OpenTSN网络控制器使用手册

（版本1.1）

OpenTSN开源项目组

2020年04月

**当前版本**

|  |  |
| --- | --- |
| 文件标识 | OpenTSN3.0工程使用 |
| 当前版本 | 1.1 |
| 完成日期 | 2021.04.02 |

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修订时间** | **修订人** | **修订内容** |
| 1.0 | 2021.01.28 | 李军帅 | 完成TSN集中控制器使用手册的初步版本 |
| 1.1 | 2021.04.02 | 李军帅 | 修改文件格式，按照统计的文件标准重新修改文本 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[OpenTSN网络控制器使用手册 1](#_Toc68271253)

[**一、引言** 4](#_Toc68271254)

[**二、运行环境** 4](#_Toc68271255)

[**三、文件说明** 4](#_Toc68271256)

[3.1 ARP应用 5](#_Toc68271257)

[3.2 通用函数库 5](#_Toc68271258)

[3.3状态监测应用 7](#_Toc68271259)

[3.4 PTP时间同步应用 8](#_Toc68271260)

[**四、编译和运行步骤** 9](#_Toc68271261)

[4.1 编译 9](#_Toc68271262)

[4.2运行 10](#_Toc68271263)

[**附录一：Libxml2库安装教程** 11](#_Toc68271264)

[**附录二：问题记录** 12](#_Toc68271265)

**一、引言**

本文档为TSN集中控制器使用手册，主要描述控制器运行环境、文件说明、编译和运行步骤、组网示例，用户可以参考该文档使用网络控制器。

网络控制器主要功能包含ARP代理、网络初始配置、状态监测和PTP时间同步，所有功能需要与硬件配合才能实现。

**二、运行环境**

网络控制器的运行环境为Linux系统，需要安装以下库支撑程序运行。

* 需要安装libpcap和libnet库，用于收包和发包.。
* 需要安装libxml2库，用于进行xml文件解析。
* Linux设备需要把网络接口开启混杂模式

**三、文件说明**

网络控制器包含的源文件如下图所示，arp文件夹下存放的是arp应用程序，cnc\_api文件下存放通用基础库，cnc\_ptp文件下存放PTP时间同步应用程序，net\_init存放网络初始化应用程序，state\_monitor文件夹存放状态监测应用程序。



图3-1 网络控制器源文件目录

## 3.1 ARP应用

arp文件夹存放ARP代理应用程序，用于响应ARP请求报文，包含的文件如下图所示。

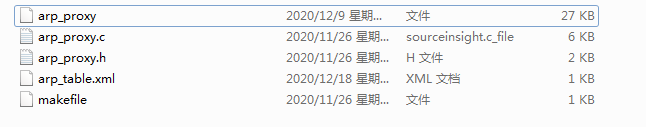


图3-2 ARP应用源文件目录

各个文件的具体含义：

* arp\_proxy：进行编译后生成的可执行文件
* arp\_proxy.c：arp代理的源文件，所有的函数在该文件中实现
* arp\_proxy.h：arp代理的头文件，arp数据结构定义在该文件里面。
* arp\_table.xml：用于存放arp表项
* makefile：可以进行编译的文件

## 3.2 通用函数库

cnc\_api文件夹为基础函数库，包含include文件夹（主要存放通用函数的api头文件）和src文件夹，包含的文件如下图所示。

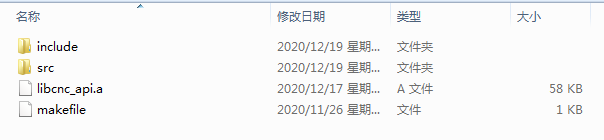


图3-3 通用函数库源文件目录

各个文件的具体含义：

* include：主要存放通用函数的API头文件和通用的数据结构定义
* src：通用API的实现
* libcnc\_api.a：编译基础函数生成的动态库文件
* makefile：可以进行编译的文件

include文件夹中包含的文件：



图3-4 include源文件目录

* beacon\_report.h：芯片上报报文的数据结构定义
* cnc\_api.h：通用API的头文件
* reg\_cfg.h：芯片配置和HCP配置报文的数据结构定义
* tsmp\_protocol.h：tsmp协议的数据结构定义

src文件夹中包含的文件：

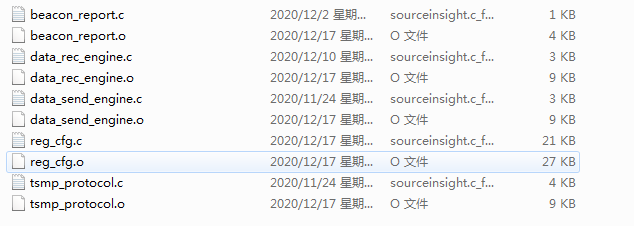


图3-5 cnc\_api中src源文件目录

* beacon\_report.c：解析芯片上报报文和HCP上报报文的通用函数实现
* beacon\_report.o：编译beacon\_report.c生成的目标文件
* data\_rec\_engine.c：数据接收通用函数实现
* data\_rec\_engine.o：编译数据接收文件生成的目标文件
* data\_send\_engine.c数据发送通用函数实现
* data\_send\_engine.o：编译数据发送文件生成的目标文件
* reg\_cfg.c：寄存器配置通用函数实现
* reg\_cfg.o：编译寄存器配置文件生成的目标文件
* tsmp\_protocol.c：tsmp协议通用函数实现
* tsmp\_protocol.o: 编译tsmp协议文件生成的目标文件

## 3.3状态监测应用

state\_monitor文件夹存放状态监测应用程序的源文件，用于对网络进行状态检测，具体报文源文件如下图所示：



图3-6 状态监测源文件目录

各个文件的具体含义：

* makefile：make该文件生成可执行文件monitor
* monitor：生成的可执行文件
* state\_monitor.c：状态监测模块的主函数和关键函数的实现
* state\_monitor.h：状态监测的头文件
* topolopy\_info\_xml.xml：拓扑信息的xml文本

## 3.4 PTP时间同步应用

cnc\_ptp文件夹存放时间同步PTP应用程序的源文件，用于实现网络时间同步的功能，具体报文源文件如下图所示：

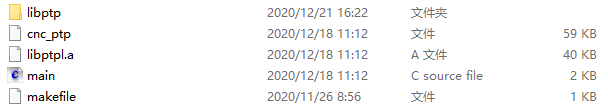


图3-7 时间同步应用PTP源文件目录

各个文件的具体含义：

* libptp：主要存放时间同步相关函数的API头文件和数据结构定义以及时间同步相关函数API的实现
* cnc\_ptp：生成的可执行文件
* Libptpl.a：编译PTP时间同步函数生成的动态库文件
* main：时间同步主函数
* makefile：make该文件生成可执行文件cnc\_ptp

libptp文件夹中包含的文件：

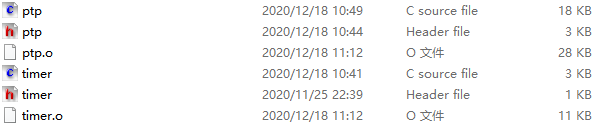


图3-8 cnc\_ptp中libptp源文件目录

* ptp.c：PTP同步处理模块的关键函数实现
* ptp.h：PTP同步处理的头文件
* ptp.o：编译时间同步处理文件生成的目标文件
* timer.c：定时器模块的关键函数实现
* timer.h：定时器处理的头文件
* timer.o：编译定时器处理文件生成的目标文件

**四、编译和运行步骤**

本章主要讲述程序运行之前对程序的编译，和程序运行的主要步骤。

## 4.1 编译

在编译时按以下步骤进行操作：

（1）首先编译cnc\_api文件夹下的库文件，在cnc\_api文件夹下执行以下命令

* make clean
* make

（2）编译net\_init文件夹下的网络初始化进程

* make clean
* make

（3）编译state\_monitor文件夹下的状态监测进程

* make clean
* make

（4）编译arp文件夹下的arp代理进程

* make clean
* make

（5）编译cnc\_ptp文件夹下的ptp时间同步代理进程

* make clean
* make

## 4.2运行

在运行时按以下步骤进行操作（注：在root权限下执行）：

1. 运行网络初始化进程

* 在net\_init目录下执行：./init 网络接口名

本设备的网络接口名为enp0s17，因此在执行时输入为 ./init enp0s17



图4-1 网络接口名查看结果

1. 运行状态监测进程

* 在state\_monitor目录下执行：./monitor 网络接口名

1. 运行arp代理进程

* 在arp目录下执行：./arp\_proxy 网络接口名

1. 运行PTP时间同步进程

* 在cnc\_ptp目录下执行：./cnc\_ptp

**附录一：Libxml2库安装教程**

* 拷贝Libxml2库文件夹到Linux机器中
* 进入libxml2-2.6.2文件夹目录下，在终端中打开
* 拷贝库文件夹到Linux机器中
* 进入libxml2-2.6.2文件夹目录下，在终端中打开
* 使用root权限执行以下命令
* make distclean
* ./configure
* make
* make install

**附录二：问题记录**

* 执行./init enp0s17时，出现没有可执行权限，需要执行chmod 777 init，赋予init文件可执行权限