

三期替身机器人底层控制逻辑

Robot Main Body (ARM-Linux & MCU) Control Logic

版本号：Beta-0.1

制定人: 贾林

2014年12月30日

1. 概要

底层通讯协议更换为严格标准的CANOpen通讯协议后，根据协议定义的各子协议特性及机器人端的控制逻辑，将机器人的控制层级分为更为明晰的两层Master-Slave形式。

本文档将详细阐述具体的控制逻辑，主要包括Master，即机器人端主控（ARM-Linux），与上层控制终端的交互及其和底层各控制模块之间的控制逻辑。

具体目标

制定清晰明了的机器人端的控制逻辑，完成对机器人各模块的合理调度和控制。

解决方案

依托以下协议规范及说明文档：

*/* *[PROTOCOL] CANOpen Protocol*

*/* *[PROTOCOL] Object Dictionary – BMS*

*/* *[PROTOCOL] Object Dictionary – CHASSIS*

*/* *[PROTOCOL] Object Dictionary – ARM*

*/* *[PROTOCOL] Object Dictionary – NECK*

*/* *[PROTOCOL] Object Dictionary – Lifter*

修订规则

对本文档的修订，请利用Word本身自带的修订及批注功能：

* 点击“审阅”-“修订”按钮后，word修订模式打开。之后就可以在文档中进行修改了，所有修改的内容都会有记录。
* 点击“审阅”-“新建批注”按钮后，可对选中文字内容进行批注说明。

请利用修订规则以更好的维护说明文档的版本。

2. 机器人本体硬件结构及需求

机器人本体机械结构上自上而下分为如下几个部分：

头部：包括显示屏、摄像头 --- 直连ARM-Linux板(Linux PCB)

颈部：包括2自由度颈部活动关节 --- 颈部控制板(Neck PCB)

上身：包括两条五自由度机械臂及手掌 --- 手臂控制板(ARM PCB)

升降机构：升降杆，用来做机器人高度控制 --- 升降台控制板(Lifter PCB)

底盘：机器人的行走机构 --- 底盘控制板(Chassis PCB)

此外，机器人的电力供应还需BMS的控制，即 --- BMS控制板(BMS PCB)

机器人本体的控制逻辑即ARM-Linux通过与各控制板、上层控制端(PC or Web)的通讯，根据某些条件的逻辑关系对机器人各部分执行终端进行控制和调度的规则。

本章主要描述机器人各部分的结构及控制特性。

2.1 ARM-Linux

ARM-Linux是机器人端的控制和通讯中心。除了接受上层控制指令外，还需获取并实时监测各机器人各部分的当前状态，并据此自发完成某些控制任务。

为了实现此功能，在硬件上，ARM-Linux至少需搭载无线网卡(WIFI Block)、CAN收发设备、显示屏(Display Screen)和音频系统(Sound)。

除此之外，ARM-Linux的正常工作还需Linphone软件的支持，用来发送SIP消息和上层控制系统通讯。

所以，ARM-Linux需要具备以下功能：

* 消息转发功能：这里指双向的消息转发，即“上位机->ARM-Linux->执行机构”和“执行机构->ARM-Linux->上位机”。
* 机器人主控功能：根据机器人各部分状态，合理规划控制任务优先级和组合。
* 信息记录功能：记录本身和下层机构的重要信息，如报错及关键参数的备份。

2.2 颈部 (NECK PCB)

颈部链接头部和机器人身体，可根据指令做2自由度运动（摇摆和俯仰），由两个舵机来控制。颈部需要具有的功能如下：

* 控制功能：根据所接受指令控制颈部运动。
* 通讯功能：接收和发送数据，包括控制指令接收、错误信息发送、上位机查询信息解析和返回。

具体的控制命令定义和颈部的可查询信息及查询方式可见文档：

*[PROTOCOL] Object Dictionary – NECK.docx*

2.3 手臂 (ARM PCB)

手臂是机器人上身的重要执行机构，每条手臂可根据指令做5自由度运动：肩部2个自由度；上臂1个自由度；肘部1各自由度；小臂1各自由度。除此之外，手臂末端还接有手掌，每个手掌包含五个手指，每根手指都可做弯曲动作。

手臂所需功能：

* 控制功能：根据所接受指令控制颈部运动。
* 通讯功能：接收和发送数据，包括控制指令接收、错误信息发送、上位机查询信息解析和返回。

具体的控制命令定义和颈部的可查询信息及查询方式可见文档：

*[PROTOCOL] Object Dictionary – ARM.docx*

2.4 升降台 (LIFTER PCB)

升降台链接底盘和机器人身体，可根据指令做单自由度运动（上下运动），由升降杆来控制。升降台需要具有的功能如下：

* 控制功能：根据所接受指令控制升降杆运动。
* 通讯功能：接收和发送数据，包括控制指令接收、错误信息发送、上位机查询信息解析和返回。

具体的控制命令定义和颈部的可查询信息及查询方式可见文档：

*[PROTOCOL] Object Dictionary – LIFTER.docx*

2.5 底盘 (CHASSIS PCB)

底盘是机器人端最重要的执行机构，可根据指令在平面内做360°移动。结构上底盘由3个电机及各自驱动器、3个拉压力传感器、6个红外测距传感器、8个碰撞传感器、2个红外接收头、1个激光扫描仪组成。

底盘的功能较为复杂，且传感器较多，所需基本功能如下：

* 控制功能：根据所接受指令控制底盘运动。
* 通讯功能：接收和发送数据，包括控制指令接收、错误信息发送、上位机查询信息解析和返回。

具体的控制命令定义和可查询信息及查询方式可见文档：

*[PROTOCOL] Object Dictionary – CHASSIS.docx*

2.6 BMS (BMS PCB)

BMS为机器人提供稳定电源并检测电源的实时状态，可以提供有关电池的所有关键信息并控制机器人各部分用电（开关）。所需基本功能如下：

* 控制功能：根据所接受指令控制控制机器人各部分电源开关。
* 通讯功能：接收和发送数据，包括控制指令接收、错误信息发送、上位机查询信息解析和返回。

具体的控制命令定义和可查询信息及查询方式可见文档：

*[PROTOCOL] Object Dictionary – BMSdocx*

3. 各部分控制流程图

机器人端的控制结构可分为两部分：

Master：ARM-Linux Slave：NECK、ARM、LIFTER、CHASSIS、BMS

这种分类方式不只是针对控制结构，同时也表征机器人端的通讯结构。特别是在应用CANOpen通讯协议后，这种Master-Slave的结构显得尤为重要，关于CANOpen的相关说明请参看文档：

*/* *[PROTOCOL] CANOpen Protocol.docx*

与此同时，由于在三期替身机器人的设计中，底层的控制平台升级为FreeRTOS+CANOpen的形式，机器人每一个子模块内部的控制流程（或者说是程序控制结构）可以参看文档：

/STM32-FreeRTOS-CANOpen Platform.docx

2.1 主控 (ARM-Linux)

主控(Master)的启动：



图2.1 ARM-Linux 启动流程

当ARM-linux文件系统启动后，开始运行控制程序，这时，需要检测系统是否有错误，若发生错误，应将其存入Log文件中并在允许的情况下将错误发回至上位机。

若无错误产生，则正常运行程序。此时，ARM-Linux应可以接收机器人各系统的心跳数据，据此可判断各子模块的运行状态，此时，各子模块处于Pre-Operational状态，无法发送和解析PDO数据，但可以接收Master主站的SDO和NMT指令。若子模块启动发生问题，ARM-Linux能够收到其发送的EMCY信息。

以BMS为例，当其上电后，会立即向CAN总线发送Bootup信息：0x002|0x00；然后BMS自动进入Pre-Operational状态，此时发送Heartbeat信息：0x702|0x7F，0x7F表示Pre-Operational状态。由于首次启动时，ARM-Linux和各子模块同时上电，ARM-Linux启动较慢，故当ARM-Linux启动成功后，可以接收到各子模块的Heartbeat信息及其当前状态。

若此时ARM-Linux和各子模块均无错误产生，ARM-Linux发送NMT指令，将各个模块的状态转为Operational状态，此时，机器人系统整体启动完毕，等待上位机控制指令。

2.2 子模块 (NECK、ARM、LIFTER、CHASSIS)

各个执行机构的启动过程均相同，如图2.2所示：



图2.2 子模块启动过程

2.3 控制流程分析

如图2.3，整体机器人端控制流程图如下：

图2.3 机器人端底层控制流程图

承接2.1所述内容，当机器人子模块及主控（ARM-Linux）均成功启动后，处于等待状态，此时，上位机的控制指令是通过SIP消息下发到ARM Linux主控板上的，Linphonec程序将收到的SIP消息通过Socket消息发送给RobotCtrl，RobotCtrl直接读写CAN总线，对机器人的控制逻辑大部分将在RobotCtrl中实现：



图2.4 ARM-Linux机器人控制框架简图

Linphonec和RobotCtrl采用Socket方式的进程间通信，RobotCtrl在SIP消息和CAN总线之间嵌入了一个逻辑层，在这个逻辑层中集中处理机器人控制逻辑。如下：



图2.4 ARM-Linux机器人控制框架

ARM-Linux的代码结构入股2.5所示：

图2.4 ARM-Linux代码结构

在这里，主站与从站通讯的主要内容有以下几点：

1. NMT网络管理：包括改变从站状态机运行状态从而控制从站启动与关闭；
2. PDO报文管理：包括发送RPDO到从站进行运动控制，接收TPDO获取从站信息；
3. SDO报文管理：通过SDO报文获取从站信息、更改从站通信或功能属性；
4. Heartbeat报文管理：接收从站的Heartbeat报文，监测从站状态；
5. SYNC报文管理：生成SYNC报文，作为从站周期性报文的时间基准。

错误种类及产生错误后ARM-Linux的行为对照表：

asdfasdf若无错误产生，则正常运行程序。此时，ARM-Linux应可以接收机器人各系统的心跳数据，据此可判断各子模块的运行状态，此时，各子模块处于Pre-Operational状态，无法发送和解析PDO数据，但可以接收Master主站的SDO和NMT指令。若子模块启动发生问题，ARM-Linux能够收到其发送的EMCY信息。

以BMS为例，当其上电后，会立即向CAN总线发送Bootup信息：0x002|0x00；然后BMS自动进入Pre-Operational状态，此时发送Heartbeat信息：0x702|0x7F，0x7F表示Pre-Operational

3. 指令格式对照

对于指令格式，由于

机器人本体的控制逻辑即ARM-Linux通过与各控制板、上层控制端(PC or Web)的通讯，根据某些条件的逻辑关系对机器人各部分执行终端进行控制和调度的规则。

本章主要描述机器人各部分的结构及控制特性。

2.1 ARM-Linux

ARM-Linux是机器人端的控制和通讯中心。除了接受上层控制指令外，还需获取并实时监测各机器人各部分的当前状态，并据此自发完成某些控制任务。

2.2 执行机构(NECK、ARM、LIFTER、CHASSIS)

注意，Byte1中的0x01H为底层定义，即为默认速度

2.3 BMS (BMS PCB)

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Contents** |
| Bit 7 | 0 | - |
| Bit 6 | Instruction Error | When undefined Instruction is transmitted or the Action command is delivered without the reg\_write command |
| Bit 5 | Overload Error | When the current load cannot be controlled with the set maximum torque |
| Bit 4 | CheckSum Error | When the Checksum of the transmitted Instruction Packet is invalid |
| Bit 3 | Range Error | When the command is given beyond the range of usage |
| Bit 2 | OverHeating Error | When the internal temperature is out of the range of operating temperature set in the Control Table |
| Bit 1 | Angle Limit Error | When Goal Position is written with the value that is not between CW Angle Limit and CCW Angle Limit |
| Bit 0 | Input Voltage Error | When the applied voltage is out of the range of operating voltage set in the Control Table |