

```

doc_thread.cpp

friend void* _run_thread( void *_pObject );
virtual int Join( void );
bool bStartedFlg;
bool bStopFlg;
pthread_t nThreadID;
};
#endif
---Thread.cpp
#include "Thread.h"
#include <string.h>
pthread_mutex_t Thread::thread_cond_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t Thread::thread_condition = PTHREAD_COND_INITIALIZER;
extern "C" void* _run_thread(
    void* _pObject
)
{
    if( _pObject == NULL ){
        return (void*)(RT_E_ERROR);
    }
    Thread *pThread = reinterpret_cast< Thread * >( _pObject );
    return pThread->RunThread();
}

Thread::Thread(
    int _nSetPriority,
    int _nSetPolicy,
    int _nSetScope
)
{
    this->bStartedFlg = false;
    this->bStopFlg = false;
    this->nThreadID = 0;
    this->nPolicy = 0;
    this->nScope = 0;
    memset( &this->oParam, 0x00, sizeof( this->oParam ) );
    this->oParam.sched_priority = _nSetPriority;
    this->nPolicy = _nSetPolicy;
    this->nScope = _nSetScope;
    return;
}

int Thread::StartThread(
    void
)
{
    int nResult;
    pthread_attr_t oAttr;
    nResult = 0;
    memset( &oAttr, 0x00, sizeof( oAttr ) );
    pthread_attr_init( &oAttr );
    nResult = pthread_attr_setschedparam ( &oAttr , &this->oParam );
    if( nResult != 0 )
    {
        return RT_E_ERROR;
    }
    nResult = pthread_attr_setschedpolicy( &oAttr, this->nPolicy );
    if( nResult != 0 )
    {
        return RT_E_ERROR;
    }
    nResult = pthread_attr_setscope( &oAttr , this->nScope );
    if( nResult != 0 )
    {
        return RT_E_ERROR;
    }
    if( this->bStartedFlg )
    {
        return RT_E_THREAD_ALREADY_ACTIVE;
    }
    this->nThreadID = 0;
    nResult = pthread_create( &(this->nThreadID), &oAttr, _run_thread, this );
    if( nResult != 0 ){
        return RT_E_ERROR;
    }
    bStartedFlg = true;
    return RT_S_SUCCESS;
}

int Thread::StopLoop(
    void
)
{
    this->bStopFlg = true;
    return RT_S_SUCCESS;
}

```