

```

        case   THREAD_STOP_NOMAL:
            nStatus = this->pObj->StopLoop();
            ERR_RETURN( nStatus );
            break;
        case   THREAD_STOP_JOIN:
            nStatus = this->pObj->StopLoop();
            ERR_RETURN( nStatus );
            nStatus = this->Join();
            ERR_RETURN( nStatus );
            break;
        case   THREAD_STOP_TERM:
            nStatus = this->KillThread( SIGTERM );
            ERR_RETURN( nStatus );
            this->pObj->ClearFlag();
            break;
        case   THREAD_STOP_KILL:
            nStatus = this->KillThread( SIGKILL );
            ERR_RETURN( nStatus );
            this->pObj->ClearFlag();
            break;
        default:
            return RT_E_INVALID;
    }
    return RT_S_SUCCESS;
}

int GThread::KillThread(
    int    nSignal
)
{
    int nStatus;
    int nResult;
    nStatus = 0;
    nResult = 0;
    nResult = pthread_kill( this->nThreadID , nSignal );
    switch( nResult )
    {
        case   0:
            nStatus = RT_S_SUCCESS;
            break;
        case   ESRCH:
            nStatus = RT_E_THREAD_NOT_FOUND;
            break;
        default:
            nStatus = RT_E_ERROR;
            break;
    }
    return nStatus;
}

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

int GThread::CheckThread(
    void

```