(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 103019823 B
(45)授权公告日 2016.06.08

(21)申请号 201210534623.8
(22)申请日 2012.12.12
(73)专利权人 上海航天测控通信研究所
地址 200080 上海市虹口区新港街道天宝路881号
(72)发明人 高宇 刘成芳 丁宝华 顾少华
(74)专利代理机构 上海汉声知识产权代理有限公司 31236
代理人 胡晶
(51)Int.Cl.
G06F 9/46(2006.01)
G06F 9/52(2006.01)
G06F 9/54(2006.01)

(56)对比文件
CN 102591726 A,2012.07.18,
CN 101266561 A,2008.09.17,
CN 101872317 A,2010.10.27,
CN 102591658 A,2012.07.18,

审查员 赵晓敏

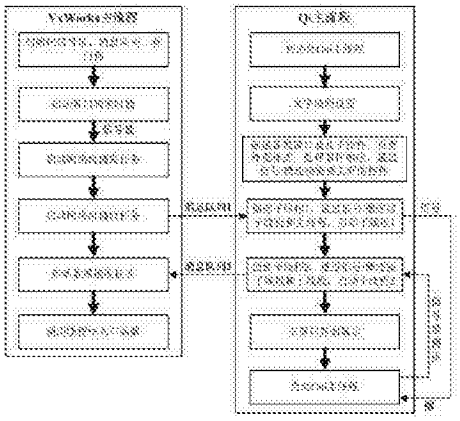
权利要求书1页 说明书3页 附图2页

(54)发明名称

实现VxWorks与Qt通信的消息队列方法

(57)摘要

一种实现VxWorks与Qt通信的消息队列方法，适用于VxWorks与Qt的混合开发，采用以下步骤：S1、在VxWorks任务中声明并创建一消息列队；S2、在Qt主线程中创建若干Qt子线程，并在若干Qt子线程的外部声明消息列队；S3、若干Qt子线程分别同对应的VxWorks任务通过消息列队机制进行通信。利用VxWorks消息队列阻塞接收机制，实现VxWorks任务与Qt子线程的同步，保证消息队列数据在任务或子线程读写时的原子操作，避免任务死锁。



CN 103019823 B

1.一种实现VxWorks与Qt通信的消息队列方法，适用于VxWorks与Qt的混合开发，其特征在于，采用以下步骤：

S1、在VxWorks任务中声明并创建一消息列队；

S2、在Qt主线程中创建若干Qt子线程，并在所述若干Qt子线程的外部声明所述消息列队；

S3、所述若干Qt子线程分别同对应的VxWorks任务通过消息列队机制进行通信；

其中在步骤S3中，所述通过消息列队机制进行通信包括：

S3.1、一第一Qt子线程阻塞接收所述消息列队，并发送给所述Qt主线程；

S3.2、所述Qt主线程释放一反馈数据至一第二Qt子线程；

S3.3、所述第二Qt子线程阻塞接收该反馈数据，并反馈给该消息列队；

S3.4、VxWorks任务阻塞接收该消息列队。

2.如权利要求1所述的实现VxWorks与Qt通信的消息队列方法，其特征在于，在步骤S1中包括，当所述消息列队为VxWorks特有数据类型时，需要先在Qt主线程中进行元类型的声明和注册，以使Qt能够识别。

3.如权利要求1所述的实现VxWorks与Qt通信的消息队列方法，其特征在于，在步骤S3中包括，第一Qt子线程接收到的消息列队通过信号/槽机制传输给Qt主线程进行画面显示。

4.如权利要求1所述的实现VxWorks与Qt通信的消息队列方法，其特征在于，在步骤S3.4中包括，Qt主线程的反馈数据通过信号量类同步传递给该第二Qt子线程进行消息列队的组织。

5.如权利要求1所述的实现VxWorks与Qt通信的消息队列方法，其特征在于，所述消息列队的数据缓冲区创建于VxWorks任务的上下文中。

实现VxWorks与Qt通信的消息队列方法

技术领域

[0001] 本发明涉及嵌入式开发领域，特别涉及一种实现VxWorks与Qt通信的消息队列方法。

背景技术

[0002] Qt是诺基亚开发的一个跨平台的C++图形用户界面应用程序框架。它提供给应用程序开发者建立艺术级的图形用户界面所需的所用功能。并且Qt是完全面向对象的，很容易扩展，并且允许真正地组件编程；使用Qt开发的软件，相同的代码可以在任何支持的平台上编译与运行，而不需要修改源代码。

[0003] VxWorks操作系统是美国WindRiver公司于1983年设计开发的一种嵌入式实时操作系统(RTOS)，是嵌入式开发环境的关键组成部分。VxWorks操作系统由于其良好的持续发展能力、高性能的内核以及友好的用户开发环境，在嵌入式实时操作系统领域占据一席之地。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通讯、军事演习、弹道制导、飞机导航等。

[0004] 为了结合Qt和VxWorks的优点，目前提出了VxWorks和Qt相结合的开发模式。

[0005] 然而，Qt用于图形界面处理的线程只能有一个，即主线程QApplication。如果VxWorks直接和Qt主线程通信，有可能导致Qt主线程因阻塞或延迟而无法连贯处理图形界面。为提高处理性能，可以创建Qt子线程专门负责与VxWorks的任务进行数据交换，再通过子线程将处理好的数据转发给Qt主线程。

[0006] 目前在VxWorks和Qt相结合的开发模式下，多采用信号量配合共享内存的方法实现二者之间的数据交换。但多个VxWorks任务或Qt子线程对同一块共享内存进行读、写操作时，由于任务或子线程之间的不同步，会破坏数据的完整性，导致读、写数据的错误。如果对共享内存数据进行临界区或互斥保护，尽管保证了数据在读、写时的完整性，但会在一定条件下引发任务优先级翻转，甚至出现任务死锁，从而使系统的实时性受到影响。

发明内容

[0007] 本发明针对现有技术存在的上述不足，提供一种实现VxWorks与Qt通信的消息队列方法，利用VxWorks消息队列阻塞接收机制，实现了VxWorks任务与Qt子线程的同步，保证了消息队列数据在任务或子线程读写时的原子操作，避免了任务死锁。

[0008] 本发明通过以下技术方案实现：

[0009] 一种实现VxWorks与Qt通信的消息队列方法，适用于VxWorks与Qt的混合开发，采用以下步骤：

[0010] S1、在VxWorks任务中声明并创建一消息列队；

[0011] S2、在Qt主线程中创建若干Qt子线程，并在若干Qt子线程的外部声明消息列队；

[0012] S3、若干Qt子线程分别同对应的VxWorks任务通过消息列队机制进行通信。

[0013] 较佳的，在步骤S3中，通过消息列队机制进行通信包括：

[0014] S1、一第一Qt子线程阻塞接收消息列队，并发送给Qt主线程；

[0015] S2、Qt主线程释放一反馈数据至一第二Qt子线程；

[0016] S3、第二Qt子线程阻塞接收该反馈数据，并反馈给该消息列队；

[0017] S4、VxWorks任务阻塞接收该消息列队。

[0018] 较佳的，在步骤S1中包括，当消息列队为VxWorks特有数据类型时，需要先在Qt主线程中进行元类型的声明和注册，以使Qt能够识别。

[0019] 较佳的，在步骤S3中包括，第一Qt子线程接收到的消息列队通过信号/槽机制传输给Qt主线程进行画面显示。

[0020] 较佳的，在步骤S4中包括，Qt主线程的反馈数据通过信号量类同步传递给该第二Qt子线程进行消息列队的组织。

[0021] 较佳的，消息列队的数据缓冲区创建于VxWorks任务的上下文中。

[0022] 本发明的有益效果是，实现了VxWorks任务与Qt子线程的同步，保证了数据交互时读、写的完整。可通过定义多个消息队列对多个任务分别进行通信，避免多个任务对同一块共享内存的竞争，有效避免了任务优先级翻转和任务死锁，保证了系统的实时性。另外，相比共享内存的全局性，消息队列数据缓冲区是在任务的上下文中创建的，它是局部的，有效降低了任务间的耦合度。

附图说明

[0023] 图1为本发明提供的一实施例的主流程图；

[0024] 图2为本发明中消息队列由VxWorks任务发送到Qt主线程的流程图；

[0025] 图3为本发明中消息队列由Qt主线程发送到VxWorks任务的流程图；

[0026] 图4为本发明的主流程时序分析图。

具体实施方式

[0027] 下面结合实施例对本发明作详细说明，本实施例在以本发明技术方案为前提下进行实施，给出了详细的实施方式，但本发明的保护范围不限于下述的实施例。

[0028] 本发明采用的技术方案是，创建所需数量的Qt子线程，分别同对应的VxWorks任务通过消息队列机制进行通信。VxWorks任务完成对消息队列的创建及初始化，Qt子线程外部引用该消息队列。任务和子线程根据需要既可作为消息队列的发送者，也可作为其接收者。其中，消息队列的数据类型可以自由定义，但VxWorks特有数据类型需要在Qt主线程中进行元类型METATYPE的声明和注册，从而使Qt能够得以识别。另外，Qt子线程接收到的消息队列可以通过信号/槽机制传递给Qt主线程进行画面显示，Qt主线程的反馈数据也可通过信号量类QSemaphore同步传递给Qt子线程进行消息队列的组织。

[0029] 请参考图1，本发明提供一较佳实施例加以说明。VxWorks主流程首先初始化了信号量、消息队列、看门狗这三个主要资源，随后启动了看门狗定时器和三个用户级任务。看门狗定时器回调函数通过释放信号量，为网络组播发送任务提供定时周期。网络组播接收任务通过套接字Socket接收到网络报文后，将报文数据发送到消息队列1。数据接收任务阻塞等待来自Qt子线程的消息队列2。VxWorks主流程最后调用Qt入口函数，启动Qt主流程。Qt主流程在进行一系列和图形界面相关的初始化后，创建了两个子线程。子线程1阻塞等待来

自网络组播接收任务的消息队列1，并将消息队列通过信号函数的参数发送给Qt主线程的槽函数，主线程即可根据数据信息进行画面显示。子线程2阻塞获取主线程释放的信号量，周期性的将反馈数据发送到消息队列2。通过主流程图，可以看到实施例中用到了四种任务及线程的通信方法。

[0030] 请参考图2，VxWorks任务发送数据到消息队列。Qt子线程1阻塞接收消息队列，并将该消息队列通过信号发送出去。Qt主线程在创建、启动子线程1并设置信号/槽连接后，槽函数会接收到信号，消息队列是通过信号/槽的函数参数进行传递的。

[0031] 请参考图3，Qt主线程创建并启动了定时器QTimer和子线程2，同时连接定时器超时信号到主线程槽函数，通过槽函数释放Qt信号量QSemaphore。子线程2阻塞等待来自主线程的Qt同步信号量，获取到该信号量后，子线程2发送反馈数据到消息队列。VxWorks任务阻塞接收该消息队列，并实现对信息的打印。

[0032] 请参考图4，可以看到主流程产生了6个用户级任务，其中，tmcast_Recv、pthr2、tmcast _Send和pthr1上的方框为消息队列发送、接收，实心三角形信号量发送，空心三角形为信号量接收，虚线为阻塞状态，波浪线为待执行状态，直线条为执行状态。tmcast_Recv是VxWorks网络组播接收任务，tmcast _Send是VxWorks网络组播发送任务、tQt_To_Vx是数据接收任务、pthr1和pthr2是Qt的两个子线程、tQtGui是Qt主线程。tmcast_Recv阻塞等待接收tmcast_Send发出的网络组播报文。收到报文后，tmcast_Recv立即发送报文到消息队列1，pthr1负责接收。另外，pthr2发送报文到消息队列2，tQt_To_Vx负责接收。

[0033] 以上公开的仅为本申请的一个具体实施例，但本申请并非局限于此，任何本领域的技术人员能思之的变化，都应落在本申请的保护范围内。

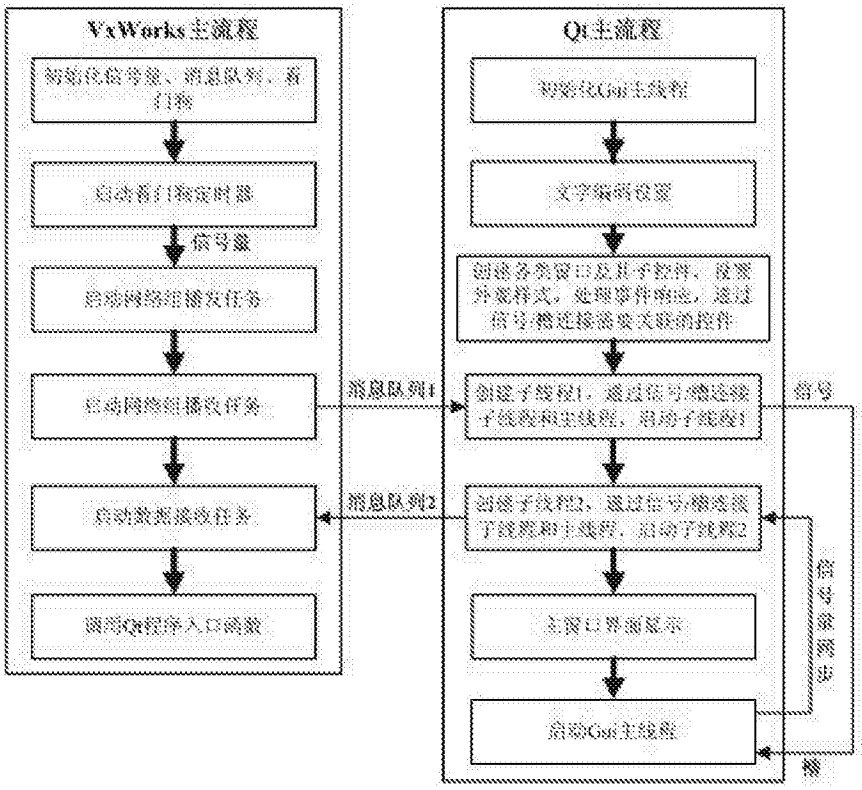

VxWorks主流程
初始化信号量、消息队列、看门狗
启动看门狗定时器
信号量
启动网络组播发任务
启动网络组播收任务
消息队列1
启动数据接收任务
消息队列2
调用Qt程序入口函数
Qt主流程
初始化Gui主线程
文字编码设置
创建各类窗口及其子控件，设置外观样式，处理事件响应，通过信号/槽连接要关联的控件
创建子线程1，通过信号/槽连接子线程和主线程，启动子线程1
信号
创建子线程2，通过信号/槽连接子线程和主线程，启动子线程2
信号量同步
主窗口界面显示
启动Gui主线程
循

图1

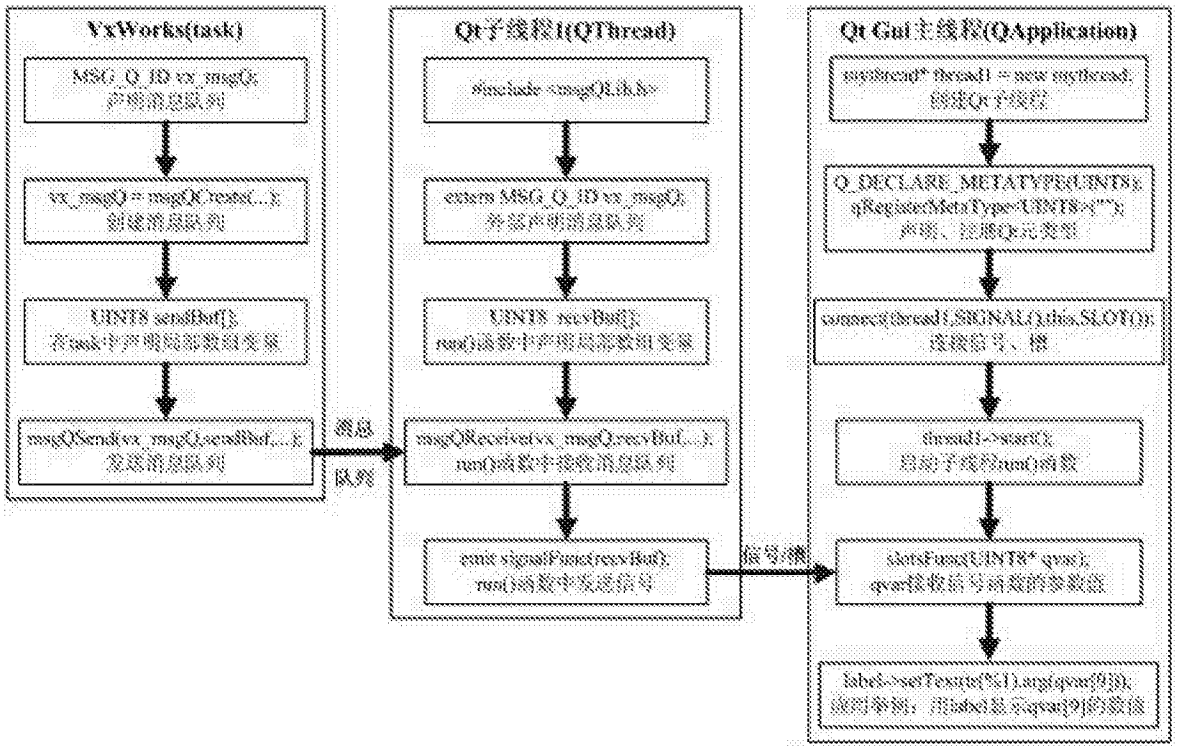

VxWorks(task)
MSG_Q_ID vx_msgQ;
声明消息队列
vx_msgQ = msgQCreate(...);
创建消息队列
UINT8 sendBuf[];
在task中声明局部数组变量
msgQSend(vx_msgQ,sendBuf,...);
发送消息队列
消息队列
Qt子线程1(QThread)
#include <msgQLib.h>
extern MSG_Q_ID vx_msgQ;
外部声明消息队列
UINT8 recvBuf[];
run()函数中声明局部数组变量
msgQReceive(vx_msgQ,recvBuf,...);
run()函数中接收消息队列
emit signalFunc(recvBuf);
run()函数中发送信号
信号/槽
Qt Gui主线程(QApplication)
mythread* thread1 = new mythread;
创建Qt子线程
Q_DECLARE_METATYPE(UINT8*);
qRegisterMetaType<UINT8*>("");
声明、注册Qt元类型
connect(thread1,SIGNAL(),this,SLOT());
连接信号、槽
thread1->start();
启动子线程run()函数
slotFunc(UINT8* qvar);
qvar接收信号函数的参数值
label->setText(tr("%1").arg(qvar[9]));
假设参数，用label显示qvar[9]的值

图2

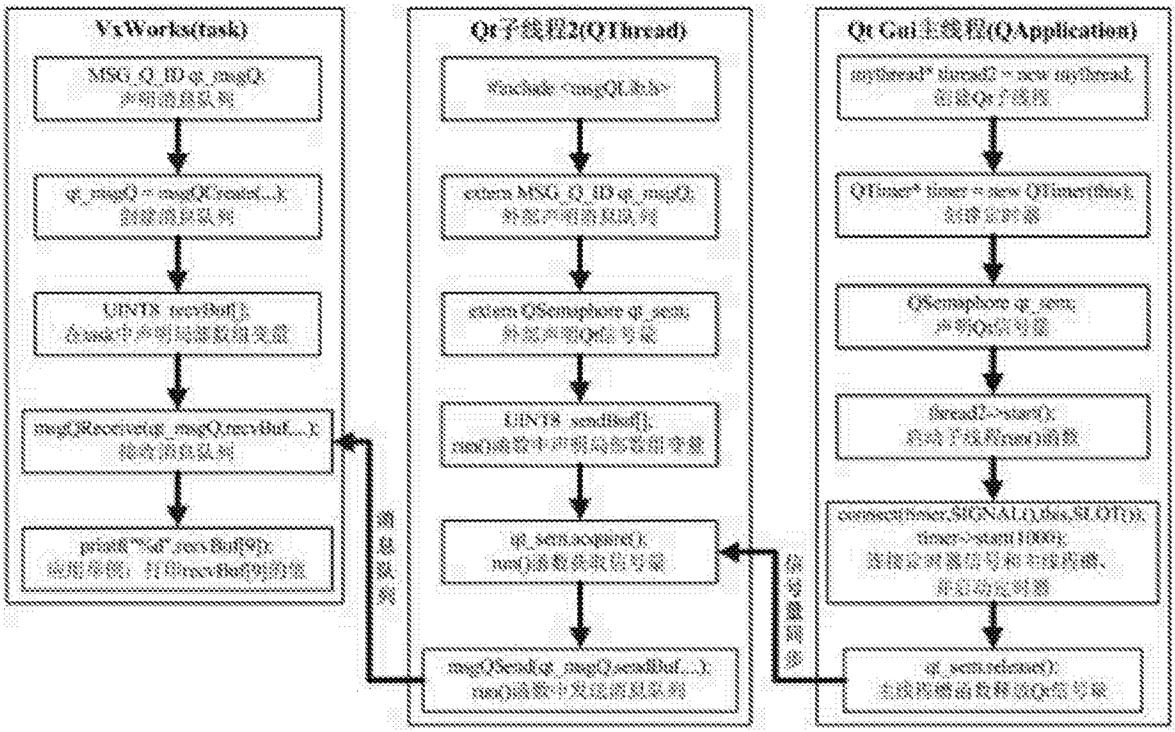

VxWorks(task)
Qt子线程2(QThread)
Qt Gui主线程(QApplication)
MSG_Q_ID qt_msgQ;
extern MSG_Q_ID qt_msgQ;
extern QSemaphore qt_sem;
QSemaphore qt_sem;
thread2->start();
qt_sem.acquire();
qt_sem.release();

图3

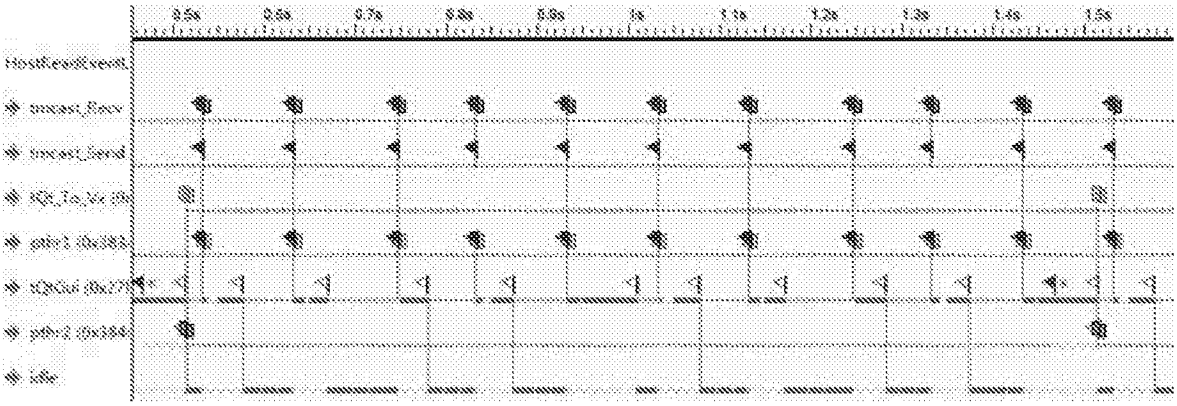

0.5s
0.6s
0.7s
0.8s
0.9s
1s
1.1s
1.2s
1.3s
1.4s
1.5s
idle

图4