OS_TASKS

```
/*lint +e956 */

/* Debugging and trace facilities private variables and macros. ------------*/

/*
 * The value used to fill the stack of a task when the task is created.  This
 * is used purely for checking the high water mark for tasks.
 */
#define tskSTACK_FILL_BYTE  ( 0xa5U )

/*
 * Macros used by vListTask to indicate which state a task is in.
 */
#define tskBLOCKED_CHAR   ( 'B' )
#define tskREADY_CHAR   ( 'R' )
#define tskDELETED_CHAR   ( 'D' )
#define tskSUSPENDED_CHAR ( 'S' )

/*-----------------------------------------------------------*/

#if ( configUSE_PORT_OPTIMISED_TASK_SELECTION == 0 )

/* If configUSE_PORT_OPTIMISED_TASK_SELECTION is 0 then task selection is
  performed in a generic way that is not optimised to any particular
  microcontroller architecture. */

/* uxTopReadyPriority holds the priority of the highest priority ready
  state task. */
#define taskRECORD_READY_PRIORITY( uxPriority )                           \
        {                                                   \
    if( ( uxPriority ) > uxTopReadyPriority )                           \
    {                                               \
        uxTopReadyPriority = ( uxPriority );                           \
    }                                               \
        } /* taskRECORD_READY_PRIORITY */

/*-----------------------------------------------------------*/

#define taskSELECT_HIGHEST_PRIORITY_TASK()                            \
        {                                                   \
    /* Find the highest priority queue that contains ready tasks. */               \
        while( listLIST_IS_EMPTY( &( pxReadyTasksLists[ uxTopReadyPriority ] ) ) )
\
        {                                               \
            configASSERT( uxTopReadyPriority );                           \
            --uxTopReadyPriority;                                   \
        }                                               \
        \
        /* listGET_OWNER_OF_NEXT_ENTRY indexes through the list, so the tasks of
```

```
\
    the same priority get an equal share of the processor time. */                 \
        listGET_OWNER_OF_NEXT_ENTRY( pxCurrentTCB, &( pxReadyTasksLists[ uxTopReadyPri
ority ] ) );    \
        } /* taskSELECT_HIGHEST_PRIORITY_TASK */

/*-----------------------------------------------------------*/

/* Define away taskRESET_READY_PRIORITY() and portRESET_READY_PRIORITY() as
  they are only required when a port optimised method of task selection is
  being used. */
#define taskRESET_READY_PRIORITY( uxPriority )
#define portRESET_READY_PRIORITY( uxPriority, uxTopReadyPriority )

#else /* configUSE_PORT_OPTIMISED_TASK_SELECTION */

/* If configUSE_PORT_OPTIMISED_TASK_SELECTION is 1 then task selection is
  performed in a way that is tailored to the particular microcontroller
  architecture being used. */

/* A port optimised version is provided.  Call the port defined macros. */
#define taskRECORD_READY_PRIORITY( uxPriority ) portRECORD_READY_PRIORITY( uxPriority,
uxTopReadyPriority )

/*-----------------------------------------------------------*/

#define taskSELECT_HIGHEST_PRIORITY_TASK()                          \
        {                                                   \
    UBaseType_t uxTopPriority;                                      \
    \
    /* Find the highest priority queue that contains ready tasks. */             \
        portGET_HIGHEST_PRIORITY( uxTopPriority, uxTopReadyPriority );
\
        configASSERT( listCURRENT_LIST_LENGTH( &( pxReadyTasksLists[ uxTopPriority ] )
) > 0 );    \
        listGET_OWNER_OF_NEXT_ENTRY( pxCurrentTCB, &( pxReadyTasksLists[ uxTopPriority
] ) );    \
        } /* taskSELECT_HIGHEST_PRIORITY_TASK() */

/*-----------------------------------------------------------*/

/* A port optimised version is provided, call it only if the TCB being reset
  is being referenced from a ready list.  If it is referenced from a delayed
  or suspended list then it won't be in a ready list. */
#define taskRESET_READY_PRIORITY( uxPriority )                          \
        {                                                   \
    if( listCURRENT_LIST_LENGTH( &( pxReadyTasksLists[ ( uxPriority ) ] ) ) == ( UBase
Type_t ) 0 )  \
    {                                                   \
        portRESET_READY_PRIORITY( ( uxPriority ), ( uxTopReadyPriority ) );
\
    }                                                   \
        }

#endif /* configUSE_PORT_OPTIMISED_TASK_SELECTION */

/*-----------------------------------------------------------*/
```

```
/* pxDelayedTaskList and pxOverflowDelayedTaskList are switched when the tick
count overflows. */
#define taskSWITCH_DELAYED_LISTS()                                          \
        {                                                                   \
    List_t *pxTemp;                                                         \
    \
    /* The delayed tasks list should be empty when the lists are switched. */
\
        configASSERT( ( listLIST_IS_EMPTY( pxDelayedTaskList ) ) );
\
        \
        pxTemp = pxDelayedTaskList;                                         \
        pxDelayedTaskList = pxOverflowDelayedTaskList;                              \
        pxOverflowDelayedTaskList = pxTemp;                                    \
        xNumOfOverflows++;                                                 \
        prvResetNextTaskUnblockTime();                                        \
        }

/*-----------------------------------------------------------*/

/*
 * Place the task represented by pxTCB into the appropriate ready list for
 * the task.  It is inserted at the end of the list.
 */
#define prvAddTaskToReadyList( pxTCB )                                      \
        traceMOVED_TASK_TO_READY_STATE( pxTCB )                                \
        taskRECORD_READY_PRIORITY( ( pxTCB )->uxPriority );                           \
        vListInsertEnd( &( pxReadyTasksLists[ ( pxTCB )->uxPriority ] ), &( ( pxTCB )-
>xGenericListItem ) )
/*-----------------------------------------------------------*/
```

Yes, this code has a direct relationship with the implementation of a ready queue.

The ready queue is an essential component of an RTOS, where tasks that are ready to run are placed in the ready queue. The ready queue is typically implemented as an array of linked lists, with each list representing a priority level. Each task is assigned a priority, and it is placed in the corresponding priority level list in the ready queue.

The code you provided implements the task selection algorithm for selecting the highest-priority task that is ready to run. The task selection algorithm is responsible for selecting the highest priority task in the ready queue for execution by the CPU. The task selection algorithm is implemented using macros, and the algorithm depends on the configuration option `configUSE_PORT_OPTIMISED_TASK_SELECTION`.

If `configUSE_PORT_OPTIMISED_TASK_SELECTION` is 0, the task selection algorithm is performed in a generic way that is not optimized for any particular microcontroller architecture. In this case, the algorithm finds the highest priority queue that contains ready tasks and selects the next task in that queue for execution.

If `configUSE_PORT_OPTIMISED_TASK_SELECTION` is 1, the task selection algorithm is optimized for a particular microcontroller architecture. In this case, the algorithm is implemented using port-specific macros.

The code also includes a macro `prvAddTaskToReadyList` which inserts a task into the appropriate ready list based on its priority. The `traceMOVED_TASK_TO_READY_STATE` macro is used to trace the task's state change from blocked to ready.