Software Engineering and Applications 软件工程与应用, 2022, 11(2), 301-307
Published Online April 2022 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2022.112032

Hans汉斯

时间触发嵌入式系统在某测试装置中的应用

夏中亚，王志刚，蒋　平

中国船舶集团有限公司第七一〇研究所，湖北 宜昌

收稿日期：2022年2月18日；录用日期：2022年4月7日；发布日期：2022年4月14日

摘　要

介绍了一种AVR单片机与时间触发方式相结合的小型嵌入式系统，在分析应用需求的基础上，描述了该系统的结构。该系统的响应精度可有效地控制在ms级，具有占用存储单元少、可靠性高、安全性高的特点。实际应用表明，该系统运行效率高、稳定可靠，可以满足绝大多数工程实际的需要。

关键词

嵌入式系统，时间触发，AVR，多任务，合作式调度器

Application for Time-Triggered Embedded System in a Test Device

Zhongya Xia, Zhigang Wang, Ping Jiang

The 710 Research Institute of CSSC, Yichang Hubei

Received: Feb. 18th, 2022; accepted: Apr. 7th, 2022; published: Apr. 14th, 2022

Abstract

This paper introduces a small embedded system which combines the AVR family of microcontrollers and time-triggered mode, and describes the system's architecture based on analysis of requirement in application. This system takes up less memory units with high reliability and security, and its response accuracy can be effectively controlled at millisecond level. Practical application indicates that: the system can operate efficiently, stably and reliably, which can meet most actual engineering requirements.

文章引用：夏中亚，王志刚，蒋平. 时间触发嵌入式系统在某测试装置中的应用[J]. 软件工程与应用，2022，11(2): 301-307. DOI: 10.12677/sea.2022.112032

Keywords

Time-Trigger, Embedded System, AVR, Multitask, Cooperative Scheduler

1. 引言

目前，嵌入式系统软件有 VxWork、Linux、μC/OS-II 等[1]，其本身所具有大量的代码，占用单片机大量的存储空间，不适用于 8 位的 AVR 微处理器。出于成本和技术上的考虑，AVR 微处理器的软件开发还是基于处理器直接编写，没有配备多任务操作系统作为开发平台，也不需要将系统软件和应用软件完全分开处理。而在实际应用中，AVR 微处理器通常会面临同时应付多任务、多外设的情况，则对它们的相互调度必不可少。

因此有必要针对 8 位微处理器设计一种简单的操作系统，在实际应用中生成少量代码，以节省单片机宝贵的存储空间，进而达到工程应用的目的[2]。时间触发嵌入式系统就是这样简单的操作系统，它可以看作是一种任务调度器[3]。设计人员能够通过仔细安排事件的可控顺序，保证系统一次只处理一个事件，有助于减少 CPU 的负荷，减少单片机存储器的占用量，提高程序流程的可预测性的同时，也简化程序结构。本文设计了基于 AVR 微控制器的时间触发多任务调度器，使用传递消息(message)的方式使得 AVR 微控制器在多个任务及外设间切换，并应用于实际。

2. 传统编程方式的局限性

嵌入式系统中，通常采用两种不同的调度方式：事件触发和时间触发[4]。传统的事件触发方式，即目前工程中常用的超级循环结构，通常采用多级中断的方式实现，其发生时间具有随机性；而时间触发方式则不同，它是通过一个全局时钟进行程序驱动的，系统的行为不仅在功能上确定，而且在时间上也是确定的[5]。

在工程实际应用中，当系统略微复杂时，采用传统的“超级循环”编程结构，也就是事件触发的方式，系统的中断数量较多，且功能稍微复杂时，就会使程序编写变得很复杂，并程序运行的可预测性迅速下降，甚至导致中断触发丢失，给产品开发带来灾难性的后果。下面以一个产品研发过程中面临的问题为例来进行说明。在设计一个用于采集某武器水下航行数据的测试装置时，其系统模块框图如图 1 所示，CPU 的程序设计需完成的任务如下所示：

① 每 10 ms 通过 A/D 采集芯片进行一次深度数据采集，执行时间 < 2 ms；
② 每 10 ms 对 I/O 口线的电平进行一次记录，执行时间 < 1 ms；
③ 每 10 ms 通过串口接收外设发送的一帧报文，速率为 115,200 bps，执行时间 < 2 ms；
④ 每 100 ms CPU 通过 SPI 总线与片外 Flash 通信，写入缓存的采集数据，执行时间 8 ms；
⑤ 依据工作时序，随机接收 485 串口的报文并返回相应指令，速率为 38,400 bps，执行时间 < 2 ms；
⑥ 依据工作时序，通过 232 串口向上位机发送测试装置工作状态信息，执行时间 < 2 ms；
⑦ CPU 通过 I2 C 总线与时钟芯片通信，执行时间 < 0.1 ms；
⑧ 依据工作时序，输出 100 ms 点火脉宽信号，执行时间 < 0.1 ms；

⑨ 每 100 ms 看门狗喂狗任务，执行时间 < 0.1 ms。

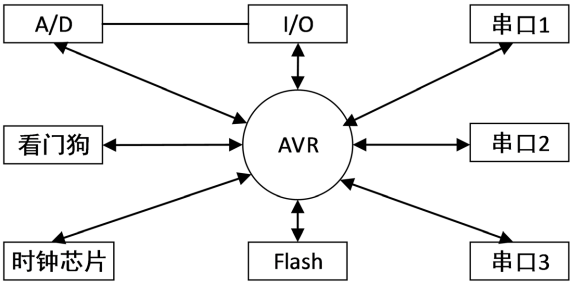

Figure 1. Diagram of system module
图 1. 系统模块框图

以上任务中，任务③、⑤和⑥属于强实时性的，系统若不及时响应串口的收发任务，接收数据时会引起字节丢失，发送数据时会导致数据字节之间的时间间隔变大，导致接收方的数据帧定界错误。而其他任务在设定的周期内能得到执行就可以。以下介绍采用“超级循环”结构实现上述程序设计时遇到的问题。

为确保任务③、⑤、⑥响应的及时性，采用了 UART 中断触发方式，当 UART 完成一个字节的收或发后触发中断，在中断响应函数中将收到的数据保存在接收缓冲区，或者从串口发送缓冲区读取下一个待发送的字节放入 UART 发送数据。为确保任务④响应的可靠性，采用双缓存交替写入 Flash 的方式。因为当系统掉电时，系统只有不到 10 ms 的过渡时间，系统如果不能在这个时间内完成相关的操作，将会丢失测试数据，导致测试的无效。

若采用事件触发的结构进行编程，代码结构如下：

```
While(1)
    {
任务①;
任务②;
……
任务⑨;
}
```

由于任务④的时间最长，在某些情况下，一个循环中会有①②③④⑤⑥⑦⑧⑨的任务，则整个循环的运行时间就会长于 10 ms，这导致各项任务响应时间的要求无法满足，甚至会导致程序中某些任务事件的丢失，比如 A/D 采样任务，I/O 口线电平记录任务。

3. 采用时间触发的模式编程

时间触发方式通过一个全局时钟进行驱动，系统的行为不仅在功能上确定，而且在时间上也是确定的，其编程的关键是设计一个基于时间触发方式的合作式任务调度器，在系统程序中尽量少用事件触发方式(减少系统中断的使用)，使用任务调度器对各项任务的调度执行[6]。时间触发编程模式的典型程序流程结构如图 2 所示。

初始化函数的主要用途是读取内存参数，并设置系统初始的软硬件参数；建立各任务参数函数则根据系统中的各项事件任务需求，建立并分配任务参数，例如用来产生驱动调度器的定时“时标”。任务刷新函数是调度器的中断服务程序，它由定时器的溢出激活。当任务刷新函数确定某个任务需要执行时，

将这个任务的运行标识符置位，然后该任务将由任务调度函数执行。任务调度函数与任务刷新函数配合激活需要运行的任务。

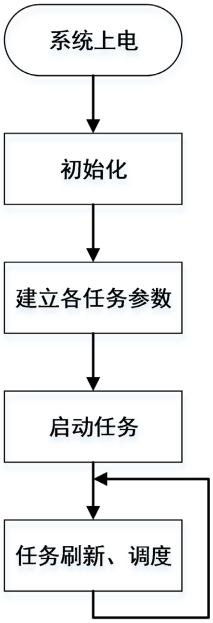

Figure 2. Program structure diagram of time-triggered embedded system
图 2. 时间触发嵌入式系统程序结构图

3.1. 消息队列

消息队列是调度器的核心，它是用户自定义的数据类型，包括了每个任务所需要的信息[7]。尽量将其存储在 DATA 区，以供快速存取。消息队列的结构如下。

```
typedef data struct
{void(code*pTask)(void);//指向任务的指针
uint16 Delay; //延时间隔
uint16 Period; //连续运行的间隔
char    Runme;//任务运行标志
}s_task;
s_task SCH_task_array[SCH_MAX_TASKS];
```

3.2. 调度器初始化函数

该函数执行各种重要操作，如准备调度器队列，但它最主要的用途是设置定时器，用来产生驱动调度器的时标。本文所选用 AVR 系列的 ATmega1280 微控制器具有六个定时器(两个 8 位，四个 16 位)，任一个都能用来驱动调度器，权衡考虑选用定时器 1。

```
void SCH_Init_T1(void)
{逐个删除各个任务；停止定时器 1；设置定时器 1；使能定时器 1 方式；启动定时器 1;
}
```

在设置定时器 1 时需要对调度器任务刷新的时间间隔进行设置，即定时器 1 的定时间隔。根据经验，一个较为可取的时间间隔是略大于一次典型的任务执行所需要的时间，使大多数进程在一个时间片内完成。经反复尝试，时间间隔选择在 1~5 ms 之间执行效率比较高，这样既可满足响应速度的要求又能把任务执行的时间降到最小，在本文中选择 2 ms。

3.3. 调度器任务添加函数

该函数用来将任务添加到消息队列，以保证条件满足时被调用，函数如下所示：

uint8SCH_Add_Task(void(code*pFunction)(),const uint16 delay,const uint16period)

{定义静态变量 i；循环判断任务队列是否有空间；若无，报错返回；否则，添加任务；}

3.4. 调度器任务刷新函数

调度器任务刷新函数即为定时器 1 触发的中断服务程序，通常采用 CTC 方式触发。在该函数中，当某个任务需要执行时，其运行标志 Runme 置 1，然后该任务被调度程序执行，初始化函数中定时器的设置决定了它的调用频率。该函数结构如下。

#pragma vector = TIMER1_CAPT_vect　//AVR 中断服务程序

__interrupt void SCH_Update(void)

{定义变量；检测是否有任务；任务需要运行，Runme+1；

任务还没准备好运行，Delay-1；

}

3.5. 任务类型的划分

对章节 1 里的任务事件进行分析，根据特点可以将这些嵌入式系统的任务事件划分为以下 3 类。

1) 及时型任务。该类任务属于事件触发型的，此类事件一旦发生，在规定的时间内系统必须对其响应，针对该种任务，通常采用中断方式来完成。③、⑤、⑥属于及时型的任务。

2) 周期型任务。该类任务是周期型的时间触发式的，在规定的周期内系统必须确保执行此类任务，通过时间触发编程模式可以较好地实现这类任务的需求。①、②、⑨属于周期型的任务。

3) 背景型任务。这类任务不是实时型的，对实时性要求并不是很高，系统在程序执行过程中可随时中断该类任务进而执行前两类任务。这类任务需要系统在充分利用系统资源的基础上尽最大速度完成即可。④、⑦、⑧属于背景型的任务。

根据上述任务分类可知，及时型任务优先级是最高的，周期型任务优先级次之，背景型任务优先级是最低的，优先级高的任务可随时中断优先级低的任务的执行，同等级别的任务之间不可以相互中断执行。

3.6. 程序结构

为确保优先级高的任务退出后，优先级低的任务执行环境的复原，可以参考中断的执行机制用如下方法进行设计：

在系统程序中设计定时中断函数 I，对周期性任务进行调度，在系统全部中断中这个定时中断的优先级是最低的。

在系统程序中再设计另一个定时中断函数 II，对周期型任务的任务管理队列进行刷新，支持任务调度，在系统中全部中断中这个定时中断函数的优先级次低。

周期型任务通常是一个函数，在这个函数入口的首要操作是打开中断，允许该类任务运行期间被中断打断，响应及时型任务。

背景型任务指的是在系统主函数的超级循环流程中执行的代码，该类程序可随时被及时型与周期型任务中断，当系统程序中不需要执行及时型任务与周期型任务时，才允许循环执行背景型任务的程序。

通过上述分析，采用“时间触发编程模式”的系统程序结构如下所示：

```
//主函数
void main(void){
SCH_Init()；//设置系统调度器
SCH_Add_Task(任务函数名称，任务调度延迟设计，任务调度周期等)；//将系统任务加入调度器的任务队列中
SCH_Start()；//刷新系统任务队列
while(1) {
背景型任务 1；
……
背景型任务 n；
}
}
//优先级次低的定时中断函数
void SCH_Update(void)interrupt{
//刷新系统任务队列
}
//优先级最低的定时中断函数
void SCH_Dispatch_Tasks(void)interrupt{
//调度系统周期型任务
}
//周期型任务的典型结构
void SCH_Cycle_Task1(void){
打开中断；
执行任务；
return；//任务返回
}
```

4. 应用分析

如上文所述，某型武器水下航行数据测试装置是一个复杂的嵌入式系统，其微控制器需要处理大量的外围设备，如图 1 所示。为了便于开发，将系统程序按照硬件的功能分别划分模块，各个软件模块之间通过传递消息的方式来完成多任务的处理。使用上述介绍的调度器既方便了程序的设计和维护，又解决了多个任务之间的调度问题。针对该型测试装置的应用，模块入口函数数组 TskTbl[]如表 1 所列，使用函数数组的方式可以增强程序的扩展能力。如果增加新的外设，只需在这里添加相应的模块入口，并完成对应的模块代码就可以增加系统的功能。

Table 1. System function module division
表 1. 系统功能模块划分

序号	TskTbl[]	描述
1	SERIAL_wvdPrcMcg	处理串口任务
2	AD_wvdPrcMcg	处理 AD 数据采集任务
3	WD_wvdPrcMcg	处理看门狗任务
4	CLOCK_wvdPrcMcg	处理时钟任务
5	FLASH_wvdPrcMcg	处理存储器任务
6	IO_wvdPrcMcg	处理 IO 操作任务

该系统在 Atmel 公司 AVR 系列单片机的开发平台上用 C 语言实现，调度器在某型测试装置系统中很好地发挥了作用，经应用证明，系统的测试程序流畅，保存的数据完好。

5. 结束语

通过上述分析可知，在内存资源较少的单片机小型嵌入式系统采用“时间触发编程模式”进行编程，与采用 RTOS 进行编程相似，设计人员设计好系统中的任务后，就可以专心设计每个任务，每个任务占用处理器的时间间隔可以由系统程序进行管理，减少各个任务相互之间的耦合，使得产品的程序设计、改动与优化变得简洁明晰。采用“时间触发编程模式”较好地解决了某测试装置产品设计过程中面临的复杂设计困扰，实际应用证明，这种方法简洁有效，稳定可靠，适用于对成本和稳定性均有要求的小型嵌入式系统。

参考文献

[1] 何立民. 嵌入式系统的定义与发展历史[J]. 单片机与嵌入式系统应用, 2004(1): 6-8.

[2] Pont, M.J. (2002) Patterns for Time-Triggered Embedded Systems: Building Reliable Applications with the 8051 Family of Microcontrollers. Addison Wesley, New Jersey.

[3] 李奇, 樊晓平. 一种时间触发的多任务调度器设计[J]. 单片机与嵌入式系统应用, 2008(5): 17-19.

[4] 朱凤新, 姚竹亭. 基于 AVR 的时间触发嵌入式系统[J]. 工业控制计算机, 2006(7): 56-57.

[5] 周全, 窦振中, 孙传群, 等. 为嵌入式软件建立统一软件系统框架的方法[J]. 电子产品世界, 2002(17): 17-18.

[6] 廖晓文, 廖京盛. 时间触发模式的任务调度与分解策略[J]. 单片机与嵌入式系统应用, 2006(7): 8-9.

[7] Kim, K.H. (2004) The Distributed Time-Triggered Simulation Scheme. Core Principles and Supporting Execution Engine. *Real-Time Systems*, **26**, 9-28. https://doi.org/10.1023/B:TIME.0000009304.92936.70