第35卷 第19期 计 算 机 工 程 2009年10月
Vol.35 No.19 Computer Engineering October 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)19—0257—03 文献标识码：A 中图分类号：TP301.6

# 面向嵌入式 PLC 的调度算法

严 义，胡峰令
(杭州电子科技大学智能与软件技术研究所，杭州 310018)

**摘 要：**针对信号扫描周期要求不同导致嵌入式可编程逻辑控制器(PLC)控制效率下降的问题，提出一种基于群 I/O 任务的嵌入式 PLC 调度算法。该算法实现了快慢任务子集的自动划分，并通过对 2 个子集采用不同调度策略以改进机器的控制效率。实验结果表明，该算法系统开销小，可移植性强，适用于资源有限的嵌入式 PLC 系统。

**关键词：**嵌入式可编程逻辑控制器；调度算法；扫描周期

# Embedded PLC Oriented Schedule Algorithm

YAN Yi, HU Feng-ling
(Institute of Intelligent and Software Technology, Hangzhou Dianzi University, Hangzhou 310018)

**【Abstract】** Aiming at the problem that different requirements for signal scan cycle lead to the low control efficiency in embedded Programmable Logic Controller(PLC), this paper proposes a schedule algorithm based on Group I/O Task. It realizes the automatic division of fast and slow task sets, and adopts different schedule strategies for two sets to improve the control for machine. Experimental results show that the algorithm has low system cost, good portability, and it is applicable to limited resource of embedded PLC.

**【Key words】** embedded Programmable Logic Controller(PLC); schedule algorithm; scan cycle

## 1 概述

嵌入式可编程逻辑控制器(Programmable Logic Controller, PLC)应用现场存在各类实时性不同的信号，实时性不同造成信号扫描周期的差异，高实时性信号对扫描周期有严格要求，低实时性信号则允许不精确的扫描周期。很多学者对解决传统 PLC 单周期扫描执行方式没有考虑信号扫描周期差异性的问题进行研究。文献[1-2]提出以抢占式实时内核作为嵌入式 PLC 内核的设计方案，保证了高优先级信号的实时性，但系统开销较大。文献[3-4]提出分布式架构的 PLC 设计方案，增加处理模块以适应不同扫描周期的信号，但模块间调度关系复杂，硬件成本高。文献[5]提出简单快慢逻辑设计方案，以快慢 2 个级别的任务提高控制效率，系统开销小，但该方案无法实现快慢任务的在线划分，且慢任务的单周期扫描方式影响了控制响应时间。

针对已有方法的局限性，以单处理器的架构方案为基础，提出一种新的嵌入式 PLC 调度算法，记为动态中断和时间触发(Dynamic Interrupt and Time Triggered, DITT)调度算法。DITT 调度算法实现了快慢任务子集的自动划分，并通过对 2 个子集分别采用定时中断抢占调度策略和时间触发[6]调度策略的方式。为了适应当今 PLC 的发展方向，DITT 算法参考国际标准化组织(国际电工技术委员会)制定的 IEC 61131-3 标准[7]。该标准描述了 PLC 的多任务模型，但没有阐述具体实现方案。

## 2 动态调度模型

### 2.1 任务模型

DITT 调度算法的任务模型符合 IEC61131-3 标准中的单资源任务模型，1 个资源内部存在多个任务，每个任务可以运行独立的控制程序。DITT 调度算法中的任务模型具有如下定义：

**定义 1** 群 I/O(Group I/O)：相互间具有逻辑依赖关系的一组 I/O 端口，记作 GPO。2 个 GPO 如图 1 所示，其中，($X0$,$Y0$,$Y1$)是一个 GPO；($X2$,$Y2$)是另一个 GPO。

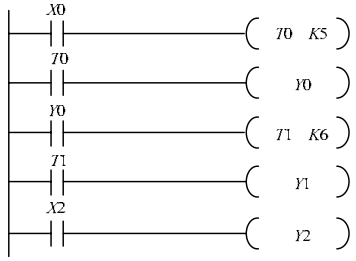


**图 1 2 个 GPO**

**定义 2** GPO 优先级：一个代表 GPO 实时性程度的量，记作 GPOP，GPOP 越小，实时性越高。

**定义 3** GPO 周期：每个 GPO 的扫描周期(单位为 ms)，记作 GPOC，当 GPOC 为 0 时表示该 GPO 为非周期性 I/O。每个非周期性 GPO 中信号的最短持续时间被称为有效时间，记作 GPOE。

**定义 4** GPO 控制体：基于某个 GPO 的 PLC 控制程序，以及该 GPO 的输入输出操作，记为 GPOT。

任务 $T_i$ 可以由多元组<$GPO_i$, $GPOP_i$, $GPOC_i$, $GPOE_i$, $GPOT_i$>, $i$ = 0, 1, …, $N$-1 表示。

该任务模型遵循以下 2 个规则：

**规则 1** PLC 工作负载由大量短任务(执行时间为 10 μs~100 μs)，小量的长任务(执行时间为 1 ms~200 ms)组成。

**作者简介：**严 义(1961－)，男，教授，主研方向：仪器仪表，智能控制，嵌入式技术；胡峰令，硕士研究生

**收稿日期：**2009-04-11 **E-mail：**yybjyyj@163.com

**规则 2** 高优先级的任务均为短任务，且大多数长任务为周期性任务。

**定义 5** 中断占用率：快任务集的执行时间与总时间之比，记为 ITO，ITO 是 DITT 算法任务划分的依据。

**定义 6** 任务响应成功率：任务实际满足扫描周期要求或有效时间要求的执行次数与预期执行次数之比，记为 *CS*。响应成功率越高，表示系统性能越好，控制效率越高。

平均任务响应成功率定义为 $\overline{CS}=\frac{\sum_{i=0}^{N-1}CS_i}{N}$，其中，$N$ 为任务数。

DITT 算法中的系统目标可以描述为在系统 ITO 的约束下，调度任务集$\{T_0, T_1, \cdots, T_{N-1}\}$，保证高优先级任务的 *CS*，并尽量提高系统 $\overline{CS}$。

### 2.2 调度模型

DITT 算法的调度过程表述为在初始阶段，所有任务属于慢任务集且系统处于不稳定状态；调度器实时监测系统 ITO 根据当前 ITO 和设定值的比较结果调整快慢任务集的内容，直到遍历所有任务或 ITO 等于设定值，系统处于稳定状态。快任务集采用定时器中断抢占策略，慢任务集采用时间触发调度策略。系统稳定后，调度器继续实时监测系统 ITO，如果检测到系统再次不稳定，那么调度过程重新开始。DITT 调度模型如图 2 所示。

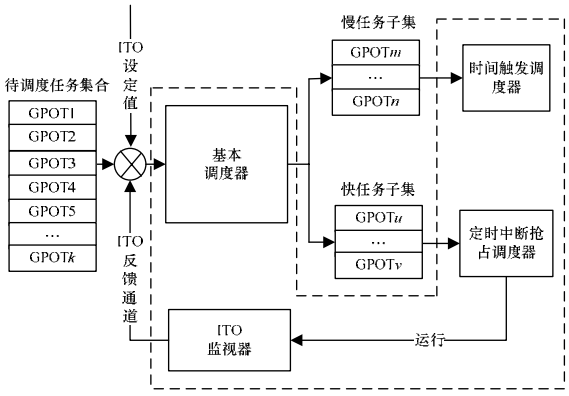


**图 2 DITT 调度模型**

由此可知，DITT 调度由调度器、待调度任务集合、快任务子集和慢任务子集组成。调度器又包括基本调度器、ITO 监视器、时间触发调度器和定时中断抢占调度器。其中，基本调度器根据 ITO 监视器记录当前系统 ITO 将待调度任务集合划分为快任务子集和慢任务子集；时间触发调度器和定时中断抢占调度器则按各自的调度策略独立执行自身任务集的任务。

## 3 DITT 调度算法实现

### 3.1 数据结构描述

为了实现上述调度模型，引入以下数据结构：GPO 任务表，下一个可以调度的任务索引 nexttask 和系统稳定标志 issteady。GPO 任务表中存放基于 GPO 的控制任务，调度器根据该表进行任务调度。GPO 任务表以一维结构体数组 TaskQueue 实现，按优先级从小到大依次排列。

任务节点的结构定义如下：

```
typedef struct TASKTYPE
{
unsigned int cycletime;        //扫描周期 CPOC
        unsigned int currentime;       //再次执行的剩余时间
        unsigned char intrunflag;      //中断运行标记
unsigned char cyclerunflag; //周期运行标记
        void(*gptaskfun)(void);        //GPOT 的入口地址
    } TASKTYPE;
```

其中，中断运行标记 intrunflag 用于表明任务类型，1 表示任务属于快任务集，0 表示任务属于慢任务集；周期运行标记 cyclerunflag 只在任务属于慢任务集时有效，1 表示该任务扫描周期已到，0 表示该任务扫描周期未到。

### 3.2 算法步骤

DITT 调度算法占用 2 个硬件定时器：(1)用于驱动快任务集和产生系统时钟节拍(如 *T*0)；(2)用于计算 ITO(如 *T*1)。通过硬件定时器计算 ITO 的方式，实现 ITO 监视器基本不占用额外的处理器时间的效果。下文给出具体的 DITT 算法步骤，包括基本调度器、ITO 监视器、时间触发调度器和定时中断抢占调度器 4 个部分的内容。

(1)初始化任务表(TaskQueue), nexttask 和 issteady。

(2)如果产生定时器中断，那么在其中断服务例程中执行以下步骤：

1)执行中断运行标记 intrunflag 为 1 的任务并更新慢任务集中周期任务的时标 currenttime；

2)实时计算系统 ITO，并更新当前 ITO 值；

3)若系统从稳定状态变为不稳定状态时，则重新开始调度；

4)若系统 ITO 大于设定值且系统未稳定，则选择一个快任务到慢任务集；

5)若此时选择的任务为最后一个任务或 ITO 等于设定值，则置位系统稳定标记 issteady。

其中，步骤 4)、步骤 5)置于中断处理例程中，虽然会略微增加中断处理时间，但可使系统快速稳定。

(3)循环任务集处理程序中，执行以下步骤：

1)关定时器中断；

2)若更新后的系统 ITO 小于设定值且系统未稳定，则选择下一个任务进入快任务集；

3)若此时选择的任务为最后一个任务或 ITO 等于设定值，则置位系统稳定标记 issteady；

4)开定时器中断；

5)选择慢任务集中时标为 0 的任务，重新初始化任务时标并置位标记 cyclerunflag；

6)执行慢任务集中 cyclerunflag 为 1 的任务，任务完成后复位标记 cyclerunflag。

(4)重复步骤(2)~步骤(3)，直至结束。

## 4 实验及结果分析

实验硬件环境：嵌入式 PLC 控制器，MCU Atmega128，时钟频率 7.372 8 MHz。

实验方法：设置任务数 20，优先级依次降低。第 1 个~第 18 个为非周期性短任务，运行时间分布在 30 μs~90 μs 之间，有效时间为 10 ms；第 19 个任务为周期性长任务，运行时间约 20 ms，扫描周期为 100 ms，第 20 个任务为周期性长任务，运行时间为 150 ms，扫描周期为 1 000 ms；ITO 依次取 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%；时钟中断设定值为 1 ms。

本次实验共需 125 Byte 的数据空间(RAM)，ITO 为 50% 的任务响应率如表 1 所示。

**表 1　ITO 为 50%的任务响应率**　(%)

| 任务序号 | 1~6 | 7~18 | 19 | 20 | 平均 |
|---|---|---|---|---|---|
| DITT 调度算法 | 100.0 | 42.0~48.0 | 64.0 | 100.0 | 65.7 |
| 传统单周期扫描算法 | 0.0 | 0.0 | 0.0 | 100.0 | 5.0 |
| 简单快慢调度算法 | 100.0 | 0.0 | 0.0 | 100.0 | 35.0 |

由此可知，对于存在变扫描周期信号的应用场合，DITT 算法可以自动保证高优先级任务的实时性，如任务 1~任务 6，还可以提高系统平均响应成功率。对于只存在单一长任务的应用场合，DITT 算法和其他算法有相同效果，这可以从第 20 个任务的统计结果看出。不同中断占用率下的快任务个数如图 3 所示，不同中断占用率下的平均任务响应率如图 4 所示。

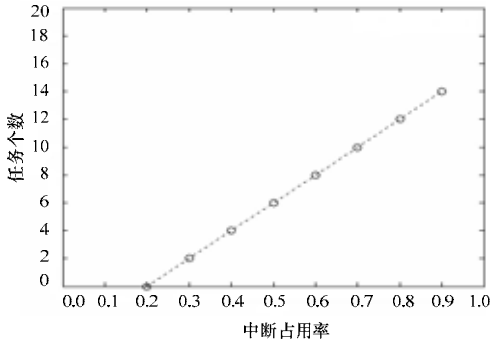


**图 3　不同中断占用率下的快任务个数**

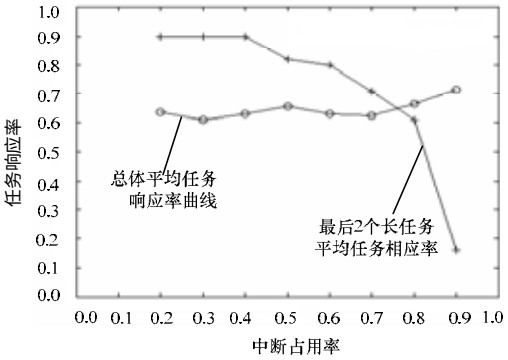


**图 4　不同中断占用率下的平均任务响应率**

由图 3、图 4 可知，提高系统中断占用率可以增加快任务个数，还可以在一定程度上提高总体平均响应成功率，但同时会引起长任务响应成功率的下降，因此，实际应用时应选取一个合适的中断占用率。

## 5　结束语

本文以基于 GPO 的任务模型为基础，提出动态中断抢占策略和时间触发策略相结合的 DITT 调度算法。实验结果表明，DITT 调度算法能较好地解决该问题。现有 PLC 程序与 DITT 算法的任务模型的不一致限制了 DITT 算法在工程实际中的应用。由于 ITO 值选择的多样性，ITO 值和时钟中断设定值间的紧密关系，增加了 DITT 算法应用难度，因此下一步要研究分离现有 PLC 程序的 GPO，以及自动选择合适的 ITO 值和时钟中断设定值。

## 参考文献

[1] Kong Yaguang, Wang Wenhai. Design of Real Time Control Software Based on QNX[C]//Proc. of the 32nd Annual Conference on IEEE Industrial Electronics. Paris, France: [s. n.], 2006: 579-584.

[2] 葛益军, 王文海. IEC61131-3 之多任务运行系统的设计和实现[J]. 国内外机电一体化技术, 2007, 10(8): 27-29.

[3] Roengruen P, Suesut T, Tipsuwanporn V, et al. Design of PLC Networks Using Remote I/O Module Based on Controller Area Network[C]//Proc. of Canadian Conference on Electrical and Computer Engineering. Ottawa, Canada: [s. n.], 2001: 1023-1027.

[4] Jeong S, Kim Y S, Kwon W H. Scheduling Algorithm for Programmable Logic Controllers with Remote I/Os[C]//Proceedings of the 4th International Workshop on Real-time Computing Systems and Applications. Taiwan, China: [s. n.], 1997: 87-94.

[5] 幺开宇. 基于 IEC61131-3 标准的嵌入式软 PLC 虚拟机[J]. 制造技术与机床, 2005, (11): 67-70.

[6] 郭丽娟, 刘双与, 张　激. 基于时间触发的高可靠性实时系统架构[J]. 计算机工程, 2006, 32(4): 272-274.

[7] John K H, Tiegelkamp M. IEC61131-3: 工业自动化系统的程序编制[M]. 中国机电一体化技术应用协会秘书处, 译. 北京: 中国机电一体化技术应用协会出版社, 2002.

编辑　陆燕菲

（上接第 256 页）

时间。在进入 S0/D2/C2 状态后不会立即进入 S2/D3/C4 状态，而会反复几次退出到 idle 状态后再进入 S0/D2/C2。若期间无应用需求，则会进入 S2/D3/C4 状态。这样同时兼顾对应用的响应时间和能量消耗，但不一定能够达到最好效果。在实际应用系统中，若电池能量要求较高，则应尽快进入 S2/D3/C4 状态；若对响应的要求较高，则可以适当延长 CPU 处于 idle 或者 S0/D2/C2 状态的时间。

## 5　结束语

本文基于 Monahans 的智能导航平台，通过驱动层与各设备协调工作完成对频率、电压和 CPU 温度的实时调整。在应用层通过模拟各种应用程序的实际运行与电源管理器进行交互。当有应用程序运行时，系统在满足应用程序正常运行的前提下运行最低功耗。当无应用程序运行时，电源管理器为各省电模式设置相应的参数并同时考虑系统的功耗和响应时间。以上各功能模块相互配合来完成电源管理的功能，达到了预期目的。

## 参考文献

[1] Marvell Semiconductor, Inc. System and Timer Configuration Developers Manual[EB/OL]. (2008-02-07). http://www.marvell.com/products/cellular/application/pxa300.jsp.

[2] Ramamurthy P, Palaniappan R. Performance-directed Energy Management Using BOS[J]. ACM SIGOPS Operating System Review, 2007, 41(1): 66-77.

[3] Bovet D P, Cesati M. 深入理解 LINUX 内核[M]. 陈莉君, 冯　锐, 牛欣源, 译. 南京: 东南大学出版社, 2006.

[4] 魏永明, 耿　岳, 钟书毅. LINUX 设备驱动程序[M]. 北京: 中国电力出版社, 2006.

[5] Liu Xiaotao, Shenoy P, Corner M. Chameleon: Application Level Power Management with Performance Isolation[C]//Proceedings of the 13th Annual ACM International Conference on Multimedia. New York, USA: ACM Press, 2005: 839-848.

编辑　陆燕菲