昆明学院学报　2017, 39(3):49~53　ISSN 1674-5639　CN 53-1211/G4
Journal of Kunming University

轻量级 MQX 工程框架的研究与应用

文　瑾,赵　卿,景　东,刘昌亮
(昆明学院 信息技术学院,云南 昆明 650214)

摘要:依据软件工程的思想和构件的基本原理,在分析比较标准嵌入式实时操作系统 MQX 和轻量级 MQX (MQXL-ite)的工程组织结构和共性的基础上,对 MQX-Lite 的工程代码进行分析与整理,给出一种不依赖于开发平台、结构清晰、文件命名及布局规范的 MQX-Lite 的工程框架. 结合具体芯片给出板级支持包 BSP 的配置方法,最后对应用任务程序代码与标准 MQX 不同的使用方法加以详细说明,并对所创建的嵌入式实时操作系统 MQX 的轻量级工程框架进行适当修改,就可从现处的 KDS 开发平台移植到 IAR 和 KEIL 平台.

关键词:嵌入式实时操作系统;轻量级 MQX;工程框架;BSP
中图分类号:TP391　**文献标识码**:A　**文章编号**:1674-5639(2017)03-0049-05
DOI:10.14091/j.cnki.kmxyxb.2017.03.012

Research and Application on MQX Lightweight Engineering Framework

WEN Jin, ZHAO Qing, JING Dong, LIU Changliang
(College of Information Technology, Kunming University, Kunming, Yunnan, China　650214)

Abstract: According to the basic principles of software engineering ideas and components, on the basis of comparing standard MQX RTOS with lightweight MQX (MQX-Lite) project in organizational structure and common, after analyzing and sorting MQX-Lite project code, an independent MQX-Lite engineering framework conformed to the clear structure, file naming and content layout specifications was given. Combined with the specific chip, the configuration of the board support package BSP was acquired. Then the application task process code and the applications that distinguishes from the standard MQX were stated in details. Appropriate modifications has been made to the lightweight engineering framework of the embedded real-time operating system MQX so as to transplant the existing KDS development platform to the IAR and KEIL platform.

Key words: embedded RTOS; MQX-Lite; engineering framework; BSP

MQX(Message Queue eXecutive,消息队列执行)是一款嵌入式实时操作系统(RTOS),1989 年由 Precise Software Technologies 公司开发,2009 年被飞思卡尔半导体公司(现为恩智浦公司,NXP)收购,公司组织强大的团队对 MQX 进行维护并提供免费技术支持和开源源代码[1]. MQX 具有架构清晰、内核精简、外设驱动丰富、开发工具成熟、实时性强、性价比高、应用范围广等特点. 与嵌入式 Linux 相比较,虽然 Linux 的 OpenGL 和 MMU 功能强大,但占用资源较多,而 MQX 的内核较精简,其实时性和效率较高,更适用于家用电器、工业控制、医疗电子等领域[2]. 与 μCOS 相比,核心大小相近,但 MQX 的维护团队强大,能提供众多的驱动,实际应用方便,并降低了用户开发成本[3].

嵌入式轻量级 MQX 实时操作系统(MQX-Lite)是标准 MQX 的轻量级版本,其内核精简,只包含标准 MQX 内核的必要组件和某些组件的轻量级版本,是标准 MQX 的一个真子集. 主要针对资源有限(FLASH ROM 或 RAM)的 MCU,可安装在 Freescale Kinetis K 系列、L 系列与 E 系列的 MCU 中,能在最小 4 KB 的 RAM 空间运行[1].

良好框架结构是规范工程的基本,设计出好的框架能更好地展示操作系统的优点[4]. 但目前业界对 MQX-Lite 下的编程框架没有统一标准,导致程序

收稿日期:2017-04-19
基金项目:2016 年云南省大学生创新创业训练计划建设资助项目"基于 ZigBee 无线 RFID 读写器研究与设计".
作者简介:文瑾(1963—),男,云南昆明人,副教授,主要从事嵌入式系统及物联网技术研究.

代码可复用性和可移植性较差,降低了程序开发效率,增加了后期维护成本. 为解决这些问题,在对 MQX 架构机制充分分析后,结合软件工程中可靠性、可复用性和可移植性的思想,以及嵌入式构件封装原则和封装规范[5-6],在对标准嵌入式实时操作系统 MQX 和 MQX-Lite 的工程充分分析比较的基础上,给出了 MQX-Lite 的工程框架的设计原则和方法、BSP 包的配置过程等. 本文提出的框架以构件的方式组织文件,工程结构清晰、易于复用和移植. 因此,本文对 MQX 架构机制的分析,以及 MQX-Lite 工程框架的设计,对 MQX 操作系统的推广和应用有着重要的作用.

1 轻量级 MQX 的内核、PSP 与 BSP 分析

标准 MQX 嵌入式实时操作系统主要包括内核(kernel)、处理器支持包(Processor Support Package, PSP)和板级支持包(Board Support Package, BSP)3 部分.

1.1 MQX 内核

操作系统的质量由内核的性能和稳定性决定,内核是 MQX 中最基本的部分,包含系统初始化、任务管理、任务调度、自启动任务的创建、中断系统、任务间通信等. 标准 MQX 的内核由核心组件和可选组件组成,采用微内核结构设计. 其中核心组件是 MQX 内核的必选项,它提供操作系统最基本的服务. 可选组件为应用程序提供扩展服务,相对于核心组件,可选组件可以实现更全面、更丰富的功能[7].

轻量级 MQX(MQX-Lite)不随标准的 MQX 一起独立发布,而是作为处理器专家(Processor Expert, PE)的一个组件提供,目前其最新版为 MQX-Lite V1.1.0. PE 提供的 MQX-Lite 由轻量级内核与 PSP 组成,它不包含标准 MQX 中所需的 BSP,而用 PE 提供的相应逻辑驱动设备和组件来实现板级初始化及对外设 I/O 的访问. MQX-Lite 的内核是在标准 MQX 的内核的基础上适当精简得到的,精简前后的差异如图 1 所示[1](图中带删除线的为裁减的功能,下图同).

由图 1 可知,MQX-Lite 内核裁减了标准 MQX 内核中诸如 I/O 子系统、队列、任务队列调度、日志、信号量和 RR 调度等功能,而对于定时器、事件、消息、信号量只保留了它们的轻量级版本,表 1 列出了 MQX-Lite 内核中组件的功能和作用.

图1 标准MQX内核与MQX-Lite内核的差异

表 1 MQX-Lite 内核组件的功能和作用

类型	内核组件	功能和作用
核心组件(必选)	初始化	对系统初始化
	内核存储服务	分配和释放可变大小的内存块
	自启动任务	创建自启动任务
	FIFO 调度	基于优先级的抢占式 FIFO 任务调度
	任务管理	创建、管理和终止任务
	轻量级信号量	轻量级信号量同步机制
	任务错误	任务错误代码,异常处理,实时测试
可选组件	中断	对硬件中断进行处理
	轻量级事件	用轻量级事件实现任务间的同步机制
	轻量级消息	实现任务间通信
	互斥量	用互斥量实现任务间的同步机制
	内核日志	记录 MQX-Lite 的活动
	轻量级定时器	用于定时或周期性地调用应用函数

1.2 处理器支持包 PSP

PSP 内含与 CPU 相关的一系列代码,涉及 MQX 操作系统中断控制、上下文切换等与 CPU 结构相关的功能. 本文所使用的测试样例硬件平台的主控芯片使用 NXP 的 KL25 芯片,其 CPU 是 Cortex-M0 +,可把轻量级工程文件夹下有关内容复制到工程架构的 PSP 文件夹中使用[8].

1.3 板级支持包 BSP

板级支持包(BSP)是硬件与操作系统之间的桥梁,它既涉及对硬件的控制和处理,又涉及操作系统的调用,主要包括外设驱动接口,以及硬件设备底层系统功能代码等. 其中外设的设备驱动代码只与设备有关,而与设备所在的硬件板无关. BSP 包是嵌入式应用开发的关键环节,开发人员应对它有一个清晰的认识和理解[9]. 在本文第 3 部分将给出 BSP 的开发.

2 轻量级 MQX 工程分析与设计

MQX-Lite 实现了标准 MQX 的基本功能,如多任务处理、中断处理、共享资源的同步访问、基于优先级的抢占式任务调度等. 除图 1 所示差异外,与标

准 MQX 还存在以下差异[1]:

1)MQX-Lite 任务在编译时需预先分配好,它不支持动态创建任务;

2)MQX-Lite 使用第三方外设驱动或 PE 提供的逻辑驱动来访问外部 I/O 设备,它不含 BSP 或任何外设驱动;

3)MQX-Lite 不含 MFS(Moose File System,MooseFS、分布式文件系统)、RTCS(Real-Time TCP/IP Communication Suite)组件、MQX Shell 库;

4)标准 MQX 的初始化函数_mqx()在 MQX-Lite 中被拆分为_mqxlite()和_mqxlite_init().前一个负责创建空闲任务和自启动任务,后一个负责对任务就绪队列、内核数据结构、中断栈、同步任务所需的轻量级信号量等初始化.

2.1　设计方式

设计 MQX-Lite 工程分为以下两种方式[1].

1)PE(Processor Expert)方式:使用开发平台 CodeWarrior 中的处理器专家(PE),进行相关配置后就可创建一个 MQX-Lite 工程.该方式的优点是可以利用 PE 提供的各种组件来辅助开发,缺点是应用开发过程中不能脱离 PE,不利于工程的复用和移植.

2)非 PE 方式:不依赖 PE,独立配置与开发 MQX-Lite 工程.由于脱离了 PE,工程能方便地在不同开发环境中移植和复用,需要开发独立驱动或使用第三方的驱动来实现对外设 I/O 的访问.因此本文采用该方式设计轻量级工程框架.

2.2　设计框架

本文所设计的 MQX-Lite 工程框架,是以苏州大学飞思卡尔嵌入式中心设计的工程框架 AMQXFW(All-In-One MQX FrameWork)为基础修改而来.它保留了 AMQXFW 框架的基本架构,仅将原来标准 MQX 文件夹下的内容替换为由 PE 方式中生成的 MQX-Lite 工程的相应内容,且针对 MQX-Lite 没有 bsp 及 I/O 子系统的特点,参照标准 MQX 中的 bsp 代码及 PE 方式所生成工程中的相关内容,创建自己的 bsp 代码,并使用独立开发的外设驱动程序来访问外部 I/O 设备[10].MQX-Lite 工程框架的目录结构如图 2 所示.

3　设计轻量级 MQX 的 BSP 包

BSP 是一组与用户硬件开发板相关的程序代码,主要是在 MQX 操作系统启动过程中对硬件系统,如时钟、中断控制等的初始化,在应用程序中可能用到的外设模块的初始化过程也被安排在 BSP 中.官方发布 MQX 时,都为每款芯片所设计的 Demo 板提供专门的 BSP 工程参考样本,用户在深入了解修改的基本准则、方法与步骤后,可对官方提供的 BSP 进行修改,用于自己所设计的硬件电路,这也是设计 BSP 的常用方法之一[11].本文所采用的开发板是 ARM Cortex-M0+内核的 MKL25Z128VLK4 芯片(简称 KL25).MQX-Lite 的 BSP 包文件及内容如表 2 所示.

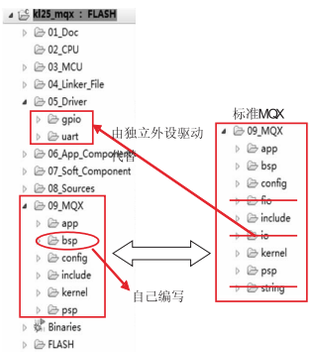

图2　MQX-Lite工程框架的目录结构

表 2　MQX-Lite 的 BSP 包文件及内容

BSP 包	名称	内容
bsp.h	总头文件	硬件相关文件
init_bsp.c	板级支持包初始化文件	硬件平台初始化时所用到的函数
mqx_init.c	MQX 初始化信息文件	缺省 MQX INITIALIZATION 结构体的定义
mqx_init.h	MQX 初始化信息文头件	系统初始化定义

3.1　设计 bsp.h 文件

该文件包含了#include 语句,使应用程序能访问主板的资源和设备驱动的 API.它还声明了公共 BSP 函数原型,分离出来供应用程序和 IO 驱动程序使用(例如特定板引脚初始化函数,<gpio.h>和<uart.h>),以及处理器相关的常数宏定义,具体如下.

1)根据所使用的是 KL25 开发板,对内核时钟频率、系统时钟频率、SYSTICK 的频率、SYSTICK 的周期等的宏定义:

```
#define BSP_CORE_CLOCK　(48000000UL)
#define BSP_SYSTEM_CLOCK　(48000000UL)
#define BSP_ALARM_FREQUENCY　(200)
#define BSP_ALARM_RESOLUTION　(1000/BSP_ALARM_FREQUENCY)
```

2)对系统滴答定时器中断及中断优先级级别

定义：

```
#define BSP_TIMER_INTERRUPT_VECTOR INT_SysTick
#define BSP_TIMER_INTERRUPT_PRIORITY 0
```

3)最后执行初始化外设系统：

```
extern uint_32 _bsp_enable_card ( void )
```

3.2 设计 init_bsp.c 文件

用于实现 MQX 中断处理系统的环境初始化、Systick 初始化，具体如下.

1)初始化外设系统，获得对内核数据区的访问，设定 CPU 类型：

```
register KERNEL_DATA_STRUCT_PTR kernel_data;
_GET_KERNEL_DATA(kernel_data);
_mqx_set_cpu_type(MQX_CPU);
```

2)初始化中断系统，向 kernel 注册systick 中断，注册systick 中断服务函数到中断管理模块，systick 中断服务例程.

```
result = _psp_int_init(FIRST_INTERRUPT_VECTOR_USED,LAST_INTERRUPT_VECTOR_USED);
_time_set_timer_vector(BSP_TIMER_INTERRUPT_VECTOR);
if (_int_install_isr(BSP_TIMER_INTERRUPT_
VECTOR,(void (_CODE_PTR_)(pointer))_
bsp_systick,NULL) = = NULL)
staticvoid_bsp_systick(pointer dummy)
{   _time_notify_kernel();
}
```

3)初始化系统定时器 systick(产生 tick)，完成关闭 SYSTICK，清除当前值计数器，设定倒计时计数值，设定 SysTick 优先级，使用内核时钟，倒计时到 0 时产生 SYSTICK 中断，使能 SYSTICK，登记系统时钟系统配置.

```
staticvoidsystick_init(void)
{   SYST_CSR =0;SYST_CVR =0;
    SYST_RVR = BSP_CORE_CLOCK/BSP_ALARM_FREQUENCY -1;
    SCB_SHPR3 | = SCB_SHPR3_PRI_15
    (CORTEX_PRIOR(BSP_TIMER_INTERRUPT_PRIORITY));
    SYST_CSR = 7;
}
_GET_KERNEL_DATA(kernel_data);
```

3.3 设计 mqx_init.c 文件

定义 MQX 初始化参数，预先分配系统占用空间. 由于精简了动态存储管理系统，取而代之的是使用静态分配存储空间的方式. 这种方式适合于 RTOS 的应用，其原因是：1)预先分配存储空间，可以减少运行时错误. 若内存空间不够，在编译时就可发现；2)动态分配存储空间，效率低，同时提高了系统的复杂度. 具体完成在轻量级初始化结构体 MQX_init_struct 中，有处理器数量、内核数据区开始、任务模板清单等.

```
const MQXLITE_INITIALIZATION_STRUCT MQX_init_struct =
{   1,                          //处理器数量
    _KERNEL_DATA_START,         //内核数据区开始位置
    _KERNEL_DATA_END,           //内核数据区结束位置
    1,                          //最大硬件中断级别
    sizeof(mqx_interrupt_stack),  //中断栈大小
    mqx_interrupt_stack,          //中断栈位置
    sizeof(mqx_idle_task_stack),  //空闲栈大小
    mqx_idle_task_stack,          //空闲栈位置
    (TASK_TEMPLATE_STRUCT_PTR) &MQX_template_list[0]  //任务模板清单
};
```

3.4 设计 mqx_init.h 文件

该文件包含单片机缺省的 MQX 初始化结构体，以便简单的应用程序或通过使用缺省值(在 target.h 中定义)并无需定义一个结构体的应用程序使用. 应用程序能够创建一个新的 MQX 初始化结构体，该结构使用一些缺省值和其他的给定值. 它包含 MQX 系统栈、任务模板结构体、系统中的最低任务优先级的定义、中断起止位置等.

4 测试与应用

4.1 测试环境

本文中的 MQX-Lite 工程以恩智浦公司的 KDS 3.0 作为嵌入式软件集成开发平台. 硬件平台为苏州大学恩智浦嵌入式中心设计的 SD-FSL-KL25-EVB 评估板(简称 KL25 开发板)，该评估板采用 ARM Cortex-M0 + 内核的 MCU. 该板按工业级标准

设计,可以用于实际工业控制项目开发,本文中的测试工程已在该硬件平台下调试通过[5,11].

4.2　应用结果与分析

本测试工程在 09_MQX/app 目录中添加相关的用户应用任务程序代码,包含 4 个任务函数,即三色灯闪烁任务函数 task1( ),task2( )和 task3( ),使用串口 1 发送数据到 PC 机的 task4( )任务函数等.

测试结果. 任务 1:红灯每隔 2 s 亮 1 s;任务 2:绿灯每隔 2 s 亮 1 s;任务 3:蓝灯每隔 2 s 亮 1 s;任务 4:接收到完整数据帧时,将全部内容发送至 PC 机. 串口 1 初始化波特率为 9 600,1 位停止位,无校验,如图 3 所示.

测试结果表明,MQX-Lite 操作系统工作正常,任务函数的执行和切换稳定可靠.

下面讨论 MQX-Lite 与标准 MQX 不同之处:

1)MQX-Lite 中没有_time_delay( )函数,用_time_delay_ticks( )函数代替.

图3　工程测试结果

2)MQX-Lite 无 I/O 子系统,无 printf( )等输出函数,需用独立开发的串口驱动来实现发送任务数据输出.

3)由于 MQX-Lite 的任务资源均需在编译时预先分配好,所以任务公共头文件 app_inc. h 中还需确定任务栈大小并静态地为它们分配空间. 此外,在任务模板 task_templates. c 文件中还需为自启动任务注册任务栈,给 MQX-Lite 启动过程中创建自启动任务时使用.

4)MQX-Lite 裁减掉了 I/O 子系统,自启动任务需利用独立的外设驱动程序,对使用到的 I/O 外设进行初始化,并使能总中断和模块中断. 此外,由于 MQX-Lite 不支持动态创建任务,所以不能使用_task_create( )函数来创建任务,而是使用_task_create_at( )函数,利用已静态分配好的任务栈空间来创建任务.

5　结语

标准 MQX 的写入文件 AMQXFW. elf 大小为 29 190 KB,MQX-Lite 的写入文件 kl25_mqx. elf 大小为 5 979 KB,从中可以看出,MQX-Lite 比标准 MQX 减小了 5 倍.

本文的创新点是创建嵌入式实时操作系统 MQX 的轻量级工程框架,该框架进行适当修改就可从现处的 KDS 开发平台移植到 IAR 和 KEIL 平台,同时将恩智浦半导体公司的基于 ARM Cortex-M0 + 架构的 Kinetis L 系列 MKL25Z128VLK4 芯片移植到该框架下. 同时介绍了板级支持包 BSP 的开发过程,并对开发过程中常遇到的问题,如延时函数、输出函数、任务模板、自启动任务以及任务栈等的使用和设置进行了说明,为系统进一步扩展和应用打下基础,同时对深入了解 MQX 的运行、内核的裁剪和移植、建立应用环境等有一定的启迪意义.

[参考文献]

[1]FREESCALE. MQX support user guide[EB/OL]. [2017 - 02 - 26]. http://www. frescale. com.

[2]蒋建武,王宜怀. MQX 操作系统启动流程剖析[J]. 福州大学学报(自然科学版),2016,44(2):64 - 68.

[3]文瑾. 基于 ARM Cortex M0 + 内核的 MQX-Lite RTOS 启动流程剖析[J]. 昆明学院学报,2016,38(6):48 - 51.

[4]任慰,何顶新,赵金. 一种新型嵌入式实时操作系统的移植研究[J]. 计算机工程,2013,39(5):284 - 287.

[5]王宜怀,吴瑾,蒋银珍. 嵌入式系统原理与实践-ARM Cortex-M4 Kinetis 微控制器[M]. 北京:电子工业出版社,2012.

[6]LIMITED A R M. ARM v7-M architecture reference manual [EB/OL]. [2017 - 03 - 05]. http://www. xueshu. baidu. com.

[7]朱仕浪,王宜怀,冯德旺. MQX 实时操作系统构件化工程框架研究[J]. 武汉理工大学学报,2013,35(10):135 - 140.

[8]吴瑾,程玉娟,王宜怀. MQX 实时操作系统在 MCF52259 上移植的研究与实现[J]. 微计算机信息,2011,27(3):95 - 97.

[9]周洁,郭重汝. 嵌入式系统板级支持包的研究与构建[J]. 计算机工程与设计,2012,33(10):3812 - 3816.

[10]屈新怀,张先燏,丁必荣. 可控嵌入式构件框架的研究与实现[J]. 计算机应用与软件,2014(2):14 - 16.

[11]FREESCALE. KL25 sub-family reference manual[EB/OL]. [2017 - 03 - 15]. http://www. frescale. com.