

doc_thread.cpp

```
int Thread::StopThread(
    void
)
{
    int    nStatus;
    int    nReturnStatus;
    nStatus = 0;
    nReturnStatus = RT_S_SUCCESS;
    nStatus = pthread_kill( this->nThreadID , SIGTERM );
    switch( nStatus )
    {
        case 0:
            nReturnStatus = RT_S_SUCCESS;
            break;
        case ESRCH:
            nReturnStatus = RT_E_THREAD_NOT_FOUND;
            break;
        default:
            nReturnStatus = RT_E_ERROR;
            break;
    }
    return nReturnStatus;
}

int Thread::Join(
    void
)
{
    int    nStatus;
    int    nReturnStatus;
    void * pThreadStatus;
    nStatus = 0;
    nReturnStatus = RT_S_SUCCESS;
    pThreadStatus = NULL;
    nStatus = pthread_join( this->nThreadID , &pThreadStatus );
    switch( nStatus )
    {
        case 0:
            nReturnStatus = RT_S_SUCCESS;
            break;
        case ESRCH:
            nReturnStatus = RT_E_THREAD_NOT_FOUND;
            break;
        case EDEADLK:
            nReturnStatus = RT_E_THREAD_DEADLOCK;
            break;
        default:
            nReturnStatus = RT_E_ERROR;
            break;
    }
    return nReturnStatus;
}

int Thread::DeleteCondition( void )
{
    int nStatus;
    nStatus = 0;
    nStatus = pthread_mutex_destroy( &thread_cond_mutex );
    nStatus = pthread_cond_destroy( &thread_condition );
    return RT_S_SUCCESS;
}

int Thread::SetPolicy(
    int    _nSetPriority,
    int    _nSetPolicy
)
{
    int    nStatus;
    nStatus = 0;
    this->oParam.sched_priority = _nSetPriority;
    this->nPolicy = _nSetPolicy;
    if( this->bStartedFlg == false )
    {
        return RT_S_SUCCESS;
    }
    if( this->bStopFlg == true )
    {
        return RT_S_SUCCESS;
    }
    nStatus = pthread_setschedparam(
        this->nThreadID,
        this->nPolicy,
        &this->oParam
    );
    if( nStatus != 0 )
}
```