基于 ARM 处理器的 RealMonitor™ 实现

作　者：
张邦术
上海杰得微电子有限公司
嵌入式软件部经理

摘　要：

本文从运行机制，实现原理，以及通信协议方面详细分析了 RealMonitor™ 在目标机上的实现。RealMonitor 是 ARM 公司提供的一个轻量级实时调试代理，提供了普通调试手段之外的一些实时特性。对于实时系统调试具有独特的优越性，并且可用于作为最终产品的监控代理。

一、前言

RealMonitor 及其重要性

随着SoC功能的复杂化，引入实时调试与监控手段在嵌入式设备开发过程中，对于早期的硬件调试、系统软件开发，以及后期的产品维护都是至关重要的。

RealMonitor 是一个微小程序，它内嵌在运行于目标平台的固件(Firmware)程序或是实时操作系统(RTOS)中，以提供对目标系统硬件平台、嵌入式操作系统以及其它应用程序实时监控、实时调试的强有力的工具。

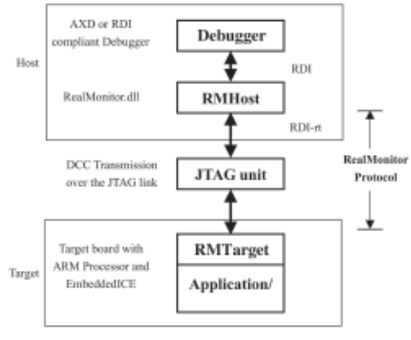
RealMonitor 的优越性

RealMonitor 兼具了硬件调试工具和软件调试工具的优点，它是一个轻量级的调试代理，具有体积小、时延小、不间断等诸多优点。它弥补了一般硬件调试工具下中断不能得到服务的缺陷(一般调试时处理器须置于 debug 状态)，也克服了重量级的软件调试工具所带来的处理时延以及大量占用目标CPU资源所带来的不足。

二、RealMonitor 原理分析

1.　运行机制

RealMonitor 的体系结构如图1所示。



▲ 图 1 RealMonitor 组件及体系结构

RealMonitor 由 RMHost 和 RMTarget 两部分组成，它们通过基于 JTAG 可靠连接之上的 DCC 通道进行通信。

RealMonitor 将一个系统的软件分为两部分。一是连续运行的前台应用程序，典型地工作在用户(usr)、系统(sys)、监控(svc)模式。另一部分是后台应用程序，包含中断、异常等的处理例程。后者由中断或异常所触发，正常情况下，系统服务程序将过滤和处理这些中断或异常。未被处理的异常将触发 RealMonitor 使其进入 DCC 轮询，并停止正在运行的前台应用程序。

RealMonitor 在暂停前台应用程序执行的同时，允许IRQ以及FIQ继续被服务，从而保证了不间断调试(Nonstop Debugging)，这对于实时性要求比较高的系统，以及在实时运行时所诱发故障错误的调试是极其关键的。

2.　连接通信机制

RM– 主机与 RM– 目标机之间通过使 DCC (Debug Communications Channel)进行通信，DCC是依附于JTAG调试接口之上的一条通信途径，通过 CP14 协处理器访问 CPU，在主机和目标平台之间传递数据。在 CP14 里面有两个 RX 和 TX 寄存器，并实现了简单的握手机制。从而可以在目标端使用轮询或中断机制来访问这两个寄存器，与主机侧程序交换信息。

三、RealMonitor 的实现

1.　RealMonitor 的硬件需求

RealMonitor 需要 EmbeddedICE 支持，从而实现JTAG的连接通信。如果EmbeddedICE包含了(–RT)扩展，它还可以提供不间断的断点和监视点的调试支持。如果希望中断驱动DCC，必须将 DCC 的输出信号 COMMRX 和 COMMTX 连接到中断控制器的一根输入线，否则就只能通过软件轮询DCC来处理异步请求。

2. RealMonitor 的软件实现

1) 编译生成RealMonitor 运行库 ARM 随 AFS 软件包提供了基本的 RMTarget 的实现，针对于特定的处理器以及开发板需要对 RMTarget 进行移植，详情参见ARM RMTarget Guide。

使用 ARM 开发工具包编译 RMTarget 可以得到 RealMonitor 运行时库 rm.a。后面的第四节“RealMonitor协议”将详细分析RealMonitor 程序库的具体实现机制。

2) RMTarget 与系统应用程序的集成

要将RMTarget集成到应用程序或是实时操作系统(RTOS)，必须为RMTarget设置堆栈，解决中断复用的问题。

堆栈的初始化

必须在应用程序中为每种处理器工作模式设置足够的栈空间。下表列出 RealMonitor 所需要的最小栈空间。

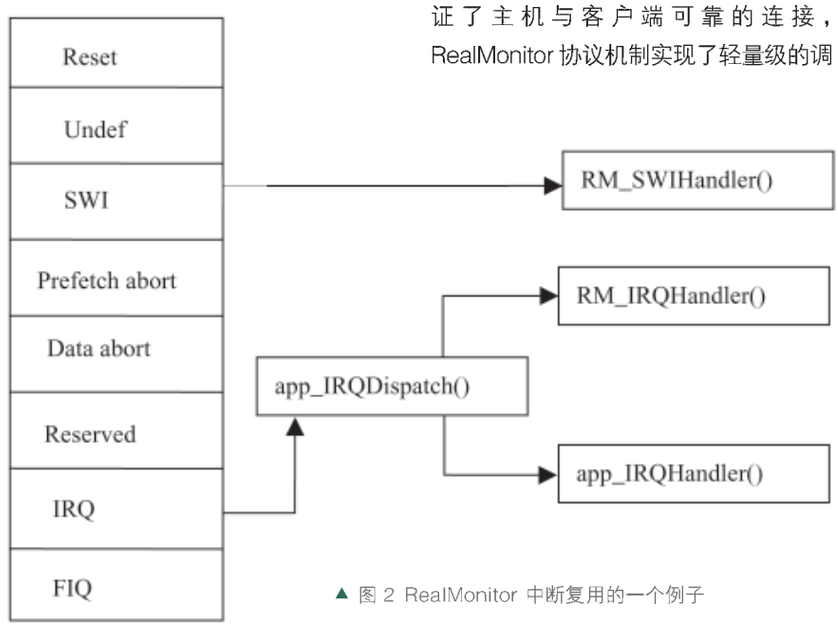
处理器模式	最小栈空间数(WORDS)	备　　注
FIQ	0	RM 不使用这种模式
IRQ	2	必需
Undef	12	必需
SVC	4	如果需要支持 SWI semihosting 时必需
Abort	4	如果需要支持硬件断点或监视断点时必需
User	0	RM 不使用这种模式
System	0	RM 不使用这种模式

中断和异常复用的处理

对于每种Exception, RealMonitor 提供一个异常处理例程。如果应用程序需要共享异常处理，则应用程序需要提供一个分发程序，根据异常的性质分发到相应的处理程序。

最简单的例子是应用程序没有这个异常处理程序，则可以把 RealMonitor 的底层异常处理程序直接安装到异常向量表。如果不希望改变应用程序(中断)的处理架构，则可以采取轮询 DCC 的方式来使用 RealMonitor.

图 2 显示了在应用程序以及 RealMonitor 之间中断复用的一个例子。



▲ 图 2 RealMonitor 中断复用的一个例子

RMTarget 的初始化

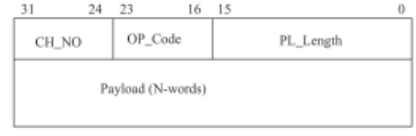
如上所述，RMTarget 的初始化依赖于组件的配置。ARM 定义了 RM_Init()函数的默认实现。在对应用程序进行调试之前，必须对先调用 RM_Init()进行初始化。RM_InitVectors()将安装 RealMonitor 默认的异常处理例程，应用程序必须替代这些向量以提供对中断复用的支持。

四、RealMonitor 协议

RealMonitor 协议是 RMTarget 与 RMHost 通信的核心，以及主机调试器对目标板进行监控与调试的基础。

1. 命令包

采用基于 JTAG 连接的 DCC 传输保证了主机与客户端可靠的连接，RealMonitor 协议机制实现了轻量级的调试控制处理。RealMonitor 基于包(Packet)进行通信与控制。每个命令包(Packet)包含一个 4 字节的包头部以及可选的数据负载。图3显示了数据包的格式:



▲ 图 3 RealMonitor 协议包

其中CH_NO从逻辑上区分不同的调试代理，对于RealMonitor, CH_NO等于0，OP_Code是命令代码，PL_Length是可选的数据负载的长度(N)，随后是数据负载，如果有的话。

断开目标机

断开目标机时应该移去所有的断的状态，此时主机软件应该启动超时处理功能。

五、结语

轻量级的软件代理RealMonitor的实以使用DCC通道向目标机发送其它控制命令，来实现其余调试工作。

▼ 表一列出了 RealMonitor 所使用的 Host-to-Target 命令字代码。

Opcode	Name	Status	Opcode	Name	Status
0x00	NOP	M	0x0A	ReadHalfWords	O
0x01	GetCapabilities	M	0x0B	WriteHalfWords	O
0x02	Stop	O	0x0C	ReadWords	O
0x03	Go	O	0x0D	WriteWords	O
0x04	GetPC	O	0x0E	N/A	Reserved
0x05	SyncCaches	O	0x0F	N/A	Reserved
0x06	ExecuteCode	O	0x10	ReadRegisters	O
0x07	InitializeTarget	O	0x11	WriteRegisters	O
0x08	ReadBytes	O	0x12	ReadCPRegisters	O
0x09	WriteBytes	O	0x13	WriteCPRegisters	O

M: Mandatory, O: Optional

Opcode	Name	Status	Opcode	Name	Status
0x00	NOP	M	0x93	HardWatch	O
0x80	OK	M	0x94	SWI	O
0x81	Error	M	0xA0	Undef	O
0x90	Stopped	O	0xA1	PrefetchAbort	O
0x91	SoftBreak	O	0xA2	DataAbort	O
0x92	HardBreak	O			

▲ 表二列出了 RealMonitor 所使用的 Host-to-Target 命令字代码。

2. 协议描述

异步控制

RealMonitor 能过 Escape 控制字实现对目标机(Target)的同步和复位处理。

连接到目标机

目标机的当前状态很复杂，可能正在运行，或是已经停止，或是处于异常状态，甚至处于接收命令包的过程中。为了使目标机处于一致的状态，主机必须使用 Escape 序列来同步和复位目标机。极端情形下，目标机处于极度混乱点，监视点，让目标处于一致的正常运行状态。

获取目标支持功能信息

从主机到目标发送代码为 0x01 的命令(GetCapabilities)请求，可以获取目标机所支持的调试功能的信息。目标机接收到GetCapabilities的命令之后，将所支持的命令以及其它信息格式化，置于内存，将该信息块的起始地址返回给主机。

主机一旦获得目标机所支持的命令功能集，以及一列内部指针信息之后，就可

现思想，既克服了硬件调试工具对于实时部件调试的不足，同时把繁杂的调试监控任务交给资源强大的主机助理去完成，从而克服了重量级软件调试代理所带来的巨大时延以及代码开销，使得RealMonitor即使用于拥有RTOS的终端产品也不致于对系统产生太多空间和时间的开销，使得终端产品也具备在线监控功能。

参考文献

ARM RMTarget Integration Guide.
http://www.arm.com.2000

ARM RMHost User Guide.
http://www.arm.com.2000