IN1009 GUTTA 在不同 CPU 上的速度测试

2009/02/15

概述

随着目前单片机处理器性能的不断提升，单片机应付一般性的逻辑运算，是绰绰有余的。采用 GUTTA 平台开发的控制器，执行速度会比直接采用 C 语言开发的控制器慢上很多。速度的差异是几方面形成的。其一是 PLC 寻址所消耗的时间。由于单片机无法做指令地址的硬解析，解释型 PLC 要逐位分析 PLC 指令中特定的地址格式，从而得到在内存中的实际地址（同时还要完成地址操作数、变量操作数、指针操作数的判断等）。然后根据这个真实地址，初始化参数内存。其二是 PLC 指令派发所消耗的时间。初始化参数后，解释型 PLC 需要根据 PLC 指令中的代码号，调用指定的处理子程序。其三是中断处理所消耗的时间，解释型 PLC 有自己的中断处理系统，每进行一个最原子操作，都要进行一次中断事件的判断。对于编译型的 PLC，除了指令地址的解析不需要做之外（直接赋值），初始化参数内存、指令派发、中断处理和解释型的基本一致。

相对于硬 PLC 系统（含专用位处理器），单片机开发的 PLC 系统，在逻辑指令上慢上 2 个数量级左右。不过硬 PLC 系统在遇到字运算指令时，会产生异常，从而调用通用处理器核进行字运算，通用处理器核处理完毕后，再将控制权归还位处理器。此过程相对复杂，因此硬 PLC 系统对于字运算指令的处理速度相对于单片机 PLC 系统没有太大的优势。（字节、双字运算的情况同字。）

目前 GUTTA 平台在常用的单片机系统上都有移植。这里我们以试验板 CPU-EC20 (AVR)、CPU-EC20 (Cortex-M3)、CPU-EC20 (ARM)为基础，进行指令速度的测试。AVR 是 8 位精简指令集处理器的代表，ARM7 是 32 位精简指令集处理器的代表。Cortex-M3 是 ARM 公司为微控制特别开发的 32 位处理器，架构上也最为先进。

参数

处理器参数对比

CPU-EC20	AVR	Cortex-M3	ARM
处理器内核	AVR® 8-bit Microcontroller	ARM 32-bit Cortex™-M3 CPU	16/32-bit ARM7TDMI-S CPU
处理器型号	ATMEGA64	STM32F103	LPC2134
核心频率	11.0592MHz	72MHz	35MHz
SRAM	4KB	20KB	16KB
FLASH	64KB	64KB	128KB
ISP/IAP	Y/Y	Y/Y	Y/Y
Timer/Counter	8bit X 2 + 16bit X 2	16bit X 4	32bit X 2
USART	2	3	2
USB	N	1	N
供电电压	5.0V	3.3V	3.3V

PLC 实现参数对比

CPU-EC20	AVR	Cortex-M3	ARM
PLC 名称	CPU-EC20-AVR	CPU-EC20-CM3	CPU-EC20-ARM
PLC 信息	CPU-EC20 (AVR)	CPU-EC20 (Cortex-M3)	CPU-EC20 (ARM)
系统页大小（字节）	50	50	50
数据页数量	16	16	32
数据页数据项数量	16	16	32
中断程序个数	8	8	16
子程序个数	8	8	16
中断程序参数个数	32	32	32
子程序参数个数	32	32	32
常数区大小（字节）	128	256	256
指令区大小（字节）	18944 (18.5K)	21504 (21K)	42240 (41.25K)
通讯包有效数据长度	64	64	64
最大程序嵌套层数	8	8	8
是否支持单步调试	是	是	是

PLC 编译器参数对比

CPU-EC20	AVR	Cortex-M3	ARM
固件编译器	gcc version 4.3.0 (WinAVR 20080512)	IAR ARM ANSI C/C++ Compiler V4.42A	IAR ARM ANSI C/C++ Compiler V4.42A
固件编译器参数	-Os	-s9	-s9
PLC 逻辑编译器	gcc version 4.3.0 (WinAVR 20080512)	IAR ARM ANSI C/C++ Compiler V4.42A	IAR ARM ANSI C/C++ Compiler V4.42A
PLC 逻辑编译器参数	-Os	-s9	-s9

特别说明，CPU-EC20 (ARM) 处理器工作在 ARM 模式（32bit 指令长度）。

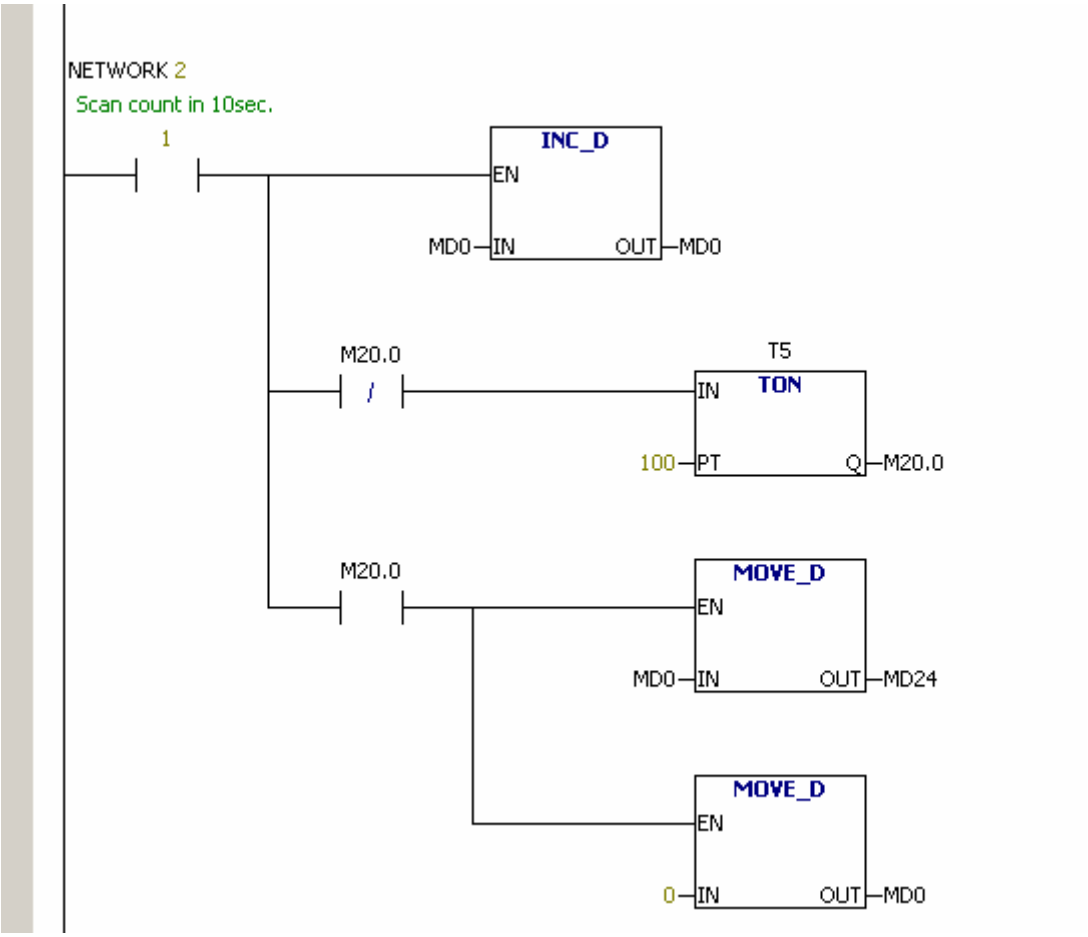
测试方法

由于 GUTTA 系统没有提供主循环扫描执行时间的状态寄存器。这里我们只能够通过主循环在一定时间内的扫描次数来估计主循环扫描执行时间。然后根据主循环程序中的指令数，求得平均每条指令的执行时间。需要注意的是，PLC 的一个扫描周期除了对主循环进行扫描之外，还要更新 I/O，进行通讯处理等。这里假设 PLC 做这些服务的时间为常数，我们通过改变主循环程序中指令的数量，观察主循环扫描执行时间变化，通过增减量来求得平均每条指令的执行时间，同时也能通过计算得到 PLC 做其他服务需要的时间。

测试程序

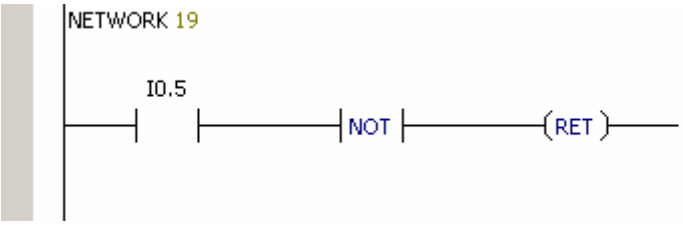
http://visiblecontrol/technologies/inindex/misc_in1009/SpeedTest.vcw

NETWORK 2 统计 10 秒内，程序执行的次数。由于 INC_D 功能块，程序每执行一次，MD0 的值就加 1。每 10 秒（T5 触发）后，MD0 的值被存入 MD24 中，同时 MD 被清零，以便开始统计下个 10 秒的扫描次数。



NETWORK 3 到 NETWORK 18 是重复的代码，没有特殊含义，只是为了测量指令的执行速度。

NETWORK 19 通过判断 I0.5 开关的通断，决定是否继续执行本程序。由于能流取反开关，当 I0.5 没有接通时，执行 RET 线圈，程序结束。当 I0.5 接通时，不执行 RET 线圈，程序继续执行。



NETWORK 20 到 NETWORK 51 是重复的代码，没有特殊含义，只是为了测量指令的执行速度。当 I0.5 接通时，这些指令得到执行。

通过观察 STL 指令表模式下的指令（注意：NETWORK 也是一条指令），我们可以知道，

当 I0.5 没有被按下时，执行的指令数为 153。当 I0.5 被按下时，执行的指令数为 425。

测试结果

CPU-EC20	AVR	Cortex-M3	ARM
解释型（I0.5 = 0）	MD = 1580	MD = 12249	MD = 6489
编译型（I0.5 = 0）	MD = 4201	MD = 24970	MD = 13948
解释型（I0.5 = 1）	MD = 591	MD = 4489	MD = 2425
编译型（I0.5 = 1）	MD = 1659	MD = 9279	MD = 5295

结论分析

根据指令数变化后，扫描次数的变化，我们可以求得平均每条指令的执行时间为：

CPU-EC20	AVR	Cortex-M3	ARM
解释型	38.93μS	5.188μS	9.494μS
编译型	13.40μS	2.489μS	4.307μS

除了主循环扫描，系统服务所用时间为：

CPU-EC20	AVR	Cortex-M3	ARM
解释型	371.4μS	22.55μS	88.23μS
编译型	328.7μS	19.54μS	57.88μS

根据每条指令的平均耗时，得到每 100ms 可以执行的指令数：

CPU-EC20	AVR	Cortex-M3	ARM
解释型	2568	19275	10532
编译型	7462	40176	23218

总结

可以看出，ARM 公司的 Cortex-M3 凭借 72MHz 的核心频率，执行速度上遥遥领先。ATMEL 公司的 8 位的 AVR 单片机，在解释运行时，只能够勉强满足一般的工控要求（100ms 周期内执行 2568 条指令）。值得注意的是 AVR 在编译运行时，速度提升很大，达到将近 300%。ARM7TDMI 中规中矩，不过这里处理器运行于 32bit ARM 指令集模式。对 FLASH 容量的需求几乎是 Cortex-M3 的一倍。

从上面可以看出，这 3 种内核的处理器都能基本上满足控制要求。对于低成本的 8 位 AVR 方案，最好运行于编译模式。对于 32 位处理器，由于编译模式对速度提升不是特别大，建议优先选择解释运行模式。