

doc_thread.cpp

```
{
    return RT_E_ERROR ;
}
return RT_S_SUCCESS;
}
#include "Thread.h"
#include "dlj_rt_data.h"
class JoinThread : public Thread
{
private:
protected:
    virtual void * RunThread( void );
public:
    JoinThread();
    virtual ~JoinThread( void );
    virtual int Init( void );
    virtual int JoinThreads();
};
#include "JoinThread.h"
#include <string.h>
JoinThread::JoinThread(
    void
) : Thread( THREAD_PRIORITY_MID , THREAD_POLICY_DEF )
{
}
JoinThread::~JoinThread(
    void
)
{
    void * pStatus;
    pStatus = NULL;
    pthread_cond_broadcast( &(amp; Thread::thread_condition ) );
    if( this->nThreadID != 0 )
    {
        pthread_join( this->nThreadID , &pStatus );
    }
}
int JoinThread::Init(
    void
)
{
    int nStatus;
    nStatus = 0;
    nStatus = this->StartThread();
    ERR_RETURN( nStatus );
    return RT_S_SUCCESS;
}
void * JoinThread::RunThread(
    void
)
{
    int nStatus;
    int nReturnStatus;
    nStatus = 0;
    nReturnStatus = RT_S_SUCCESS;
    nStatus = pthread_cond_wait(
        &(Thread::thread_condition) ,
        &(Thread::thread_cond_mutex)
    );
    if( nStatus != 0 )
    {
        nReturnStatus = RT_E_ERROR;
    }
    return (void *) ( nReturnStatus );
}
int JoinThread::JoinThreads(
    void
)
{
    int nStatus;
    nStatus = 0;
    nStatus = this->Join();
    this->nThreadID = 0;
    return RT_S_SUCCESS;
}
---GThread.h, GThread.cpp
#ifdef _GTHREAD_
#define _GTHREAD_
#include <pthread.h>
#include <sched.h>
#include "rt_sts.h"
#include "GThreadBase.h"
```