步相关任务：有HLS\_syn2、HLS\_rstsyn、HLS\_firstsyn、HLS\_syn等任务。这些任务的具体功能（即输出信息）自行定义，但需要事件机制与前述的周期任务进行同步，在得到相应周期任务的事件通知后启动运行，输出相关信息。这些同步任务的优先级需要与相关的周期任务优先级相适应。

通信任务：设计4个任务与同步任务进行通信。在四个同步相关任务中，都进行了消息的发送，而在四个通信任务中，进行了消息的接收，实现通信。

**五、任务配置**

按照实验设计对总共17个任务进行配置，项目中任务的相关配置如下。这里重点关注任务初始状态的配置，对于需要使用事件机制的任务需要配置扩展任务，任务默认是可抢占，对于初始任务需要配置不可抢占。

|  |
| --- |
| /\* task configurations \*/  const T\_OSEK\_TASK\_ConfigTable\_Struct osekConfig\_TaskTable[OCC\_NTSKS]=  {  /\*ini\_LLD ID:0 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-0-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskini\_LLD, //task function  ini\_LLD, //task ID  0 | OSEK\_TASK\_ACTIVE | OSEK\_TASK\_NONPREEMPT | OSEK\_TASK\_EXTENDED, //task initial attributes  &taskStack[ini\_LLD][TASK\_STACK\_SIZE-1], //top of the task stack  &taskStack[ini\_LLD][0] //bottom of the task stack    },  /\*HLS\_ini ID:1 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-1-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskHLS\_ini,  HLS\_ini,  OSEK\_TASK\_NONPREEMPT | OSEK\_TASK\_EXTENDED,  &taskStack[HLS\_ini][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_ini][0]  },  /\*HLS\_inisyn ID:2 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-2-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskHLS\_inisyn,  HLS\_inisyn,  OSEK\_TASK\_NONPREEMPT | OSEK\_TASK\_EXTENDED,  &taskStack[HLS\_inisyn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_inisyn][0]  },  /\*HLS\_ini2 ID:3 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-3-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskHLS\_ini2,  HLS\_ini2,  OSEK\_TASK\_NONPREEMPT | OSEK\_TASK\_EXTENDED,  &taskStack[HLS\_ini2][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_ini2][0]  },  /\*ini\_LLD2 ID:4 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-4-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskini\_LLD2,  ini\_LLD2,  OSEK\_TASK\_NONPREEMPT | OSEK\_TASK\_EXTENDED,  &taskStack[ini\_LLD2][TASK\_STACK\_SIZE-1],  &taskStack[ini\_LLD2][0]  },  /\*period50 ID:5 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-5-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskperiod50,  period50,  OSEK\_TASK\_EXTENDED,  &taskStack[period50][TASK\_STACK\_SIZE-1],  &taskStack[period50][0]  },  /\*period100 ID:6 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-6-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskperiod100,  period100,  OSEK\_TASK\_EXTENDED,  &taskStack[period100][TASK\_STACK\_SIZE-1],  &taskStack[period100][0]  },  /\*period200 ID:7 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-7-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskperiod200,  period200,  OSEK\_TASK\_EXTENDED,  &taskStack[period200][TASK\_STACK\_SIZE-1],  &taskStack[period200][0]  },  /\*period1000 ID:8 priority:CONFIG\_OSEK\_TASK\_PRIORITY\_NUMBER-8-1 \*/  {  (T\_OSEK\_TASK\_Entry)FuncTaskperiod1000,  period1000,  OSEK\_TASK\_EXTENDED,  &taskStack[period1000][TASK\_STACK\_SIZE-1],  &taskStack[period1000][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskHLS\_syn2,  HLS\_syn2,  OSEK\_TASK\_EXTENDED | OSEK\_TASK\_ACTIVE,  &taskStack[HLS\_syn2][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_syn2][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskHLS\_rstsyn,  HLS\_rstsyn,  OSEK\_TASK\_EXTENDED | OSEK\_TASK\_ACTIVE,  &taskStack[HLS\_rstsyn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_rstsyn][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskHLS\_firstsyn,  HLS\_firstsyn,  OSEK\_TASK\_EXTENDED | OSEK\_TASK\_ACTIVE,  &taskStack[HLS\_firstsyn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_firstsyn][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskHLS\_syn,  HLS\_syn,  OSEK\_TASK\_EXTENDED | OSEK\_TASK\_ACTIVE,  &taskStack[HLS\_syn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_syn][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskCOM\_task1,  COM\_task1,  OSEK\_TASK\_EXTENDED ,  &taskStack[HLS\_syn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_syn][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskCOM\_task2,  COM\_task2,  OSEK\_TASK\_EXTENDED ,  &taskStack[HLS\_syn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_syn][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskCOM\_task3,  COM\_task3,  OSEK\_TASK\_EXTENDED ,  &taskStack[HLS\_syn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_syn][0]  },  {  (T\_OSEK\_TASK\_Entry)FuncTaskCOM\_task4,  COM\_task4,  OSEK\_TASK\_EXTENDED ,  &taskStack[HLS\_syn][TASK\_STACK\_SIZE-1],  &taskStack[HLS\_syn][0]  },    /\*Task Idle \*/  {  0,  OSEK\_TASK\_IDLE\_ID,  OSEK\_TASK\_ACTIVE,  },  }; |

**六、实验步骤**

1、修改osprop.h中关于任务数、时钟数量和通信消息数量的配置。

2、在cfg.h和cfh.c配置相关任务，包括任务的定义声明以及任务相关属性的配置，重点配置任务的初始状态。

配置时钟和消息，相关配置如下。

|  |
| --- |
| //ALARM configuration lists  #define TaskAlarm1 &osekAlarm\_AlarmTable[0]  #define TaskAlarm2 &osekAlarm\_AlarmTable[1]  #define TaskAlarm3 &osekAlarm\_AlarmTable[2]  #define TaskAlarm4 &osekAlarm\_AlarmTable[3]  #if defined (OCC\_LOCALMESSAGESONLY)  //message IDs  #define Msg1 &osekMsg\_MessageTable[0]  #define Msg2 &osekMsg\_MessageTable[1]  #define Msg3 &osekMsg\_MessageTable[2]  #define Msg4 &osekMsg\_MessageTable[3]  T\_OSEK\_MESSAGE\_ConfigTable\_Struct osekConfig\_MsgTable[OCC\_NMSGS] =  {  {  0, //ID  &messageBuf[0], //message butter  sizeof(messageBuf[0]), //size of the message buffer  &messageQueue[0], //message queue  MESSAGE\_BUF\_SIZE //size of the queue  },  {  1, //ID  &messageBuf[1],  sizeof(messageBuf[1]),  &messageQueue[1],  MESSAGE\_BUF\_SIZE  },  {  2, //ID  &messageBuf[2],  sizeof(messageBuf[2]),  &messageQueue[2],  MESSAGE\_BUF\_SIZE  },  {  3, //ID  &messageBuf[3],  sizeof(messageBuf[3]),  &messageQueue[3],  MESSAGE\_BUF\_SIZE  }  };  //ALARM configuration  T\_OSEK\_ALARM\_ConfigTable osekConfig\_AlarmTable[ ] =  {  {  period50, //the task notified by the Alarm  0x00, //the event to be set for the task 如果是0 代表就是激活事件  SysCounter, //name of the counter that the alarm uses  },  {  period100, //the task notified by the Alarm    0x00, //the event to be set for the task 如果是0 代表就是激活事件  SysCounter, //name of the counter that the alarm uses  },  {  period200, //the task notified by the Alarm    0x00, //the event to be set for the task 如果是0 代表就是激活事件  SysCounter, //name of the counter that the alarm uses  },  {  period1000, //the task notified by the Alarm    0x00, //the event to be set for the task 如果是0 代表就是激活事件  SysCounter, //name of the counter that the alarm uses  },  }; |

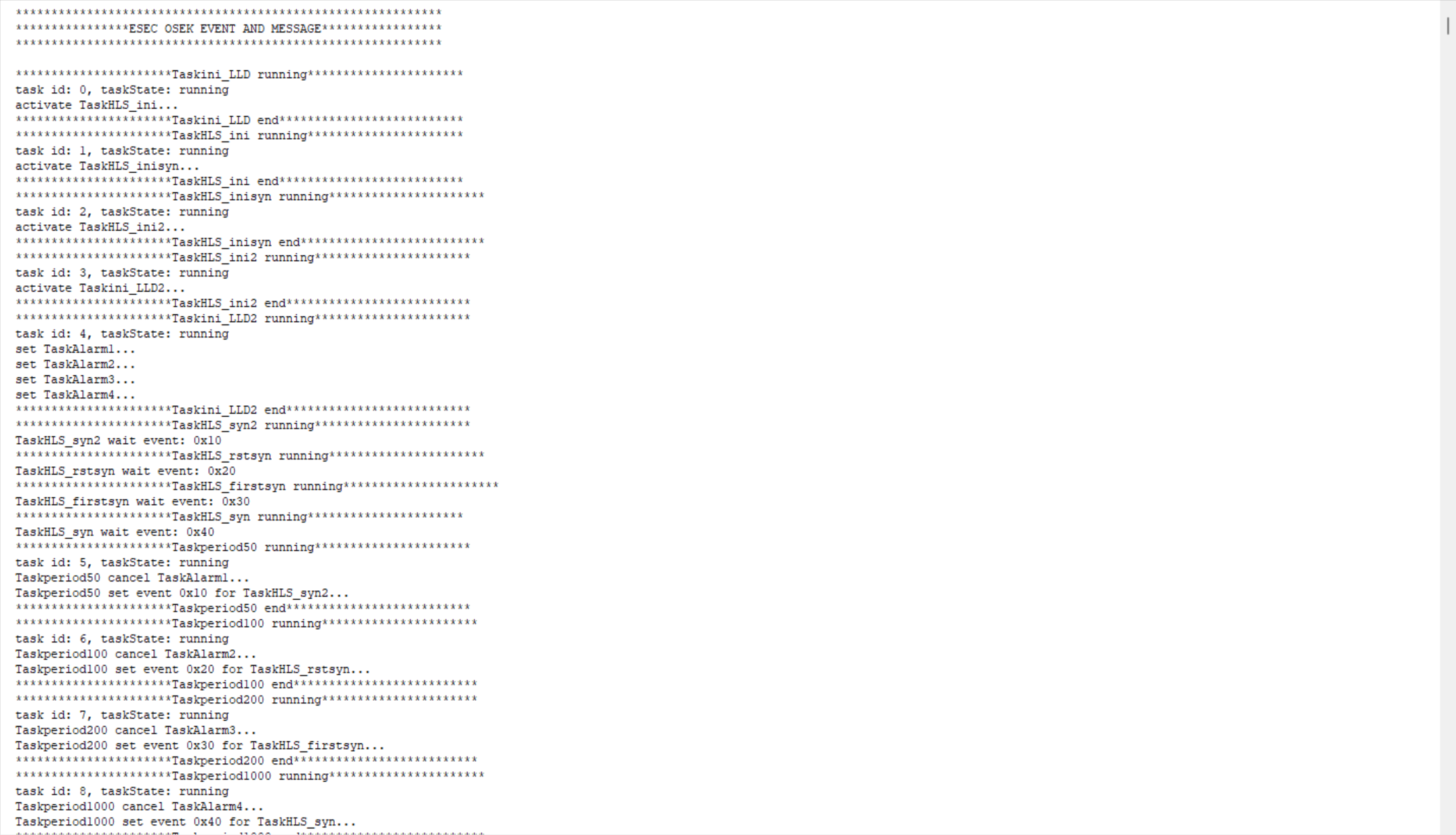
配置时钟以及资源，时钟的相关配置如下。

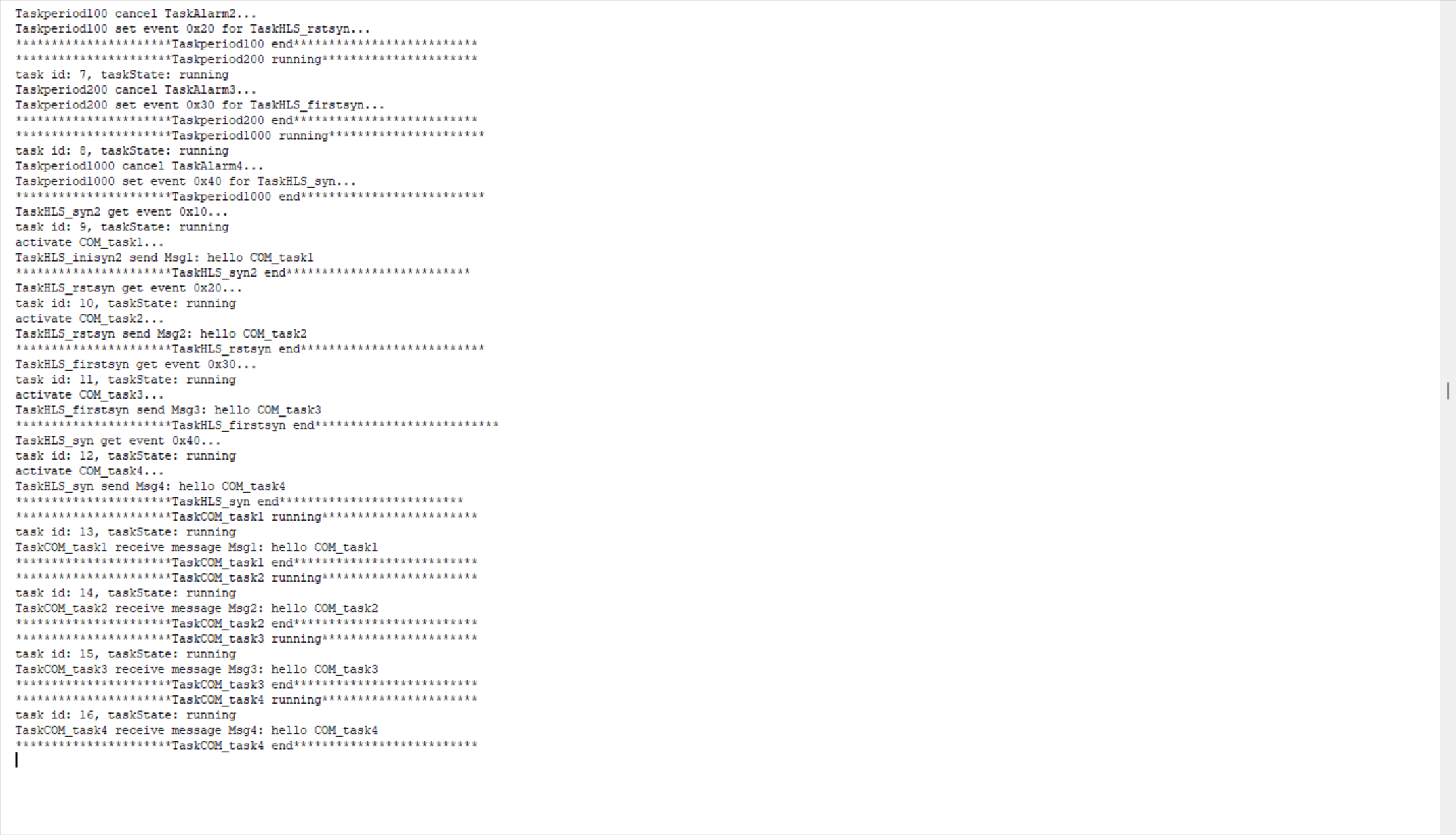
3、在init.c中编写每个任务函数，实现每个任务的相关功能。

任务的功能按照之前的设计进行编写，完成整个实验的功能。

**七、实验结果及分析**

在IDE中编译，并运行，实验结果如下：





分析实验结果，可以看到程序按照预先设计进行，首先四个初始化任务按顺序初始化，第一个初始化任务是自动启动的，之后的任务是由前一个初始化任务激活的。

在最后一个初始化任务中设置了初始化了四个时钟，四个时钟分别用来激活激活时钟任务，根据实验结果可以看到，四个时钟任务定期执行。

在每个周期任务中，为同步相关任务设置了event，四个同步相关任务只有在周期任务执行后，得到需要的event，这些同步相关任务才会执行。

在四个同步相关任务中，分别与四个通信任务进行通信，可以看到四个通信任务顺利收到了信息，成功进行了通信。

**八、实验结论及心得体会**

通过本次实验，综合了之前学过的知识，任务的配置、事件机制的使用，通过事件机制来达成任务之间的同步，任务之间进行通信，时钟任务的使用以及资源机制。

通过这门实验课学习到了很多的东西，之前并没有接触过太多底层开发相关知识，通过这门嵌入式实验，学习到了如何进行底层开发，学习了很多底层开发相关知识，收获颇丰。