

doc.cpp

```
#ifndef _DataQue
#define _DataQue
#include "rt_sts.h"
#include "Deque.h"
#include <pthread.h>
#include <sys/time.h>
#include <errno.h>
#define QUE_UNINIT 0
#define QUE_INITED 1
#define QUE_DEFAULT_MAX_SIZE 50
#define QUE_DEFAULT_WAIT_TIME 0
#define QUE_TIMEVAL_U_N 1000
template <class T>
class DataQue
{
protected:
    Deque<T> oQue ;
    pthread_mutex_t mutex ;
    pthread_cond_t PushCond ;
    pthread_cond_t PopCond ;
    int nMaxQue ;
    int nTimeout ;
    int nInitStatus ;
    virtual int Lock ( void ){
        return ( ( pthread_mutex_lock ( &this->mutex ) == 0 ) ?
            RT_S_SUCCESS : RT_E_LOCK );
    }
    virtual int Unlock ( void ){
        return ( ( pthread_mutex_unlock ( &this->mutex ) == 0 ) ?
            RT_S_SUCCESS : RT_E_UNLOCK );
    }
    virtual int Wait( pthread_cond_t *pCond );
    virtual int TimedWait( pthread_cond_t *pCond );
public:
    DataQue( int _nMaxQue = QUE_DEFAULT_MAX_SIZE,
            int _nMaxTimeout = QUE_DEFAULT_WAIT_TIME );
    DataQue( const DataQue<T> & _oObj );
    virtual ~DataQue( void );
    virtual int Init ( void );
    virtual int Push ( T _oData );
    virtual int PushFirst( T _oData );
    virtual int Pop ( T & _pData );
    virtual int Size ( unsigned int & _nSize );
    virtual int Empty ( void );
    virtual int Clear ( void );
    virtual DataQue<T> & operator = ( DataQue<T> &_oObj );
};

template <class T> DataQue<T>::DataQue(
    int _nMaxQue,
    int _nMaxTimeout
) : oQue( _nMaxQue, 1 )
{
    this->nTimeout = 0;
    this->nInitStatus = QUE_UNINIT;
    this->nMaxQue = _nMaxQue;
    if ( _nMaxTimeout > 0 )
    {
        nTimeout = _nMaxTimeout;
    }
}

template<class T> DataQue<T>::DataQue(
    const DataQue<T> & _oObj
){
    this->nTimeout = 0;
    this->nInitStatus = QUE_UNINIT;
    this->oQue = _oObj.oQue;
    this->nMaxQue = _oObj.nMaxQue;
    this->nTimeout = _oObj.nTimeout;
    pthread_mutex_init ( &( this->mutex ), NULL );
    pthread_cond_init ( &( this->PushCond ), NULL );
    pthread_cond_init ( &( this->PopCond ), NULL );
    this->nInitStatus = QUE_INITED;
}

template <class T> DataQue<T>::~DataQue(
    void
){
    int nStatus ;
    nStatus = 0;
    nStatus = pthread_mutex_destroy( &(this->mutex) );
    nStatus = pthread_cond_destroy( &(this->PushCond) );
    nStatus = pthread_cond_destroy( &(this->PopCond) );
}
```