[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/50 (2006.01)
G06F 9/48 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200810119100.0

[43] 公开日 2009 年 1 月 7 日

[11] 公开号 CN 101339581A

[22] 申请日 2008.8.29
[21] 申请号 200810119100.0
[71] 申请人 北京中星微电子有限公司
地址 100083 北京市海淀区学院路 35 号世宁大厦 16 层
[72] 发明人 周雪珍 胡庆庆

[74] 专利代理机构 北京安信方达知识产权代理有限公司
代理人 龙 洪 霍育栋

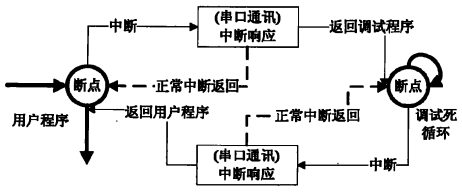
权利要求书 3 页 说明书 15 页 附图 4 页

[54] 发明名称

基于中断通讯的嵌入式系统的在线调试仿真方法

[57] 摘要

一种基于中断通讯的嵌入式系统的在线调试仿真方法，需要将运行着的用户程序挂起时，调试主机发送中断信号和准备调试的指令；嵌入式系统响应后，保存堆栈中的程序指针和用户程序断点现场信息，修改椎栈中的程序指针为监控通讯接口的一死循环的入口地址，将运行状态置为调试态后推栈，返回到该死循环；需要恢复用户程序运行时，调试主机发送中断信号和准备运行用户程序的指令；嵌入式系统响应后，将保存的程序指针和用户程序断点现场信息重新存入对应堆栈位置，将运行状态置为用户程序运行态后，推栈恢复用户程序断点现场，返回到用户程序被挂起的位置继续运行。本发明无需额外硬件即可实现在线调试仿真，且资源占用少。



1、 一种基于中断通讯的嵌入式系统的在线调试仿真方法，该嵌入式系统同时作为调试目标系统和仿真器并具有一通讯接口，可接收调试主机通过该通讯接口发送的中断信号及操作指令，该在线调试仿真方法包括:

需要将运行着的用户程序挂起时，所述调试主机向所述嵌入式系统发送中断信号后再发送准备调试的指令；所述嵌入式系统响应中断后，接收和执行所述指令，先保存堆栈中的程序指针和用户程序断点现场信息，再修改堆栈中的程序指针为监控所述通讯接口的一死循环的入口地址，将运行状态置为调试态后推栈，中断返回到该死循环;

需要恢复用户程序运行时，所述调试主机向所述嵌入式系统发送中断信号后再发送准备运行用户程序的指令；所述嵌入式系统响应中断后，接收和执行所述指令，将保存的程序指针和用户程序断点现场信息重新存入对应堆栈位置，将运行状态置为用户程序运行态后，推栈恢复用户程序断点现场，中断返回到用户程序被挂起的位置继续运行。

2、如权利要求 1 所述的在线调试仿真方法，其特征在于:

所述嵌入式系统处于调试态或用户程序运行态时，所述调试主机向所述嵌入式系统发送中断信号后发送指令对所述嵌入式系统的数据和/或寄存器进行读、写操作；所述嵌入式系统响应中断后，接收和执行所述指令，判断当前运行状态为调试态或用户程序运行态时，推栈恢复断点现场，中断返回到所述死循环或用户程序。

3、如权利要求 1 或 2 所述的在线调试仿真方法，其特征在于:

需要实现用户程序的单步运行时，在所述嵌入式系统处于调试态时，所述调试主机向所述嵌入式系统发送中断信号后发送准备单步运行的指令，所述嵌入式系统响应中断，接收、执行所述指令后，将保存的程序指针和用户程序断点现场信息重新存入对应堆栈位置，置位所述通讯接口中断标识产生软中断并将运行状态置为单步运行态后，推栈恢复用户程序断点现场，中断返回到用户程序，运行一条指令后响应所述软中断;

所述嵌入式系统响应软中断后，判断当前运行状态为单步运行态，先保存堆栈中的程序指针和用户程序断点现场信息，再修改椎栈中的程序指针为所述死循环的入口地址，将运行状态置为调试态后，中断返回到该死循环，完成了对用户程序的单步运行。

4、如权利要求 1 或 2 所述的在线调试仿真方法，其特征在于:

需要实现用户程序的多步运行时，在所述嵌入式系统处于调试态时，所述调试主机向所述嵌入式系统发送中断信号后发送准备单步运行的指令，并设置一步数值，所述嵌入式系统响应中断，接收、执行所述指令后，将保存的程序指针和用户程序断点现场信息重新存入对应堆栈位置，置位所述通讯接口中断标识产生软中断并将运行状态置为单步运行态后，推栈恢复用户程序断点现场，中断返回到用户程序，运行一条指令后响应软中断;

所述嵌入式系统响应软中断后，判断当前运行状态为单步运行态，将所述步数值减一后判断该步数值是否为 0，如果不为 0，再置位串口中断标识产生软中断，推栈恢复用户程序断点现场，中断返回到用户程序运行一条指令，再按相同的方式响应软中断；如果该步数值为 0，先保存堆栈中的程序指针和用户程序断点现场信息，再修改椎栈中的程序指针为所述死循环的入口地址，将运行状态置为调试态后，中断返回到该死循环，完成此次用户程序的单步或多步运行。

5、如权利要求 3 所述的在线调试仿真方法，其特征在于:

所述嵌入式系统运行状态用一运行状态标志来指示，所述调试主机向所述嵌入式系统发送的准备调试、准备运行用户程序和准备单步运行的指令，是分别将该运行状态标志置为准备调试态、用户程序准备运行态和准备单步运行态;

所述嵌入式系统执行所述调试主机发送的指令后，先根据所述运行状态标志判断当前运行状态，根据该当前运行状态来决定收到所述指令后应执行的操作。

6、如权利要求 5 所述的在线调试仿真方法，其特征在于，通过以下方式

实现断点设置:

在用户程序中的断点位置插入两条指令，一条指令将嵌入式系统运行状态置为准备调试态，另一条指令置位所述通讯接口中断标识产生软中断，用户程序运行到该位置时立即进入中断响应;

所述嵌入式系统进入中断响应后，判断当前运行状态为准备调试态，先保存堆栈中的程序指针和用户程序断点现场信息，再修改椎栈中的程序指针为所述死循环的入口地址，将运行状态置为调试态后，中断返回到该死循环。

7、如权利要求 5 所述的在线调试仿真方法，其特征在于:

所述嵌入式响应中断，接收调试主机发送的指令后，先判断收到的是否对所述运行状态标志的写操作指令，如果是，再判断其是否合法，如果合法再执行该指令，否则，不执行该指令; 所述指令合法是指: 当前运行状态为用户程序运行态时，写入值为“准备调试态”的值; 或者，当前运行状态为调试态时，写入值为“用户程序准备运行态”或“准备单步运行态”的值。

8、如权利要求 1 或 2 所述的在线调试仿真方法，其特征在于:

所述通讯接口为所述嵌入式系统支持的可与调试主机通讯的接口，所述调试主机发送的中断信号及命令为所述通讯中断信号及命令，所述死循环为无任何数据操作的所述通讯端口监控死循环程序。

9、如权利要求 1 或 2 所述的在线调试仿真方法，其特征在于:

所述通讯接口为调试仿真和用户程序所共享，所述嵌入式系统响应中断时还需对用户程序发起的通讯进行处理。

10、如权利要求 1 或 2 所述的在线调试仿真方法，其特征在于:

所述嵌入式系统的中断有优先级，将所述调试主机发送的中断设为最高优先级的中断，以调试其他优先级低于最高优先级的中断服务子程序。

基于中断通讯的嵌入式系统的在线调试仿真方法

技术领域

本发明涉及嵌入式系统，尤其涉及对嵌入式系统的在线调试仿真方法。

背景技术

嵌入式系统的调试仿真一般分为软件离线仿真和在线硬件调试仿真，软件离线仿真对仿真程序要求比较高，且得不到实际运行过程中硬件的实时信息，而嵌入式系统一般直接和硬件打交道，因而此方法受到局限。

在线硬件调试仿真可直接在运行硬件系统上进行在线调试仿真，传统的单片机仿真器硬件系统有的采用专用仿真的单片机。如 ARM 架构的微处理器（MCU），通常都有 JTAG 协议及硬件支持来实现在线调试仿真，能够得到运行过程中的实时信息，最大程度地提高了调试仿真效率。对于自身没有专用硬件支持的微处理器（如 51 系列 MCU）也有仿真器支持在线调试仿真，但使用仿真器无疑提高了开发成本，且受到仿真器的制约。有的采用两套单片机，一个单片机用于仿真，另一个单片机则用于目标单片机。这样同样也会提高开发成本。

发明内容

本发明要解决的技术问题是提供一种基于中断通讯的嵌入式系统在线调试仿真方法，无需额外硬件即可实现在线调试仿真，且资源占用少。

为了解决上述技术问题，本发明提供了一种基于中断通讯的嵌入式系统的在线调试仿真方法，该嵌入式系统同时作为调试目标系统和仿真器并具有一通讯接口，可接收调试主机通过该通讯接口发送的中断信号及操作指令，该在线调试仿真方法包括：

需要将运行着的用户程序挂起时，所述调试主机向所述嵌入式系统发送

中断信号后再发送准备调试的指令；所述嵌入式系统响应中断后，接收和执行所述指令，先保存堆栈中的程序指针和用户程序断点现场信息，再修改堆栈中的程序指针为监控所述通讯接口的一死循环的入口地址，将运行状态置为调试态后，中断返回到该死循环；

需要恢复用户程序运行时，所述调试主机向所述嵌入式系统发送中断信号后再发送准备运行用户程序的指令；所述嵌入式系统响应中断后，接收和执行所述指令，将保存的程序指针和用户程序断点现场信息重新存入对应堆栈位置，将运行状态置为用户程序运行态后，推栈恢复用户程序断点现场，中断返回到用户程序被挂起的位置继续运行。

进一步地，上述在线调试仿真方法还可具有以下特点：

所述嵌入式系统处于调试态或用户程序运行态时，所述调试主机向所述嵌入式系统发送中断信号后发送指令对所述嵌入式系统的数据和/或寄存器进行读、写操作；所述嵌入式系统响应中断后，接收和执行所述指令，判断当前运行状态为调试态或用户程序运行态时，推栈恢复断点现场，中断返回到所述死循环或用户程序。

进一步地，上述在线调试仿真方法还可具有以下特点：

需要实现用户程序的单步运行时，在所述嵌入式系统处于调试态时，所述调试主机向所述嵌入式系统发送中断信号后发送准备单步运行的指令，所述嵌入式系统响应中断，接收、执行所述指令后，将保存的程序指针和用户程序断点现场信息重新存入对应堆栈位置，置位所述通讯接口中断标识产生软中断并将运行状态置为单步运行态后，推栈恢复用户程序断点现场，中断返回到用户程序，运行一条指令后响应所述软中断；

所述嵌入式系统响应软中断后，判断当前运行状态为单步运行态，先保存堆栈中的程序指针和用户程序断点现场信息，再修改堆栈中的程序指针为所述死循环的入口地址，将运行状态置为调试态后，中断返回到该死循环，完成了对用户程序的单步运行。

进一步地，上述在线调试仿真方法还可具有以下特点：

需要实现用户程序的多步运行时，在所述嵌入式系统处于调试态时，所述调试主机向所述嵌入式系统发送中断信号后发送准备单步运行的指令，并设置一步数值，所述嵌入式系统响应中断，接收、执行所述指令后，将保存的程序指针和用户程序断点现场信息重新存入对应堆栈位置，置位所述通讯接口中断标识产生软中断并将运行状态置为单步运行态后，推栈恢复用户程序断点现场，中断返回到用户程序，运行一条指令后响应软中断；

所述嵌入式系统响应软中断后，判断当前运行状态为单步运行态，将所述步数值减一后判断该步数值是否为0，如果不为0，再置位串口中断标识产生软中断，推栈恢复用户程序断点现场，中断返回到用户程序运行一条指令，再按相同的方式响应软中断；如果该步数值为 0，先保存堆栈中的程序指针和用户程序断点现场信息，再修改椎栈中的程序指针为所述死循环的入口地址，将运行状态置为调试态后，中断返回到该死循环，完成此次用户程序的单步或多步运行。

进一步地，上述在线调试仿真方法还可具有以下特点：

所述嵌入式系统运行状态用一运行状态标志来指示，所述调试主机向所述嵌入式系统发送的准备调试、准备运行用户程序和准备单步运行的指令，是分别将该运行状态标志置为准备调试态、用户程序准备运行态和准备单步运行态；

所述嵌入式系统执行所述调试主机发送的指令后，先根据所述运行状态标志判断当前运行状态，根据该当前运行状态来决定收到所述指令后应执行的操作。

进一步地，上述在线调试仿真方法还可具有以下特点：通过以下方式实现断点设置：

在用户程序中的断点位置插入两条指令，一条指令将嵌入式系统运行状态置为准备调试态，另一条指令置位所述通讯接口中断标识产生软中断，用户程序运行到该位置时立即进入中断响应；

所述嵌入式系统进入中断响应后，判断当前运行状态为准备调试态，先

保存堆栈中的程序指针和用户程序断点现场信息，再修改椎栈中的程序指针为所述死循环的入口地址，将运行状态置为调试态后，中断返回到该死循环。

进一步地，上述在线调试仿真方法还可具有以下特点：

所述嵌入式响应中断，接收调试主机发送的指令后，先判断收到的是否对所述运行状态标志的写操作指令，如果是，再判断其是否合法，如果合法再执行该指令，否则，不执行该指令；所述指令合法是指：当前运行状态为用户程序运行态时，写入值为“准备调试态”的值；或者，当前运行状态为调试态时，写入值为“用户程序准备运行态”或“准备单步运行态”的值。

进一步地，上述在线调试仿真方法还可具有以下特点：

所述通讯接口为所述嵌入式系统支持的可与调试主机通讯的接口，所述调试主机发送的中断信号及命令为所述通讯中断信号及命令，所述死循环为无任何数据操作的所述通讯端口监控死循环程序。

进一步地，上述在线调试仿真方法还可具有以下特点：

所述通讯接口为调试仿真和用户程序所共享，所述嵌入式系统响应中断时还需对用户程序发起的通讯进行处理。

进一步地，上述在线调试仿真方法还可具有以下特点：

所述嵌入式系统的中断有优先级，将所述调试主机发送的中断设为最高优先级的中断，以调试其他优先级低于最高优先级的中断服务子程序。

本发明利用通讯中断方式实现用户程序与调试仿真死循环之间的切换，实现了用户程序断点设置、挂起用户程序、恢复用户程序运行、单（多）步执行用户程序等的在线调试仿真功能；另通过运行状态机机制，保证整个在线调试仿真过程健壮运行。对没有专用硬件支持的微处理器，只要原有系统支持以中断方式与调试主机通讯就可以采用本发明方法，无需额外硬件即可实现在线调试仿真，具有廉价、简单实用、通用性高等优点。

附图说明

图 1 是本发明实施例串口通讯过程的流程图;

图 2 是本发明实现用户程序的正常运行状态与调试仿真的死循环状态之间的转换的原理图;

图 3 是本发明实施例完整的串口中断响应的流程图;

图 4 是本发明实施例状态机及其转换的示意图;

图 5 是本发明实施例状态监控过程的流程图。

具体实施方式

本发明可以用于支持以中断方式与调试主机（如电脑）通讯的任意嵌入式系统的调试仿真。

下面以 MCU 的调试仿真为例，结合附图对本发明的具体实施方式进行详细的说明。

本实施例中，MCU 既作为仿真器也作目标机的 MCU 用，其实质是一种 ROM 监控器。要实现在线调试仿真，则必然需要有与 PC 主机通讯的模块，而 MCU 一般都支持串口中断通讯，串口通讯一般用于 MCU 与 PC 主机进行数据交换、命令传送等，也是一个常用的调试仿真手段。对于在线调试仿真系统而言，需要读写用户程序中的数据和寄存器信息，即利用串口中断通讯实现对 MCU 数据空间和寄存器的读写。

应该说的是，本实施例虽然以 MCU 为例，但可以适用于对支持中断通讯的任何嵌入式系统的调试仿真。本实施例虽然是以 PC 主机作为调试主机，但也可以采用其他如专用的调试设备作为主机。本实施例虽然是以串口通讯为例，但也可以通过其他的通讯接口如 USB 口等来产生中断和实现调试主机和嵌入式系统之间的通讯。

文中，除用于实现串口通讯过程的 UART_COM()函数，用于实现状态机监控过程的 Status_Machine_Monitor()函数，用于实现调试仿真死循环过程的 Debug_Task()函数，以及用于实现 MCU 状态切换的 MCU_Status_Switch()函数，其他的固件(Firmware)程序统称为用户程序。各函数将在下文中逐一进行详细的说明。

利用串口进行MCU的数据和寄存器读写需要保证MCU与PC主机准确的通讯握手，用户的读写命令能够准确无误地被解释和操作。本实施例中，PC主机与调试仿真目标系统的串口通讯过程，以格式为“通讯握手命令字+操作命令字+操作目标地址+操作目标值+通讯校验”的通讯协议将用户命令通过串口传达给MCU。其中操作目标地址是指定的数据或寄存器的读地址或写地址，操作目标值为要写入的数据。

MCU在响应串口中断后，需要在串口通讯过程中需要对该通讯协议包进行命令解释操作，本实施例通过UART_COM()函数来实现。请参照图1，在该串口通讯过程中执行以下处理：

MCU读取串口数据后，判断是否通讯握手信号指示为成功？如果否，直接结束；如果是，根据操作命令字判断该命令的类型：

如果是读数据命令，MCU 根据命令中的操作目标地址读出指定地址数据，以查询方式通过串口发送到PC主机，结束；

如果是写数据命令，MCU 先执行 Status_Machine_Monitor ()函数实现的一状态机监控过程，如监控通过，将给定数据写入到指定地址，然后再次读取指定地址的数据，以查询方式通过串口发送到PC主机，结束；

如果是读寄存器命令，MCU读取指定寄存器值，以查询方式通过串口发送到PC主机，结束；

如果是写寄存器命令，MCU将给定值写入到指定寄存器，然后再次读取指定寄存器值，以查询方式通过串口发送到PC主机，结束。

上述流程中，将写入数据再次读出是为了方便用户确认是否写操作成功。有了上述完整的串口通讯，就为在线调试仿真提供了基本通讯保障。

在线调试仿真时，要能将运行着的用户程序按用户需要随时停止、重新运行，以及进一步实现用户程序的单步运行和断点设置等功能。而对于没有特殊硬件支持的 MCU 系统，能够中断运行着的程序的只有中断方法，如果MCU在中断响应时，中断正常运行的用户程序后并不是直接正常中断返回，而是返回到一个调试仿真用的死循环中，不再正常执行被中断的用户程序，

就可使用户程序被“停止”挂起。用户程序挂起后，也可以利用中断，使MCU在中断响应时并不是正常中断返回到调试仿真死循环中，而是返回原来被中断的用户程序位置继续正常运行，其原理如图2所示。

对于中断有优先级的MCU，应将串口中断设为最高优先级，由于最高优先级的中断可以打断低优先级的中断，因此可以在线断点调试仿真任何低于最高优先级的用户中断服务子程序和用户主程序。

本实施例用 Debug_Task()函数来实现用户程序的挂起，其实质是一个无任何数据操作的串口监控死循环程序，在其他实施方式中，该死循环也可以用硬件来实现。在用户程序被挂起进入调试仿真过程后，保证 MCU 依然可以响应串口中断。PC主机通过串口通讯过程得到用户程序被挂起后的所有在线信息，或者使 MCU 接受用户命令。挂起用户程序需要中断资源，为了不占有更多的中断资源，本实施例利用已经开发好的串口通讯中断，同时用户如果需要利用该通讯中断进行用户程序工作也不会受到影响，只需在其中断响应中加入相关程序即可。

在线调试仿真而言的单步运行是指每执行一条指令就停止执行，在利用中断对用户程序运行挂起和继续运行的基础上，再利用 MCU 中断指令完整性原理可实现用户程序被挂起后恢复执行一条指令后再次挂起。所谓中断指令完整性原理就是：MCU在执行中断服务程序时（返回指令RETI之前），系统又有新的低于目前中断优先级或同等优先级的中断信号产生，MCU执行完RETI后并不是马上进入新的中断响应，而是会执行完这次中断响应时被中断位置的后面一条指令后再进入新的中断响应，以保证中断指令执行的完整性。根据这一原理，在需要单步运行时，MCU在恢复用户程序运行的串口中断服务程序中，主动置位串口中断标示位产生软中断，从而在中断返回到用户程序执行一条指令后，马上又响应串口中断将用户程序挂起，实现了单步运行。再通过增加步数计数器，利用单步原理即可实现多步运行。

用户在调试仿真时需要在指定的用户程序位置将程序挂起，即所谓断点设置，要实现断点设置只需在该位置将 MCU 运行状态修改为准备调试态，再插入一条串口软中断指令，用户程序立即进入串口中断将用户程序挂起。详见下文中的流程。

从上述描述可以看出，中断响应服务程序中需要进行不同的操作，具体执行哪种操作是根据当前的MCU运行状态来决定的。在MCU中设置有MCU运行状态MCU_RUN_STATUS标志，通过对该标志赋值来设置MCU的运行状态。根据上述原理和说明，有以下几种状态：用户程序运行态(Run)，准备调试态(Ready_Debug)，Debug_Task()函数运行而用户程序挂起的调试态(Debug)，用户程序准备运行态(Ready_Run)，准备单步运行态(Ready_Step)，以及单步运行态(Step)。需要说明的是，上述状态的名称完全可以不同，只要与本发明的该状态的作用相同即应视为同一状态。

下面将详细说明如何结合状态的设置和转换来实现上述各项功能。首先描述一下如何将运行着的用户程序挂起以进行调试仿真，以及如何恢复用户程序的运行，即用户程序运行态和调试态之间的切换。

要实现用户程序运行态与调试态的切换就需要知道用户程序运行的位置和 Debug_Task()函数的位置。Debug_Task()函数的位置是已知的，而用户程序运行时，MCU的程序指针(PC)即指示了程序的运行地址（下一条运行指令地址）。在中断时，MCU硬件中断系统响应时会将程序指针推入堆栈，在中断返回(执行RETI指令)时，从堆栈中弹出程序指针地址返回原来被中断的程序继续运行，利用中断的这一特性并加以利用，即可实现用户程序运行态与调试态之间的切换。

图3示出了MCU完整的串口中断响应流程图，为了清晰起见，下面对各项功能进行分别描述，分别描述时只写出与该功能相关的步骤。

要将运行着的用户程序挂起，即从用户程序运行态转换到调试态，PC主机在用户程序运行态下，先向 MCU 发起串口中断并在随后的串行通讯过程中发送写数据命令，将MCU_RUN_STATUS置为准备调试态(Ready_Debug)，MCU响应串口中断时，执行以下步骤：

步骤302，关中断，清串口中断标识；

步骤304，压栈保护断点现场；

这里保护的断点现场是用户程序的，如果原来在调试态，保存的断点现

场则是Debug_Task()的。

步骤 306，执行用户串口中断处理过程，这里是指执行除了用于在线调试仿真的其他串口程序，在线调试仿真可以和用户程序共享串口；

步骤 308，与 PC 主机进行在线调试仿真的串口通讯过程；

在从用户程序运行态转换到调试态的流程中，该步需要根据 PC 主机的命令将 MCU_RUN_STATUS 置为 Ready_Debug。这里是通过置运行状态标志来通知 MCU 准备调试，在其它实施例中也可以通过规定一个约定的指令来实现该通知，其它准备运行用户程序和准备单步运行的指令也同此。

步骤 314，判断 MCU_RUN_STATUS 是否为 Step 或 Ready_Debug，如果是，执行步骤 316，否则，执行步骤 328，这个分支与本流程无关，暂不描述；

图 3 中，步骤 314、328 和 340 对应的判断框内将 MCU_RUN_STATUS 简写成 MRS。

步骤 316，先将堆栈中的程序指针和用户程序断点现场信息保存到指定数据区；

本实施例使用数组 SAVE_STACK_BUF[]来保存用户程序从运行态切换到调试态的核心数据，以 MCS51 单片机为例，需要保存的堆栈信息包括堆栈指针 SP，通用寄存器组 R0-R7，通用寄存器 A 和 B，数据指针寄存器 DPTR，程序状态寄存器 PSW 和程序指针 PC 等，其总长度约为 17~20Bytes。

步骤 318，将堆栈中的程序指针修改为 Debug_Task()函数入口地址；

Debug_Task 就是一个无数据操作的死循环，无需为其另外开辟数据区保存堆栈中的现场数据，知道入口地址就可以切换到该过程。

步骤 320，将 MCU_RUN_STATUS 置为调试态(Debug)；

步骤 322，推栈，即将新的椎栈按原序推出；

步骤 324，开中断；

步骤 326，中断返回到 Debug_Task()函数。

这样，在中断返回时，MCU 将进入 Debug_Task()的死循环中运行，从而挂起了用户程序运行。在该过程中，调试仿真人员可以再次发起串口中断，

并通过串口通讯过程完成向 MCU 读写数据和寄存器值等调试仿真操作。在这些中断响应过程中，MCU 根据 MCU_RUN_STATUS 标志判断当前 MCU 运行状态为调试态时，在完成串口通讯过程后不进行状态转换相关的操作，正常返回到 Debug_Task()的死循环中。当然，本实施例在用户程序运行时，也可以通过中断通讯对 MCU 中的数据和/或寄存器进行在线读写。

请再参照图 3，要中断调试仿真用死循环，从调试态(Debug)切换到用户程序运行态(Run)时，PC 主机在调试态下，向 MCU 发起串口中断并在随后的串口通讯过程中发送写数据指令，将 MCU_RUN_STATUS 置为用户程序准备运行态(Ready_Run)，MCU 响应串口中断时，执行以下步骤：

步骤 302~步骤 314 与从用户程序运行态转换到调试态的流程基本相同，区别在于：步骤 304 中是对正在运行的调试死循环程序执行压栈保护断点现场的操作；另外，在步骤 308 中，MCU 在和 PC 主机进行的串口通讯过程中，是根据 PC 主机的命令将 MCU_RUN_STATUS 置为“Ready_Run”。

步骤 314 中，因为 MCU 判断 MCU_RUN_STATUS 不等于 Ready_Debug 和 Step，因此会执行以下步骤：

步骤 328，判断 MCU_RUN_STATUS 是否为 Ready_Run，如果是，执行步骤 330，否则，执行步骤 340，这个分支与本流程无关，暂不描述；

步骤 330，读取指定数据区保存的程序指针和用户程序断点现场信息，重新存入对应堆栈位置；

步骤 332，将 MCU_RUN_STATUS 置为运行态（Run）；

步骤 334，推栈恢复用户程序断点现场；

步骤 336，开中断；

步骤 338，中断返回到被中断的用户程序被挂起的位置继续运行。

请再参照图 3，要实现用户程序的单步运行，PC 主机在调试态(Debug)下，向 MCU 发起串口中断并在随后的串口通讯过程中发送写数据指令，将 MCU_RUN_STATUS 置为准备单步运行态(Ready_Step)，MCU 响应串口中断

时，执行以下步骤：

步骤302~步骤314与从调试态切换到用户程序运行态的流程基本相同，区别仅在于：在步骤308的MCU在和PC主机进行的串口通讯过程中，是根据PC主机的命令将MCU_RUN_STATUS置为Ready_Step;

步骤314中，因为MCU判断MCU_RUN_STATUS不等于Ready_Debug和Step，会执行步骤328，而在步骤328中，因为MCU判断MCU_RUN_STATUS不等于Ready_Run，会执行以下步骤：

步骤340，判断MCU_RUN_STATUS是否为Ready_Step，如果是，执行步骤342，否则，执行步骤360;

步骤342，读取指定数据区保存的程序指针和用户程序断点现场信息，重新存入对应堆栈位置;

步骤344，置位串口中断标识位以产生串口软中断;

步骤346，将MCU_RUN_STATUS置为单步运行态(Step);

步骤348，推栈恢复用户程序断点现场;

步骤350，开中断;

步骤352，中断返回到被中断的用户程序从被挂起的位置继续运行。

之后，用户程序单步运行即运行了一条指令后，就会响应串口软中断而进行串口中断响应，此时MCU_RUN_STATUS标志指示为单步运行态(Step)，在该串口中断响应过程中，MCU执行步骤302~306的过程与从用户程序运行态转换到调试态的流程相同；步骤308中，由于PC主机没有发出读写命令，MCU无需执行串口通讯过程;在步骤314中，MCU判断MCU_RUN_STATUS为Step后，就会执行步骤316~324，将MCU_RUN_STATUS置为Debug即调试态，中断返回到Debug_Task()函数的死循环中运行。在调试态时，用户可进行调试仿真操作，并可再次进行单步运行。

步骤360，推栈恢复断点现场;

步骤362，开中断;

步骤364，中断返回用户程序或Debug_Task()函数。

这个分支相当于在此次中断响应过程中不进行状态切换，直接返回到中断时运行的程序中继续运行。

要实现单步或多步运行时，可以在每执行一个单步后将所述计数值减一，直至减到 0 后停止单步运行从而实现单步或多步运行。具体地，在调试态下要实现多步运行时，PC 主机向 MCU 发送中断信号，并在之后的串口通讯过程将 MCU_RUN_STATUS 置为 Ready_Step，同时设置 MCU 中一步数计数器的步数值（单步运行可设置为 1），在该中断响应的后续处理与上述单步运行流程相同，也需要置位产生软中断和将 MCU_RUN_STATUS 置为 Step; 之后，MCU 执行一条指令后响应软中断，仍按上述单步运行流程执行到步骤 314，判断当前运行状态为 Step 后，先将步数值减 1，再判断该步数值是否为 0，如为 0，再执行步骤 316~324，中断返回后进入调试态，此时已实现了用户程序的多步运行。如果步数值不为 0，则再置位串口中断标识产生软中断，推栈恢复用户程序断点现场、开中断后，中断返回到用户程序，运行一条指令后，MCU 再按上述方式响应软中断。

要实现断点设置，只需在用户程序中的该位置插入两条指令，一条指令将 MCU 运行状态 MCU_RUN_STATUS 置为准备调试态(Ready_Debug)，另一条是置位产生串口软中断的指令，用户程序运行到该位置时，立即会进入串口中断，进入串口中断后 MCU 执行的流程与从用户程序运行态切换到调试态的流程基本相同，但无需通过串口通讯将 MCU_RUN_STATUS 置为 Ready_Debug(已经被置为 Ready_Debug)。执行完该次中断后，会将用户程序挂起，MCU 进入调试态，用户可进行调试仿真操作，并可恢复用户程序运行或进行用户程序的单步运行。

图 3 大方框中就是用户程序与 Debug_Task()之间切换的核心算法过程，文中将其定义为 MCU 状态切换过程 MCU_Status_Switch()。本发明通过将串口通讯过程和该 MCU 状态切换过程结合，用户就可以通过 PC 主机串口中断发调试仿真命令的方法实现对用户程序的在线调试仿真。其中涉及的 MCU 运行状态机及其转换方式如图 4 所示，图中线条的箭头方向为转换方向，线

条上的数字如“1”表示第 1 转换路径包括该线条，“2”表示第 2 转换路径包括该线条，等等。图 4 中包括以下 3 条合法的状态转换路径:

第 1 转换路径，当 MCU 处于 Run 状态时，PC 主机通过串口通讯发命令将 MCU_RUN_STATUS 置为 Ready_Debug，MCU 响应串口中断时，再将 MCU_RUN_STATUS 置为 Debug，实现 Run 到 Debug 的状态转换;

或者，当 MCU 处于 Run 状态，运行到用户程序中的断点位置时，产生串口软中断并将 MCU_RUN_STATUS 置为 Ready_Debug，MCU 响应串口软中断时，再将 MCU_RUN_STATUS 置为 Debug 状态，实现 Run 到 Debug 的状态转换。

第 2 转换路径，当 MCU 处于 Debug 状态时，PC 主机通过串口通讯发命令将 MCU_RUN_STATUS 置为 Ready_Run，MCU 响应串口中断时，再将 MCU_RUN_STATUS 置为 Run，完成 Debug 到 Run 的状态转换;

第 3 转换路径，当 MCU 处于 Debug 状态时，PC 主机通过串口通讯发命令将 MCU_RUN_STATUS 置为 Ready_Step，MCU 响应串口中断时，产生软中断并将 MCU_RUN_STATUS 置为 Step，中断返回后 MCU 运行一条用户程序指令，再次进入串口中断响应，在该次响应中将 MCU_RUN_STATUS 置为 Debug，实现 Debug 到 Run 单步运行一次后再进入 Debug 状态的转化。

除了上述状态机切换外，其他设置都是非法的，会导致整个系统崩溃。避免状态机非法切换的解决方法一：用户在 PC 主机上发送命令时，先得到 MCU 确切的运行状态后才能发出相应的合法状态切换指令，保证系统的状态正确切换；但该方法中 PC 主机与 MCU 之间的通讯需要一定时间，存在用户程序内部断点可能插入切换过程中的风险，造成系统紊乱，因此本实施例采用了在串口通讯过程中保证状态机正确切换的方法。

将如图 5 所示的用 Status_Machine_Monitor ()函数实现的状态机监控过程插入到 UART_COM()函数的数据写操作命令中(见图 1)，以保证系统运行状态机合法切换运行。MCU 收到数据写操作命令后，监控过程步骤如下:

步骤 810，判断写地址是否等于 MCU_RUN_STATUS 标志的地址，如果

不等于，继续数据写操作，结束；如果等于，执行步骤820;

步骤820，判断MCU_RUN_STATUS是否为Run，如果是，执行步骤830，否则执行步骤840;

步骤830，判断写入值是否等于Ready_Debug，如果是，继续数据写操作，结束，否则终止数据写操作，结束;

步骤840，判断MCU_RUN_STATUS是否为Debug，如果否，终止数据写操作，结束；如果是，执行步骤850;

步骤850，判断写入值是否等于Ready_Run，如果是，继续数据写操作，结束；否则，执行步骤860;

步骤860，判断写入值是否等于Ready_Step，如果是，继续数据写操作，结束；否则，终止数据写操作，结束。

本实施例基于中断通讯的在线调试仿真方法具有以下优点和特点:

A，占有资源少，该方法与用户程序共享一个串口中断，不占有额外的中断资源，该方法只多消耗数据空间约30Bytes，如果不需要对寄存器空间的访问，程序空间只需消耗约 400Bytes 代码空间，如果需要对如所有 MCS51 单片机所有寄存器进行访问，则需要的代码空间不超过2K。

B，无需专用硬件支持。对已经具有中断通讯功能的系统无需修改原有系统硬件，充分利用单片机系统硬件资源，且不影响用户对所有硬件资源的使用。

C，该方法具有通用性。只需要微处理器支持与PC主机的中断通讯即可实现，如串口，USB，并口等。

D，该方法在初始化串口通讯配置使能中断后，即可在线调试仿真除了与调试仿真用的通讯中断优先级相同的中断服务子程序以外的所有其他用户程序。

该方法与Keil公司在其Keil μVision IDE中提供的在线调试仿真方法的比较如下表，本发明方法的优势是很明显的。

Monitor-51 在线调试仿真方法	本发明中断通讯在线调试仿真方法
5K 字节的程序存储空间，用于存放监控程序	总体增加代码<2K 字节
必需的 256 字节的外部数据存储单元和可选的 5K 字节的跟踪缓冲区	数据空间增加不超过 30 字节
足够大的 von Neumann 架构的外部数据存储空间用于装载完整的用户应用程序	无需额外数据存储器存储用户程序，直接在程序空间实现全地址范围在线调试仿真
增加 6 字节的堆栈空间，用于用户程序的测试	无堆栈空间增加
串行口及一个用作波特率发生器的内部定时器	与用户程序共享一个（串口）通讯中断
用户程序大于 64K 字节需要分 Bank，则要求所有程序 Bank 空间按 von Neumann 架构连接	支持多 Bank 在线调试仿真，无需修改原来的程序空间连接架构
在线调试仿真开发受其相关协议等限制，灵活性差	用户可自行开发，灵活性强
适用于 8051 和及其衍生系列 MCU	适用于非硬件堆栈、中断通讯的微处理器

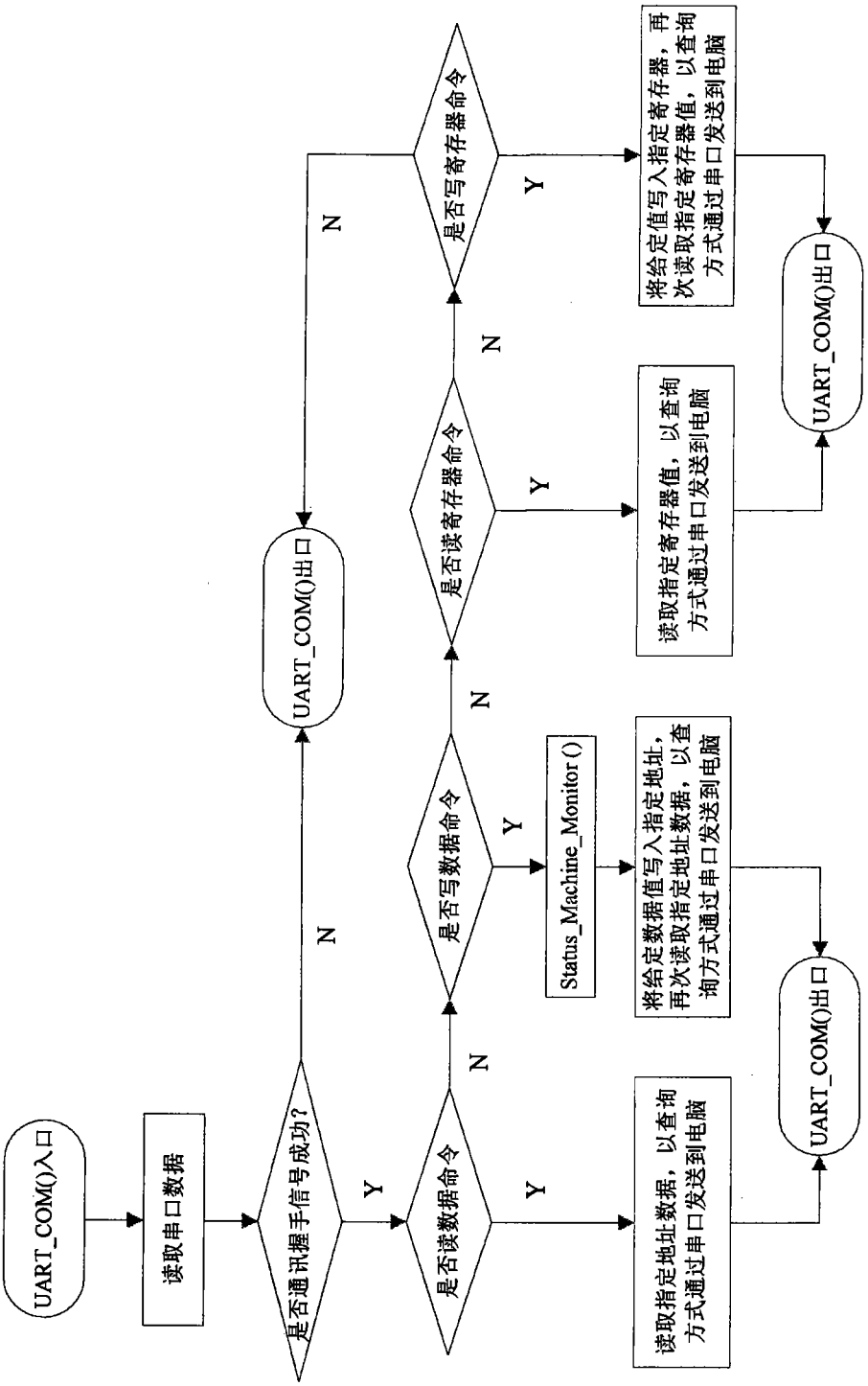

UART_COM()入口
读取串口数据
是否通讯握手信号成功?
Y
N
是否读数据命令
Y
N
读取指定地址数据，以查询方式通过串口发送到电脑
UART_COM()出口
是否写数据命令
Y
N
Status_Machine_Monitor ()
将给定数据值写入指定地址，再次读取指定地址数据，以查询方式通过串口发送到电脑
UART_COM()出口
是否读寄存器命令
Y
N
读取指定寄存器值，以查询方式通过串口发送到电脑
UART_COM()出口
是否写寄存器命令
Y
N
将给定值写入指定寄存器，再次读取指定寄存器值，以查询方式通过串口发送到电脑
UART_COM()出口

图 1

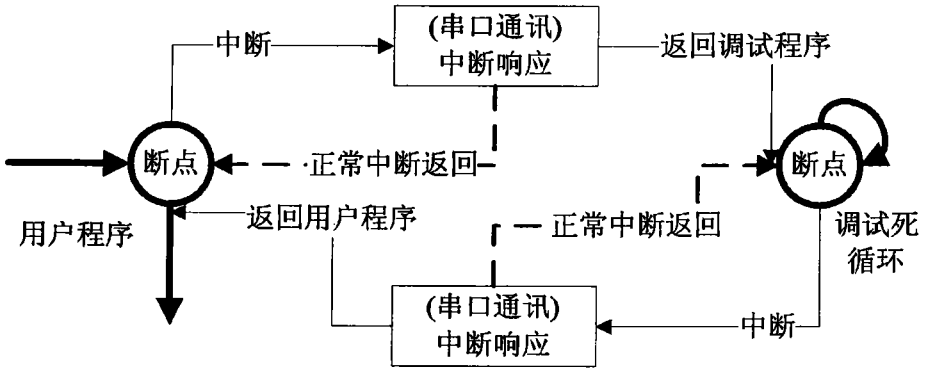

中断
(串口通讯)
中断响应
返回调试程序
断点
正常中断返回
断点
用户程序
返回用户程序
正常中断返回
调试死
循环
(串口通讯)
中断响应
中断

图 2

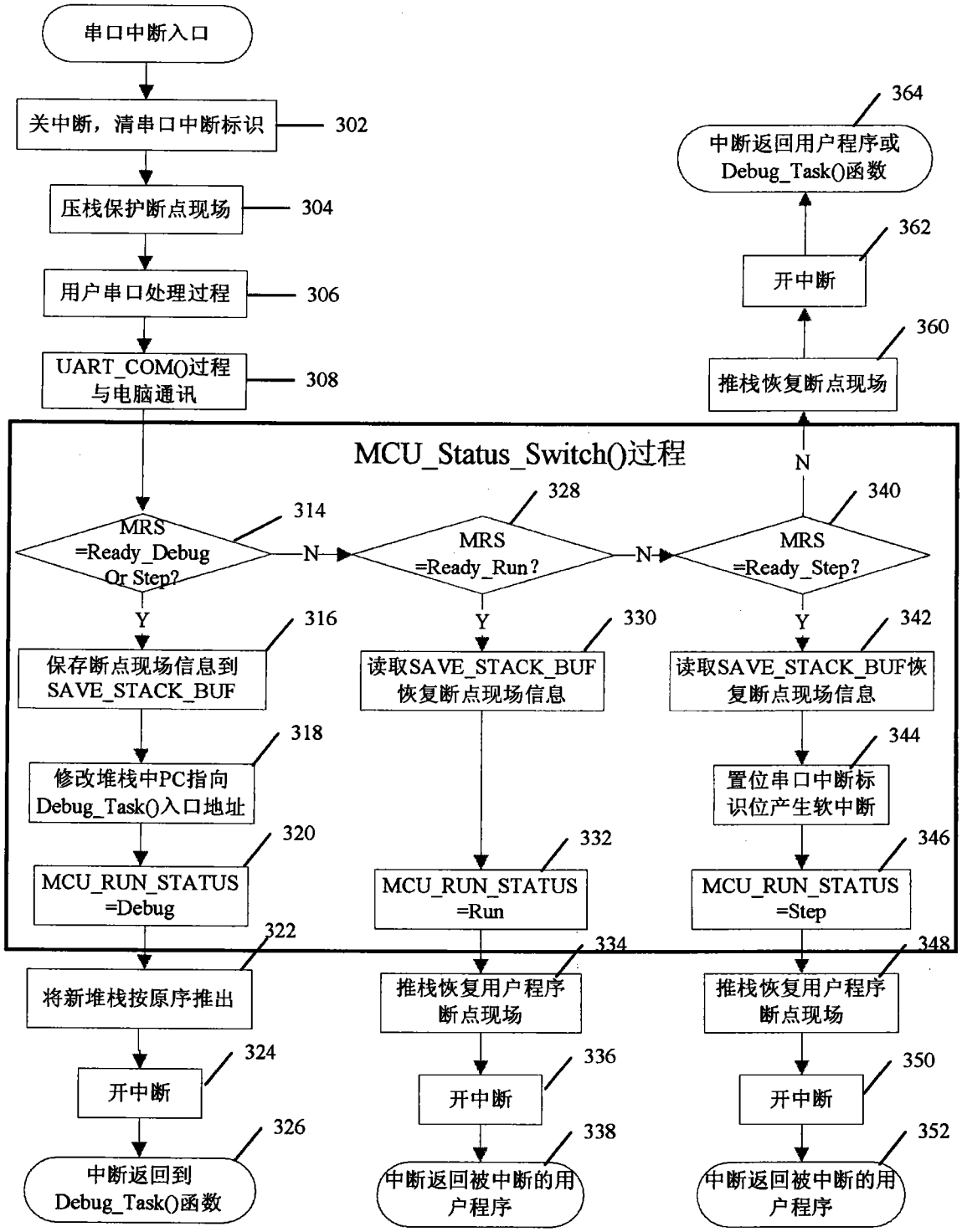

串口中断入口
关中断，清串口中断标识
302
压栈保护断点现场
304
用户串口处理过程
306
UART_COM()过程
与电脑通讯
308
MCU_Status_Switch()过程
MRS
=Ready_Debug
Or Step?
314
N
Y
保存断点现场信息到
SAVE_STACK_BUF
316
修改堆栈中PC指向
Debug_Task()入口地址
318
MCU_RUN_STATUS
=Debug
320
将新堆栈按原序推出
322
开中断
324
中断返回到
Debug_Task()函数
326
MRS
=Ready_Run?
328
N
Y
读取SAVE_STACK_BUF
恢复断点现场信息
330
MCU_RUN_STATUS
=Run
332
推栈恢复用户程序
断点现场
334
开中断
336
中断返回被中断的用
户程序
338
MRS
=Ready_Step?
340
N
Y
读取SAVE_STACK_BUF恢
复断点现场信息
342
置位串口中断标
识位产生软中断
344
MCU_RUN_STATUS
=Step
346
推栈恢复用户程序
断点现场
348
开中断
350
中断返回被中断的用
户程序
352
推栈恢复断点现场
360
开中断
362
中断返回用户程序或
Debug_Task()函数
364

图 3

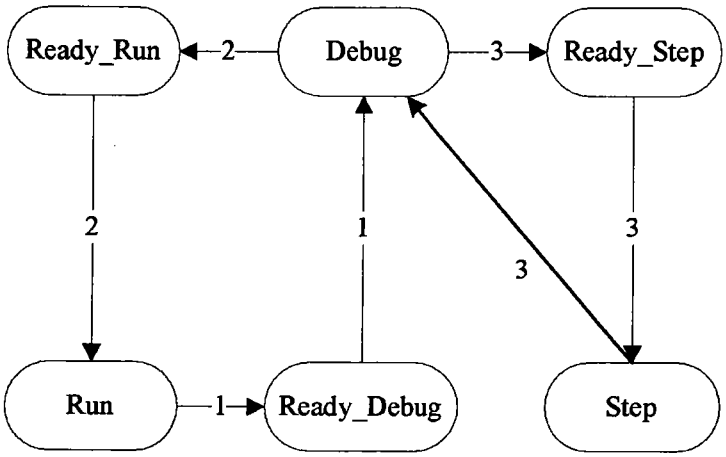

Ready_Run
2
Debug
3
Ready_Step
2
1
3
3
Run
1
Ready_Debug
Step

图 4

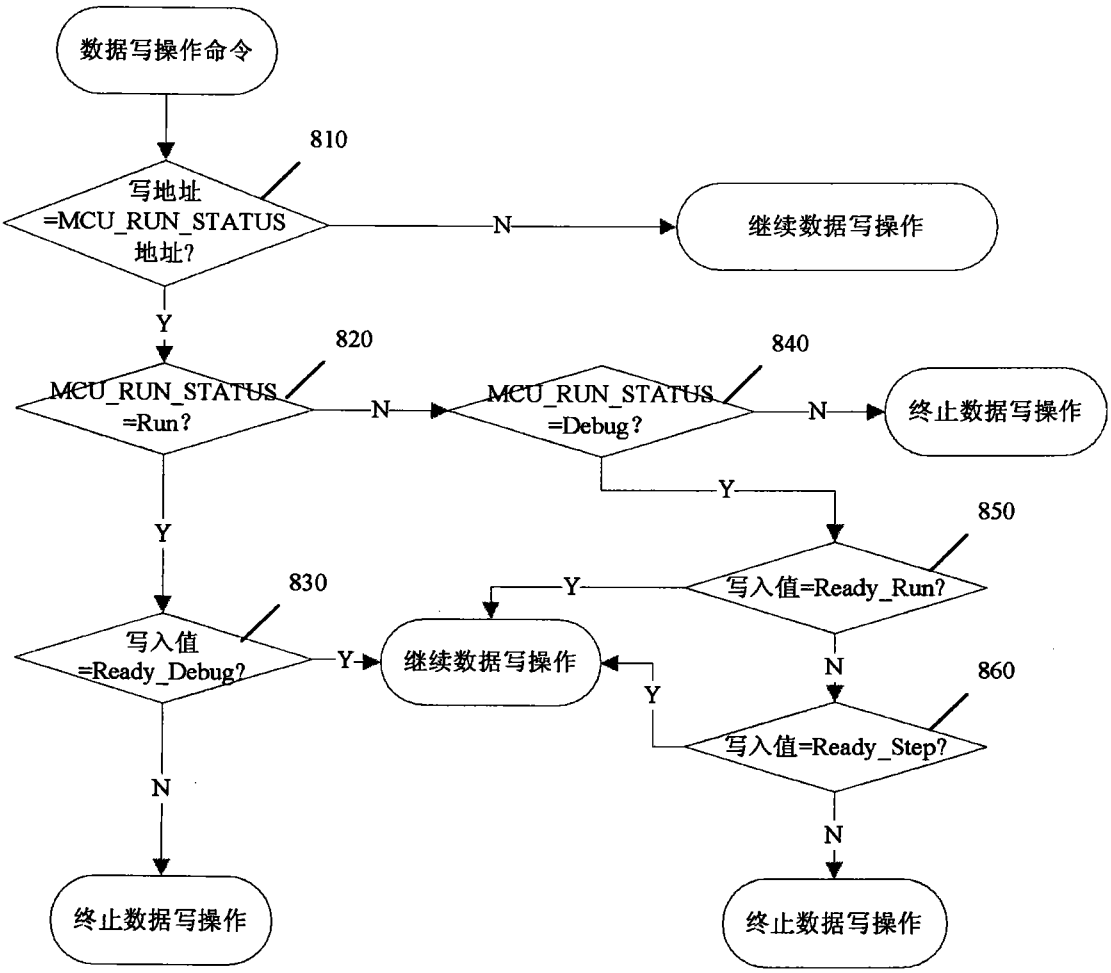

数据写操作命令
810
写地址
=MCU_RUN_STATUS
地址?
N
继续数据写操作
Y
820
MCU_RUN_STATUS
=Run?
N
840
MCU_RUN_STATUS
=Debug?
N
终止数据写操作
Y
Y
850
写入值=Ready_Run?
Y
830
写入值
=Ready_Debug?
Y
继续数据写操作
N
860
Y
写入值=Ready_Step?
N
N
终止数据写操作
终止数据写操作

图 5