# 一种基于UMTS核心网的定时器实现方案

郜 林 王 荃
(天津财经大学理工学院电子信息科学系 天津 300222)

**摘 要** 针对UMTS核心网呼叫连接协议的实现,提出了一种软件定时器的实现方案。该方案以WinCE实时嵌入式系统作为开发环境,采用单链表队列和相对时间项方式的定时器数据结构,基于时间相对算法建立了定时器单一线程和插入、删除程序,以实现协议所规定的多定时器逻辑功能。在嵌入式系统编程过程中,运用互斥量机制完成定时器线程与插入、删除程序的通信同步;运用堆管理机制完成系统内存的动态分配与释放。对该方案的主要技术和实现结构作了论述,并给出了关键代码;还对UMTS协议结构和WinCE相关机制作了分析。实验表明,该设计方案具有编程效率高、实时性能好和系统开销小的特点,适合基于嵌入式系统的通信协议定时器编程。

**关键词** 定时器,链表队列,相对时间,核心网,WinCE
**中图法分类号** TP311.1,TP316.2 **文献标识码** B

## Timer Realization Based on UMTS Core Network

GAO Lin WANG Quan
(School of Electronic Information Engineering, Tianjin Economic and Financial University, Tianjin 300222, China)

**Abstract** To implement the call connection protocol of UMTS core network, a timer design was proposed by the way of software, developed and researched in the environment of WinCE real-time embedded system. Based on relative time algorithm proposed, the timer program, which data structure is constructed by single linked list and relative time item, creates a single time thread and insertion/deletion functions to realize the multiple-timer logic regulated by UMTS protocol. In the embedded system programming, we applied mutex method to implement the synchronization in communication between the timer thread and insertion/deletion functions, and adopted heap management method to realize the dynamic allocation and release of memory. The paper discussed the technique and the structure of the plan with key code and analysed the relative WinCE mechanisms and UMTS protocols. Experiments prove that the design, which has the qualities of low hardware occupation and high program efficiency, with good real time ability, is suitable for timer programming in communication protocols based on embedded system.

**Keywords** Timer, Link queue, Relative time, Core network, WinCE

实时时延处理是通信协议的基本组成部分,直接体现协议时序关系;定时又是实时时延处理的核心内容,往往要通过多个定时器之间交互作用得以实现。因而,如何在准确反映协议内容的前提下,提高定时器的实现效率,是通信协议开发的一个重要课题。

由于硬件平台提供的定时器个数很少(一般只有几个),对于通信协议而言,定时器主要采用软件方式实现。在软件实现方式中,定时器设计又可分为由操作系统提供和由应用程序提供两种方法[1]。前者优点是定时精度高,但所提供的定时必须与中断相关联,多定时器的作用会导致大量的中断处理,必然会降低系统的实时性能。后者虽然定时精度不如前者,但对于多定时器系统而言,可通过合理设计,获得好于前者的实时性能。

目前,在定时器应用程序设计中,主要设计思路如下:从定时数据结构构建上看,主要采用静态数组和绝对时间项方法。其具有逻辑简单的优点;但当硬件定时器中断发生时,要对所有定时器节点进行减法操作,时间开销很大,且时延不确定(与定时器数目相关)[2]。从多定时器交互关系上看,主要采用多线程方式,不同定时器创建不同线程,分别作超时及其他相关任务处理。其优点是编程思路清晰、易于实现;但对于复杂定时系统来说,又不可避免地给系统带来巨大的线程调度开销。

本文就第三代移动通信的UMTS核心网的呼叫连接协议,基于WinCE实时性嵌入式操作系统,提出了一种采用时间链表和定时器相对算法的单一线程的定时器应用程序实现方案。

## 1 UMTS核心网协议分析

通用移动通信系统(UMTS)的协议结构,从功能方面可分为接入层和非接入层两大部分。其中,接入层是指UE和UTRAN间的无线接口协议集、UTRAN和核心网(CN)间的接口协议集;非接入层是指UE和CN之间的核心网协议。

郜 林(1971-),男,博士生,主要研究方向为移动通信关键技术、通信系统的开发与实现,E-mail:gavingao71@sohu.com;王 荃(1971-),男,硕士,讲师,主要研究方向为嵌入式系统。

接入层协议可划分为物理层、数据链路层和网络层。L1是物理层(PHY),物理层的主要功能是传输信道的编解码、物理信道扩频、调制/解调与解扩、同步、闭环功率控制。L2是链路层,又可划分为多个子层。其中,子层MAC和RLC共同为高层信令和用户数据提供承载。子层PDCP对冗余的网络层PDU控制信息在发送端进行压缩,接收端进行解压缩,同时也对TCP/IP头进行压缩和解压缩。子层广播/组播控制协议BMC用于在无线接口上传递由小区广播中心产生的消息。PDCP和BMC只在PS域场合有效。L3是网络层,其中,RRC通过业务接入点(通用控制Gc、通告Nt和专用控制DC业务接入点)向上层提供业务。

非接入层按支持模式划分为UMTS的非接入层和GSM的非接入层两种;按支持的服务类型,UMTS系统的非接入层可以划分为电路交换CS和分组交换PS两个域;按协议层次可划分为连接管理(CM)和移动性管理(MM)两个子层。MM层包括CS的MM与PS的GMM,主要负责两种服务模式下的移动管理与注册服务。连接管理CM可包含多个子层。其中,CS的CC与PS的SM,主要负责两种服务模式下的连接管理;SS作为电信业务的补充,基本功能是更改或增强电信业务(为非独立层,必须依附于其他业务而存在);GSMS提供基于PS域的短信服务[3]。

本文定时器方案是基于UMTS呼叫连接(CC)协议而提出的,该协议在非接入层中的结构关系,如图1所示。

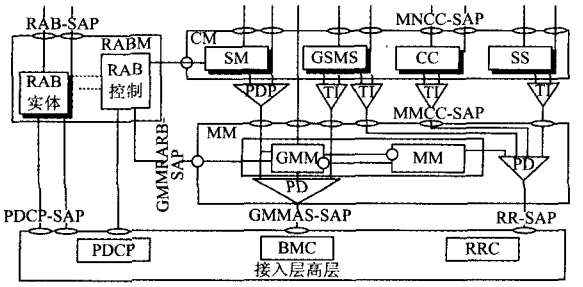


图1 支持PS/CS的UMTS非接入层模型

根据R99协议,UMTS CC事件分为与时间无关和有关两类,其中后者需要通过定时器机制来实现定时逻辑功能。R99协议对各定时器所需要实现的功能作了明确的规定[4],以定时器T301为例,其是在接收呼叫状态下受到警告信息触发后启动的,其超时时段是180秒。T301在超时时段范围内,若接收到连接信息,则停止定时(正常情况);否则,清除本次呼叫(超时情况)。

## 2 定时器总体设计

本方案是在TI的OMAP[5]硬件开发平台下,以WinCE嵌入式系统为开发环境作研究和开发的。在定时器程序设计中,应用了WinCE嵌入式系统的同步互斥量和内存管理机制。

### 2.1 定时机制的自定义数据结构

为了用软件方法高效地实现定时器机制,首先要建立一个合理的数据结构。而合理数据结构的构建是建立在对定时机制分析、研究基础之上的。

定时机制是对定时事件的处理机制,而该机制的中心环节是建立一个定时队列,并根据过程的需要将事件加入到队列中去,或将事件从队列中删除,以达到管理队列的目的。为此,本方案在定时事件的数据结构中,设定事件本身项和时间项(可由指针实现)两项基本内容。

对于时间项的设计可分为绝对时间和相对时间两种方式。若采用前种方式,每一事件的时间都需要单独计算,并进行独立的超时处理。而采用后者方式,所有时间都是关联的,亦即都是相对于前一事件的相对时间。在相对时间的时间项设计方法中,定时事件可以直观地按照时间顺序保存,每一个事件通过指针指向下一个事件。定时器只需要定期地递减表中第一项的时间值,当第一项的时间值为零时,事件发生,定时进程将该表项从表中删除,然后,原先的第二项成为第一项,以此类推。通过以上分析可知,相对时间计时方式具有实现简单,程序高效的优点。

本文所提方案采用了相对时间链表的设计思路,通过单链表数据结构,实现定时事件队列[6,7],如图2所示。

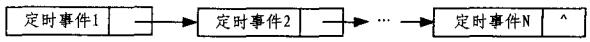


图2 链表队列实现定时机制

链表表项tqent的自定义结构,如下所示。

```
struct tqent {
  int tq_timeleft;/ * time to expire * /
  DWORD tq_time;
/ * time this entry was queued. * /
  HANDLE  tq_port;
/ * port to send the event.    * /
  DWORD  tq_portlen;
/ * length of "tq_port" * /
  NOTIFICATION tq_msg;
/ * data to send when expired. * /
  structtqent * tq_next;
  / * next in the list * /
};
```

### 2.2 定时器线程

本文所提方案采用了定时相对算法,通过单一的定时器线程实现多定时器之间的逻辑关系。定时相对算法可完成对多定时器的超时判断,并作相应的任务处理,具体算法如下:

若表项的剩余时间<delta,则判断为事件超时,将事件的消息写入到指定的消息队列,并将该事件从列表中删除。若表项的剩余时间>delta,表项的剩余时间=表项的剩余时间-delta,并继续循环。若删除事件后,链表为空,则停止执行;否则,继续循环。

变量delta算法如下:

若链表为空,delta=休眠时间+等待互斥量时间;反之,delta=等待互斥量时间。若事件超时,delta=delta-事件剩余时间。

在定时器线程设计中,本文应用了WinCE同步机制,创建了互斥量。WinCE作为嵌入式实时操作系统为程序设计提供了简单、实用的同步机制。与Windows操作系统相比较,WinCE未提供诸如消息、端口、监督程序之类的复杂机制;而是提供了等待队列机制,并且基于该项机制,定义了同步对象和等待函数。在WinCE中,同步对象主要包括事件(event)、互斥量(mutex)和信号量(semaphore)。其中,互斥量运行于内核模式。由于内核对象具有全局性,不同进程都可以访问。在一个线程访问某个共享资源时,互斥量可以保持该共享资源不被其他线程所访问(可以支持不同进程中的线程互斥访问一个共享资源)。WinCE所提供的互斥量API函数主要有建立函数、释放函数和等待函数等[8]。

(1)建立函数

```
HANDLE CreateMutex(
        LPSECURITY_ATTRIBUTES lpMutexAttributes, BOOL bInitialOwner,LPCTSTR lpName);
```

参数 lpMutexAttributes(输入参数)必须设置为零。参数 bInitialOwner(输入参数)为布尔型变量参数,如果设置成 TRUE,表示当前线程将占有互斥资源,互斥对象的线程 ID 设置成当前线程 ID,递归计数被设置为 1,互斥对象处于无信号状态;如果设置成 FALSE,表示当前进程并不占有互斥资源,互斥对象的线程 ID 和递归计数都被设置为零,互斥对象处于有信号状态。参数 lpName(输入参数)为一个长指针,该指针指向一个以零结尾的指定互斥量对象名称的字符串。如果 lpName 与一个现已存在的互斥量名称相匹配,则参数 bInitialOwner 由于已经被设定而被忽略。如果 lpName 为零值,则产生一个未命名的互斥量。

如果该函数成功,则返回一个指向所创建互斥量的句柄;否则,返回零值。

(2)释放函数

```
BOOL ReleaseMutex(HANDLE hMutex);
```

参数 hMutex(输入参数)为一个指向所要释放互斥量的句柄,该句柄由 CreateMutex 函数返回。

返回值为布尔型变量:返回非零值,指示函数成功;反之,指示函数失败。

(3)等待函数

WinCE 提供了四种等待函数,此处仅介绍与本文实现方案相关的 WaitForSingleObject 函数。

```
DWORD WaitForSingleObject(HANDLE hHandle,
        DWORD dwMilliseconds);
```

参数 hHandle(输入参数)为对象的句柄。参数 dwMilliseconds(输入参数)为等待时间,单位为毫秒。如果等待时间设定为零,该函数测试对象的状态,并立即返回;如果等待时间设定为无限值,则该函数永远不会超时。

WaitForSingleObject 函数返回值如下所示:

1)如果函数成功,返回值指示引起函数返回的事件。若返回 WAIT_OBJECT_0,指定对象的状态为有信号状态;若返回 WAIT_TIMEOUT,指示进入超时,并指定对象的状态为无信号状态。

2)如果函数失败,返回 WAIT_FAILED。

本方案定时器线程通过创建互斥量,同步插入与删除函数,共同管理和处理时间事件链表。此外,为了节约系统资源,定时器线程还调用了 WinCE 线程挂起函数;为了提高程序的健壮性能,做了必要的错误处理。定时器线程流程图如图 3 所示。

定时器线程关键代码如下所示:

```
DWORD WINAPI wcdmatimer(LPVOID lppara)
{
  DWORD now,lastrun;
/* times from system clock */
  int delta;
/* time since last iteration */
  struct tqent *tq;
  CurMSec=GetTickCount();
  Lastrun=CurMSec;
    …… ……
  tqmutex=CreateMutex(NULL,FALSE,(TEXT("wcdmatimerMutex")));
    …… ……
  while (TRUE)
  {
    if (tqhead == NULL)
    {
      printf("Wcdmatimer SuspendThread!");
      SuspendThread(tqpid);
      CurMSec=GetTickCount();
      lastrun=CurMSec;
      Sleep(TIMERGRAN);/* real-time delay */
}
    WaitForSingleObject(tqmutex,INFINITE);
    CurMSec=GetTickCount();
    now=CurMSec;
    delta=now-lastrun;
    if (delta<0||delta>TIMERGRAN*10)
      delta=TIMERGRAN;
    lastrun=now;
    while (tqhead! =0&&tqhead->tq_timeleft<= delta)
      {
      delta -= tqhead->tq_timeleft;
        …… ……
      tq=tqhead;
      tqhead=tqhead->tq_next;
      LocalFree(tq);
      }
    if (tqhead)
      tqhead->tq_timeleft -=delta;
     ReleaseMutex(tqmutex);
    }
    return OK;
}
```

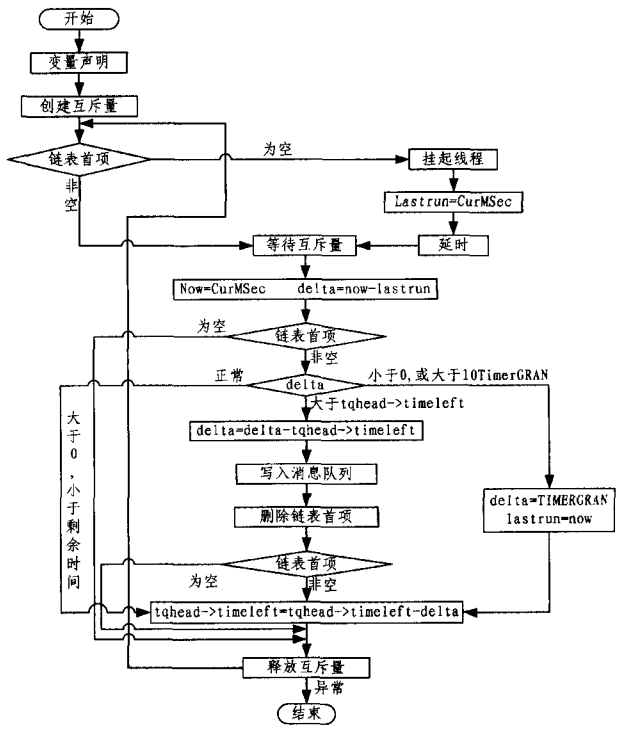


图 3 定时器线程流程图

## 3 定时器的关键模块

### 3.1 定时器插入程序

定时器插入程序可被其他线程所调用，其负责创建一个定时器事件，并将该事件插入到定时器链表队列中。插入程序的入口参数包括发送事件端口、端口的最大事件数、发送的信息和定时器时延。

定时器插入程序应用了 WinCE 内存管理机制，动态分配内存。Windows CE. NET 具有三层内存逻辑结构，即物理内存、虚拟内存和逻辑内存。与内存逻辑结构相对应，内存管理也可以分为三个部分，即物理页面管理、虚存管理、堆管理。本文所提出的方案采用了堆管理方式，可在进程内部动态释放和回收内存资源。堆管理的基本单位是堆(heap)，其粒度远小于页的粒度。在 WinCE 操作系统中，系统对堆的管理只允许申请固定大小的块。堆申请完成后，系统会根据应用程序的需要自动增加堆的大小；当释放堆的时候，堆的消减由系统自动完成。堆管理 API 函数主要包括堆创建函数、堆分配函数、堆释放函数等。

其中堆分配函数负责在堆中分配一个内存块，该函数的函数原型如下：

```
LPVOID HeapAlloc(HANDLE hHeap,
                 DWORD dwFlags,
                 DWORD dwBytes);
```

参数 hHeap 为输入参数，其指向将被分配内存块的堆句柄(由 HeapCreate 函数返回)。参数 dwFlags 为输入参数，可选择参数 HEAP_NO_SERIALIZE 或 HEAP_ZERO_MEMORY；所指定参数将覆盖相应的 HeapCreate 函数参数。参数 dwBytes 为输入参数，其为被分配的字节数。

当函数分配内存块成功时，返回一个指向所分配内存块的指针；否则，返回零值。

定时器插入程序首先通过 LocalAlloc 函数调用堆分配函数，为插入事件分配空闲存储区。然后，根据入口参数填充新建 tqent 结构字段，并进入等待互斥量阶段。在互斥量到达之后，将该事件插入到定时器链表队列中。最后，释放互斥量，并返回一个 int 值。

定时器插入程序的关键代码如下。

```
int tmset ( HANDLE port, DWORD portlen, PNOTIFICATION
    pmsg,int time)
{
  struct tqent * ptq, * newtq, * tq;
  newtq=LocalAlloc(LPTR,sizeof(struct tqent));
  newtq->tq_timeleft=time;
  CurMSec=GetTickCount();
  newtq->tq_time=CurMSec;
      … …
  (void) tmclear(port,pmsg);    WaitForSingleObject(tqmutex,IN-
  FINITE);
  if (tqhead == NULL)
  {
    tqhead=newtq;
    ResumeThread(tqpid);
    ReleaseMutex(tqmutex);
    return OK;
  }
  for (ptq=0,tq=tqhead;tq;tq=tq->tq_next)
  {
    … …
  }
    newtq->tq_next=tq;
    … …
    ReleaseMutex(tqmutex);
    return OK;
}
```

### 3.2 定时器删除程序

定时器删除程序的任务是从定时器链表队列中删除特定的定时器事件。删除程序的入口参数包括所要删除定时器事件的发送端口和发送消息。

定时器删除程序，首先等待互斥量，然后开始搜索链表队列，直到找到与参数相匹配的表项。当搜索到所要删除表项之后，将该定时事件从链表队列中删除。然后，通过 LocalFree 函数调用 WinCE 堆释放函数，将存储区释放回系统。最后，释放互斥量。本函数返回一个描述所删除定时事件(在链表中)存在时间的 int 型返回值。

WinCE 堆释放函数原形如下：

```
BOOL HeapFree(HANDLE hHeap,
              DWORD dwFlags,
              LPVOID lpMem);
```

参数 hHeap 为输入参数，其指向存储块被释放的堆句柄(由 HeapCreate 函数返回)。参数 dwFlags 为输入参数，指定参数为 HEAP_NO_SERIALIZE 0x00000001，可覆盖 HeapCreate 函数的 flOptions。参数 lpMem 为输入参数，其为指向存储块的指针，该函数是由 HeapAlloc 函数返回的。

如果函数成功，返回非零值；相反，则返回零值。

定时器删除程序关键代码如下。

```
int tmclear(HANDLE port,PNOTIFICATION pmsg)
{
  struct tqent * prev, * ptq;
  int timespent; WaitForSingleObject(tqmutex,INFINITE);
  prev=0;
  for (ptq=tqhead;ptq! = NULL;ptq=ptq->tq_next)
  {
  if(ptq->tq_port==port&&(ptq->tq_msg).m_msgcause==
    pmsg->m_msgcause&&(ptq->tq_msg).m_message==pmsg-
    >m_message&&(ptq->tq_msg).m_lpGlobalPass==pmsg->
    m_lpGlobalPass)
      {
        CurMSec=GetTickCount();
        timespent= CurMSec - ptq->tq_time;
          … …
        ReleaseMutex(tqmutex);
        LocalFree(ptq);
        return timespent;
      }
      prev=ptq;
  }
  ReleaseMutex(tqmutex);
  return ERROR;
}
```

(下转第 415 页)

智能终端具有五大功能：发电信息、业务办理、用电信息、用电预测、信息公告。其中在发电信息功能中包括：已发电量、蓄电池电量、发电功率、向电网中输出电量、净计量电价、预计发电量。业务办理功能中包括：增值业务、缴费业务、故障报修业务，其中增值业务包括：社区业务、医疗业务、订购消费、家政服务。用电信息功能中包括：家用电器用电信息（用电量、用电时间、功率）、查看用户电能质量、历史用电数据及合理用电策略。用电预测功能主要包括：查看本地用电负荷曲线、预测用电峰谷曲线、预测本地用电量及用户用电量、预测实时电价。信息公告功能包括：发布停电、电网维护通知，同时用户还可向电网公司提出投诉、举报或建议。在该框架中，用户还可看到室内的温度、湿度、三表数据、天气情况和电网公司发布的通知信息。

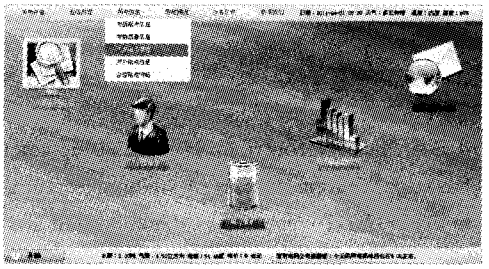

图 6　智能显示终端功能模拟界面

**结束语**　本文主要从用户用电方面入手，通过对智能小区数据模型的需求分析，根据智能小区中的用电设备及其之间的关系，构建了智能小区用电的数据模型，该模型是现在及未来智能小区建设的基础，并为我国智能小区的构建提供了理论支持。

在智能小区用电数据模型中，本文只是对用户的主要用电设备进行了模型的构建，并没有考虑其他的用电设备，如小区安防系统、信息管理子系统等一些系统中的用电设备。除此之外，在智能小区构建的过程中，还存在许多问题，首先是小区用电数据的安全传输问题，这是网络本身的问题，是所有建立在网络基础上的系统所面临的问题；其次是小区用电数据传输的实时性，能否保证数据传输的可靠性、安全性和实时性，也是当今网络面临的一大问题；最后是电网中心能否找到最合理的算法来准确预测小区的用电量，从而确保智能电网供需平衡，也是我们面临的一次重大挑战。

## 参 考 文 献


[1] 章鹿华，王思彤. 面向智能用电的家庭综合能源管理系统的设计与实现[J]. 电测仪表，2010，47(537)：35-38

[2] 刘建明. 智能用电：引领低碳、和谐、智能生活[N]. 国家电网报，2010-10-15

[3] 林海啸. 未来智能小区构想[J]. 法制与社会，2009(5)：288

[4] 陈英，张艳丽. 谈谈智能化住宅和智能小区[J]. 科技信息，2007(18)：270

[5] 刘孝利. 国外智能小区发展历程[J]. 国外视野，2008(3)：26-27

[6] 邵瑾，梁明. 智能用电综合模拟平台的设计与实现[D]. 北京：华北电力大学，2010：7-9

[7] 杨晓文，韩燮. 网络化自动测试系统的数据模型研究[J]. 计算机工程，2010，36(11)：17-49

[8] Molderink A，Bakker V，Bosman M，et al. Management and control of domestic smart grid technology[J]. IEEE Transactions on Smart Grid，2010，1(2)：109-119

[9] Kok K，Karnouskos S，Nestle D. Smart houses for a smart grid [C]//20th International Conference on Electricity Distribution. 2009，751：1-4

[10] Drury E，Denholm P，Margolis R. Modeling the U. S. Rooftop Photovoltaics Market[J]. American Solar Energy Society，2010：1-7


(上接第 379 页)

**结束语**　针对本文 CC 协议定时器实现方案，我们在 PB 仿真器和 OMAP 5910 嵌入式硬件平台上，基于开发的测试程序，作了定时器协议功能的相关测试。结果显示，其能够准确实现网络侧和移动台侧发起的呼叫与释放等协议主要过程的定时关系，具有良好的一致性和稳定性。在实时性方面，由于采用了链表结构和相对时间算法，大大减少了运算时间开销；又由于采用了单线程设计方式，也减少了系统线程调用开销。此外，本方案还采用了 WinCE 的同步互斥量机制，实现了链表元素的插入与删除，提高了系统的实时性。在硬件资源利用方面，为了适应嵌入式系统内存容量小的特点，采用了 WinCE 堆管理机制，实现了有效的动态内存回收，提高了内存使用效率，也（从硬件层面）为实现嵌入式系统的实时性能提供了可能。通过在 OMAP 5910 硬件平台上的测试，本文设计方案可满足 3G 移动终端的相关性能要求。

应该指出，虽然本文所述方案是针对 UMTS 核心网的实现而提出的，其对于类似通信网络的协议实现仍具有一定的参考和借鉴作用。

## 参 考 文 献


[1] 陈晓炜，石江宏. 一种高效率的定时器管理模块设计[J]. 单片机与嵌入式系统应用，2010，1：30-31，40

[2] 邹仕祥. 通信系统中大量定时器的设计与分析[J]. 计算机应用，2005，25(11)：2715-2716，2719

[3] 张玉艳，方莉. 第三代移动通信[M]. 北京：人民邮电出版社，2009：103-104

[4] 3GPP. Mobile Radio Interface Layer 3 Specification：Core Network Protocols [S]. ftp://ftp. 3gpp. org

[5] Taxes Instruments. OMAP 5910 Dual-Core Processor Technical Reference Manual(SPRU602)[S]. US：TI，2002

[6] Lippman S B. C++ Primer(第三版)[M]. 潘爱民，等译. 北京：中国电力出版社，2005：315-323

[7] Savitch W. C++面向对象程序设计—基础、数据结构与编程思想(第四版)[M]. 周靖，译. 北京：清华大学出版社，2003

[8] Boling D. Programming Microsoft Windows CE. NET[S]. US：Microsoft press，2003