

```

while ( ( nQueSize = this->oQue.GetElementSize() ) <= 0 )
{
    if ( this->nTimeout== 0 )
    {
        nStatus = this->Wait( &this->PushCond );
    }
    else if( this->nTimeout > 0 )
    {
        nStatus = this->TimedWait( &this->PushCond );
    }
    if( nStatus != RT_S_SUCCESS )
    {
        this->Unlock();
        return nStatus;
    }
}
nStatus = this->oQue.PopBack( _pData );
if ( ERR_CHECK( nStatus ) )
{
    this->Unlock();
    return nStatus;
}
nStatus = this->Unlock();
ERR_RETURN(nStatus);
nStatus = pthread_cond_broadcast( &( this->PopCond ) );
if ( nStatus != 0 )
{
    return RT_E_INVALID;
}
return RT_S_SUCCESS;
}

template <class T> int DataQue<T>::Size (
    unsigned int & _nSize
){
    int nStatus;
    int nElementSize;
    nStatus      = 0;
    nElementSize = 0;
    _nSize       = 0;
    if ( this->nInitStatus != QUE_INITED )
    {
        return RT_E_INITIALIZE;
    }
    nStatus = this->Lock();
    ERR_RETURN(nStatus);
    nElementSize = this->oQue.GetElementSize();
    if ( nElementSize > 0 )
    {
        _nSize = nElementSize;
    }
    nStatus = this->Unlock();
    ERR_RETURN(nStatus);
    return RT_S_SUCCESS;
}

template <class T> int DataQue<T>::Empty (
    void
){
    int nStatus;
    int nSize;
    nStatus      = 0;
    nSize        = 0;
    if ( this->nInitStatus != QUE_INITED )
    {
        return RT_E_INITIALIZE;
    }
    nStatus = this->Lock();
    ERR_RETURN(nStatus);
    nSize = this->oQue.GetElementSize();
    nStatus = this->Unlock();
    ERR_RETURN(nStatus);
    if ( nSize <= 0 )
    {
        return RT_E_INVALID;
    }
    return RT_S_SUCCESS;
}

template <class T> int DataQue<T>::Clear (
    void
){
    int nStatus;
    nStatus = 0;
    if ( this->nInitStatus != QUE_INITED )

```