监控模块说明：

机制：

被监控进程通过消息队列周期性向监控进程进行发送注册消息。

监控进程通过监测注册消息来确认被监控进程是否工作正常。

实现：

监控进程由3部分组成：

一个monitor\_rcv\_queue\_msg\_sevice线程;

一个200ms的定时器monitor\_200ms\_process;

一个“主线程main”中的空循环。

monitor\_rcv\_queue\_msg\_sevice线程：建立以0X5AA5为key的消息队列，通过此队列接收被监控进程发送过来的注册消息，注册消息的格式为：

typedef struct

{

UCHAR msgSrc; //消息来源

pthread\_t pidNum; //pid号

}

MONITOR\_MSG\_PAYLOAD;

200ms的定时器monitor\_200ms\_process：监测各个被监控进程，当注册消息没有被监控进程收到时，判定被监控进程异常，根据全局进程数组g\_MonitorAppInfoTable配置的参数，进行被监控进程拉起，或者系统复位操作。

g\_MonitorAppInfoTable，用于定义被监控进程的配置参数以及运行时状态，数组元素为pMONITOR\_APP\_INFO类型。

typedef struct monitor\_app\_info

{

char procFlag; /\* 处理标志, see monitor\_proc\_flag \*/

UINT32 checkInterval; /\* 检测间隔毫秒 \*/

UINT32 curCheckInterval; /\* 当前检测剩余间隔毫秒\*/

const char \*pAppName; /\* 应用名称 \*/

UINT32 msgSrc; /\* 应用监控消息源编号 \*/

const char \*pRestartCmd; /\* 进程重启命令 \*/

const char \*pRebootCmd; /\* 系统重启命令 \*/

UINT32 appRegCount; /\* 任务注册次数\*/

UINT32 lastAppRegCount; /\* 上次任务注册次数\*/

UINT32 restartAppCount; /\* 主动重启任务的次数，即监控程序重启任

务的次数\*/

pthread\_t pid; /\* 任务PID\*/

UINT32 appIdChangeTimes; /\* 任务ID变化的次数\*/

RESET\_TIME resetTimeTable; /\* 重启时间记录表\*/

} MONITOR\_APP\_INFO, \*pMONITOR\_APP\_INFO;

当30分钟内有进程出现5次拉起操作时，判定系统异常，系统复位。

monitor\_200ms\_process定时器还有一个功能是喂硬件看门狗，在BCR&BSDR上，硬件看门狗的周期为5秒，如果monitor\_200ms\_process定时器异常（监控进程功能简单，理论上不会出现异常，如果出现异常可以判定为系统异常），导致5秒内没有喂硬件看门狗，硬件会自动复位。

特别说明： mt7621的硬件看门狗 ,由openwrt系统下ubus接管，就不再使用monitor\_200ms\_process喂硬件看门狗，mt7621的硬件看门狗的周期是30秒，ubus每5秒喂一次。

“主线程main”中的空循环：

while(1)

{

taskDelay(100 \* 1000);

}

监控模块以动态链接库的形式提供，动态链接库名为：libtaskLib.so，提供头文件有：common.h,errCode.h,taskLib.h，outputLib.h

使用：

被监控进程需要调用void\* taskWatch(void\* pArg)函数，此函数是一个死循环函数，以10秒为周期向监控进程发送注册消息。

举例如下：

#include "common.h"

#include "errCode.h"

#include "taskLib.h"

int main()

{

//主线程进行线程监控

UCHAR msgSrc = MSG\_SRC\_SAMPLE;

taskWatch(&msgSrc);

return 0;

}

其中UCHAR msgSrc = MSG\_SRC\_SAMPLE;是进程标识，需要在common.h中统一定义。

/\*模块的定义\*/

enum em\_ap\_module

{

MIN\_MOD\_NUM = 0,

SAMPLE\_MOD\_NUM,

SYSMONITOR\_MOD\_NUM,

MAX\_MOD\_NUM

};