\* ROSA\_tcbCreate

\*

\* Comment:

\* Create the TCB with correct values.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ROSA\_tcbCreate(tcbHandle \*tcbTask, char tcbName[NAMESIZE], void \*tcbFunction, int \* tcbStack, int tcbStackSize, int taskPriority, void \*tcbArg, void \*semaphores, int semaCount)

{

int i;

//Initialize the tcb with the correct values

for(i = 0; i < NAMESIZE; i++) {

//Copy the id/name

tcbTask->id[i] = tcbName[i];

}

//Dont link this TCB anywhere yet.

tcbTask->nexttcb = NULL;

//Set the task function start and return address.

tcbTask->staddr = tcbFunction;

tcbTask->retaddr = (int)tcbFunction;

//Set up the stack.

tcbTask->datasize = tcbStackSize;

tcbTask->dataarea = tcbStack + tcbStackSize;

tcbTask->saveusp = tcbTask->dataarea;

//Set the initial SR.

tcbTask->savesr = ROSA\_INITIALSR;

//Setting our custom values

tcbTask->originalpriority = taskPriority;

tcbTask->priority = taskPriority;

for(int i =0; i<semaCount: i++)

{

ROSA\_SemaphoreRegister(semaphores[i], tcbTask);

}

/\* after registering Semaphore we will install the task \*/

tcb \* tcbTmp;

/\* Is this the first tcb installed? \*/

if(TCBLIST == NULL) {

TCBLIST = tcbTask;

TCBLIST->nexttcb = tcbTask; //Install the first tcb

tcbTask->nexttcb = TCBLIST; //Make the list circular

}

else {

tcbTmp = TCBLIST; //Find last tcb in the list

while(tcbTmp->nexttcb != TCBLIST) {

tcbTmp = tcbTmp->nexttcb;

}

tcbTmp->nexttcb = tcbTask; //Install tcb last in the list

tcbTask->nexttcb = TCBLIST; //Make the list circular

}

//Initialize context.

contextInit(tcbTask);

}

void ROSA\_tcbDelete(tcbHandle \*tcbTask, void \*semaphores, int semaCount)

{

int result = 0;

for(int i =0; i<semaCount: i++)

{

result = ROSA\_semaphoreGive(semaphores[i]);

if(result == 0)

{

ROSA\_SemaphoreUnregister(semaphores[i], tcbTask);

//Set the task function start and return address.

tcbTask->staddr = NULL;

tcbTask->retaddr = NULL;

//Set up the stack.

tcbTask->datasize = 0;

tcbTask->dataarea = 0;

}

else

{

//return error unable to delete task.

}

}

}

void ROSA\_tcbResume(void \*tcbTask)

{

ROSA\_prvRemoveFromWaitingQueue(tcbTask);

ROSA\_prvAddToReadyQueue(tcbTask);

}

void ROSA\_tcbSuspend(void \*tcbTask)

{

ROSA\_prvRemoveFromReadyQueue(tcbTask);

ROSA\_prvAddToWaitingQueue(tcbTask);

}