

常用缩写

Product相关		
CGW	Connect GateWay	网关
CDC	Cockpit domain controller	座舱域控制器
ADAS	Advanced driving assistant system	高级辅助驾驶系统
TBOX	Telematics box	网络盒子
IVI	In-vehicle infotainment	车载娱乐
OIB	Oriented intelligent brain	中央智慧大脑
VIU	Vehicle Integrated unit	车辆集成单元
VCU	Vehicle control unit	车辆控制单元
VCC	Vehicle control center	车辆控制中心
CCU	Center Control Unit	中央控制单元
ZCU	Zone Control Unit	区域控制单元
OTA相关		
UCM	Update config management	更新配置管理
FMP	Fota Management Platform	升级管理平台
RVDC	Remote Vehicle Diagnostic Control	远程车辆诊断控制
VSM	Vehicle State Management	车辆状态管理
HMI	Human Machine Interface	人机交互界面
诊断相关		
DOIP	Diagnostic Over Internet Protocol	诊断
DTC	Diagnostic Trouble Code	诊断故障码 https://blog.csdn.net/wto9109/article/details/113575426

UDS	Unified Diagnostic Service	统一诊断服务协议 https://blog.csdn.net/wto9109/article/details/116176643
OBD	On-board Diagnostic	车载诊断仪
TSP	telematics ServiceProvider	汽车远程服务提供商
工厂相关		
FCT		
ICT		
OOB	Out of box	品质依赖性测试
PCBA		
EOL	End of line	产线最后一个步骤
项目相关		
SOR	Specification Of Requirements	指顾客方针对供应商发出的产品规格要求，一般是一个项目启动后，在供应商招标时发给供应商，其中包含产品的价格、质量、数量等各方面比较详细的要求。现在汽车行业中应用较为广泛。
TR	technical review	技术评审
	kickoff	项目开始
CR	Change of request	项目变更
SOP	Start of production	开始量产
功能安全		

FTTI	<p>fault tolerant time interval :</p> <p>minimum time-span from the occurrence of a fault in an item to a possible occurrence of a hazardous event, if the safety mechanisms are not activated</p>	<p>故障容忍时间间隔：</p> <p>如果安全机制未被激活，则从相关项中故障的发生到危险事件的可能发生的最短时间跨度</p>
ASIL	Automotive Safety Integrity Level	汽车安全完整性等级
BIST	Build-in self test	内建自检
LBIST	Logic BIST	逻辑内建自检
MBIST	Memory BIST	内存内建自检
eMCEM	Extended Microcontroller error Manager	扩展的微处理器错误管理
FCCU	Fault Collection and Control Unit	错误收集控制单元
SCST	software core self-test	软件核心自检
NCF	Non-critical Fault	非致命性错误
信息安全		

MAC	mandatory access control	必须访问控制
DAC	Discretionary Access Control	自主访问控制
TE	Type Enforcement	
RBAC	Role Based Access Control	基于角色的访问
MLS	Multi-Level Security	多级别安全
MCS	Multi-Category Security	多目录安全
OM	object managers	
AVC	access vector cache	
LSM	Linux Security Module	
网络相关		
RMNET	Remote NETwork	远端网络
ECM	Ethernet Control Mode	网络控制模式
RNDIS	Remote Network Device Interface Specification	远端网络设备接口规范
QXDM	Qualcomm eXtensible Diagnostic Monitor	高通拓展诊断监控
ECU相关缩写		
PDCM(纯电)	Powertrain domain control Module	动力域控制模块
PDCM(混动)	Powertrain domain control Module	动力域控制模块
BMS(混动)	Battery Management System	电池管理系统
BMS(纯电)	Battery Management System	电池管理系统
MCUF(纯电)	Motor control unit Front1	前电机控制器
MCUF(混动)	Motor control unit Front1	前电机控制器

MCUR(纯电)	Motor control unit Rear1	后电机控制器1 (EV 160KW)
MCUR(混动)	Motor control unit Rear1	后电机控制器1 (REV 160KW)
OBC(6.6KW)	On-board charger1	主充电机 (PHEV)
OBC(10KW)	On-board charger1	主充电机 (EV)
EPB_L	Electrical Park Brake	电子驻车制动系统
EPB_R	Electrical Park Brake	电子驻车制动系统
EMS	Engine Management System	发动机管理系统
GCU	Generate Motor Control	发电机控制器
TCU	Transmission Control Unit	变速器控制器
AC	AIR Conditioner Controller1	空调控制器
ACP	Air Conditioner Panel	空调控制面板
ADCU	Adas Domain Control Unit	自动驾驶域控制器
MPC	Multi Purpose Camera	多功能摄像头
FRM	Front Radar Module	中距离毫米波雷达
RCR-L	Rear Corner Radar-L	后角雷达-左
APA	Auto park assist	自动泊车辅助
PDC	Parking Distance Control	倒车雷达控制器
AVM	Around View Monitor	全景影像控制器
EPS	Electric Power Steering1	电动助力转向系统
GSM	Gear Shift Assembly	换挡机构总成
IPB	Integrated power brake	集成制动控制
IPB	Integrated power brake	集成制动控制
ASC	Intergated Suspension Controller	集成悬架控制器
EDC	Electric Continuous Damping	电控减振器控制器
ACU	Airbag Control Unit	安全气囊控制器
POT	Power Operation Tailgate	电动尾门
PSD_L	Power Slide Door-L	电动滑门-左

PSD_R	Power Slide Door-R	电动滑门-右
NVS	Night Vision System	夜视系统
VSP	Voice Signal Processor	低速行人报警器
AMP	Amplifier1	外置功放1
WCM	Wireless Power Charger Module	无线充电控制器
SCU_F	Driver Seat Control Unit1	驾驶员座椅控制器1
SCU-RL	Driver Seat Control Unit2	驾驶员座椅控制器2
SCU-RR	Driver Seat Control Unit3	驾驶员座椅控制器3
ICM_SOC	Integrated Camera Module	集成式摄像头模块
ICM_MCU	Integrated Camera Module	集成式摄像头模块
BLE Master	bluetooth	蓝牙控制器
BCM	Body Control Module	车身控制模块
DCU-FL	Door Control Unit-FL	左前门模块
DCU-FR	Door Control Unit-FR	右前门模块
DCU-RL	Door Control Unit-RL	左后门模块
DCU-RR	Door Control Unit-RR	右后门模块
IVI_MPU	In-Vehicle Infotainment	车载信息娱乐系统
IVI_MCU	In-Vehicle Infotainment	车载信息娱乐系统
T-BOX	Telematics Box	车联网终端
HCM-L1	Headlamp Control Module-Left1	左侧大灯模块低配
HCM-L2	Headlamp Control Module-Left2	左侧大灯模块高配
HCM-R1	Headlamp Control Module-Right1	右侧大灯模块低配
HCM-R2	Headlamp Control Module-Right2	右侧大灯模块高配
RLM-L	Rear Left Lamp Module	后部左侧灯光模块
RLM-M1	Rear Middle Lamp Module1	后部中间侧灯光模块1
RLM-M2	Rear Middle Lamp Module2	后部中间侧灯光模块2
RLM-R	Rear Right Lamp Module	后部右侧灯光模块

GW	Gateway	网关控制器
ETC	Electronic Toll Collection	电子不停车收费系统
SWM	Steering wheel module	方向盘模块
EVCC	Electric vehicle communication controller	电动汽车通信控制器

开发环境搭建

a)ubuntu 20.04安装

b)企业微信安装

```
git clone https://gitee.com/azk1992/workchat.git
```

```
cd workchat
```

```
bash install.sh
```

更新版本

```
sudo dpkg -i deepin.com.weixin.work_2.8.10.2010deepin0_i386.deb
```

```
sudo dpkg -i deepin.com.wechat_2.6.8.65deepin0_i386.deb
```

c)安装中文拼音输入法

```
sudo apt-get install ibus ibus-clutter ibus-gtk ibus-gtk3
```

d)安装todesk

e)vim c开发环境搭建

<https://www.jianshu.com/p/02809c8c97eb>

1、将vimrc放到/etc/vim/目录下

2、将vim.tar.gz 解压拷贝到~/vim/目录下

f)安装nerd插件显示目录树

<https://catonmat.net/vim-plugins-nerdtree->

g)VPN安装和连接

地址:vpn.i-tetris.com:4443

账号和密码:域账号密码

h)samb服务器

smb://smb.i-tetris.com/sambashare/

账号和密码: share

i)git加速和gerrit用户名密码配置文件

vim ~/.gitconfig

Vim ~/.netrc

岚图H53测试台架上位机:

ssh autotest@10.25.12.69

岚图H53测试台架武汉上位机

ssh autotestwh@10.58.7.206

@A123456

公司台式机:

ssh bingbingcheng@10.25.8.19

Cbb123456!

岚图H53测试 电脑:

ToDesk设备代码:674 902 162

王久远开发板

ssh -p10022 qnxuser@10.25.11.47

scp -P 10022 ./src/ta/aarch64/le/kbox.ko qnxuser@10.25.11.47:/data

LD_LIBRARY_PATH=/data/jywang/fmp_test/ /data/jywang/fmp_test/fmp_test

linux指令

安装包管理	
sudo dpkg -i example.deb	安装.deb类型的安装包
sudo apt-get install vim	安装vim软件
sudo apt-get remove --purge vim	卸载 vim软件, 完全卸载
sudo update-alternatives --config java	切换java版本

https://www.cnblogs.com/liuzhenbo/p/13176869.html	切换python版本
打包压缩管理	
tar -xvf file.tar tar -c tar -f tar -z tar -j tar -v tar -x	解包file文件 创建一个包 包的名指定 使用gzip工具压缩 使用bzip工具压缩 显示详细信息 解包
unzip file.zip -d directory	解压file文件到directory目录
bzip2 -d file.bz2	解压file文件
gzip -d file.gz	解压file文件
bzip2 file	压缩file文件
gzip file	压缩file文件
文件相关	
chmod a+x file	修改file文件的所有用户的可执行权限
chown -R 属主 file	修改file文件的属主
chgrp -R 属组 file	修改file文件的属组
du -sh file/directory	查看file文件大小或者directory目录大小
df -hl directory	查看目录剩余空间大小
find . -name "*.o" xargs rm -f	递归删除当前目录下以及子目录下的.o文件
hexdump filename -s 0x0001 -n 100	filename文件从第一个字节开始显示100个字节
https://www.cnblogs.com/woods1815/p/16220074.html ls -l grep "^-" wc -l	统计文件夹下文件数量
grep -rns "string"	全局查找string字符串
系统相关指令	

uname -r	查看内核版本
uname -a	查看内核的详细信息
uname -srm	查询内核版本信息
free -h	查询内存信息
passwd	修改密码
sudo passwd	修改root密码
sudo vim /etc/hostname	修改计算机名称
export -p	查询环境变量
users	查询当前用户
cat /proc/cpuinfo	查询cpu版本信息
cat /proc/version	查询内核版本信息
cat /proc/sys/kernel/printk	查询打印级别
cat /proc/meminfo	查询内存信息
lspci grep -i vga	查看显卡设备
df -h	查看存储
fdisk -l	查询挂载的硬件
lsusb	查询usb设备信息
ulimit -a	查看所有限制信息
readelf -d test_file	查看当前文件依赖的库
readelf --syms libkbox.so	查看当前文件的符号表
aarch64-unknown-nto-qnx7.0.0-strip --strip-unneeded libkbox.so	去掉符号表
nm -A libvoytee.so	nm查看符号表 https://www.cnblogs.com/itech/archive/2012/09/16/2687423.html
watch	周期性执行命令

Crtl + r	reverse-i-search(查找历史输入指令)
ab -n 1000 -c 100 https://www.baidu.com	向web服务器发送指定并发指定次数的请求测试
lsnice/renice	设置进程优先级
进程相关	
ps tree	查看进程树
ps auxf	查看进程, 并显示进程间的关系
ps -le	查看进程优先级
ps -Z	查看进程的安全上下文
top -d 1	查看进程, 每秒刷新一次<>上下翻页
uptime	查看平均负载, 1分钟, 5分钟和15分钟
w	查看系统负载
pgrep sshd	查看sshd进程id
软连接	
ln -s src_dir dest_dir	创建一个dest_dir快照链接文件
网络相关指令	
ifconfig	查询网卡信息
ifconfig eth0 172.28.4.1	设置eth0的ip地址为172.28.4.1
ping www.baidu.com	ping网络
telnet 10.92.0.201 9090	查看端口连接
netstat -nupl	查看UDP类型的端口

netstat -ntpl	查看TCP类型的端口 a 表示所有 n表示不查询dns t表示tcp协议 u表示udp协议 p表示查询占用的程序 l表示查询正在监听的程序
tcpdump -w net.log -i rmnet_data0	抓取网络日志
Wireshark net.log	wireshark打开net.log日志
iperf	测试网络性能
tftp服务(基于UDP)	
dpkg -l tftpd-hpa	查看tftp服务器版本号
netstat -a grep tftp	查看tftp是否启动 显示结果为 udp 0 0 *:tftp *:.* udp 0 0 *:tftp *:.* //说明已经启动 如果没有启动将没有任何输出
sudo service tftpd-hpa status	查看tftphpa状态
sudo service tftpd-hpa restart	重启tftp服务
https://drive.google.com/file/d/1UTF2Z-h2VpqNIGA8x2zYd7JyQAtnvwz3/view?usp=sharing	安装配置tftp服务
sshd服务(基于TCP)	
ssh bingbingcheng@10.25.11.140	链接10.25.11.140的bingbingcheng用户
scp -r bingbingcheng@10.25.11.140:/home/bingbin gcheng/code/monza-s32g/src/qnx/src/hardwa re/devcr .	拷贝 bingbingcheng@10.25.11.140:/home/bingbin gcheng/code/monza-s32g/src/qnx/src/hardwa re/devcr到当前目录
scp -r bingbingcheng@10.25.11.140:/home/bingbin gcheng/code/s32g_lmc/src/factory/factory_se rvice/kbox .	拷贝 bingbingcheng@10.25.11.140:/home/bingbin gcheng/code/s32g_lmc/src/factory/factory_se rvice/kbox到当前目录

adb服务(基于TCP)	
Adb devices	枚举adb设备
Adb shell	adb登陆
Adb root	Adb root用户
Adb remount	Adb 挂载
openssl指令	
openssl genrsa -out private_pkcs1.pem 2048	生成2048密钥
openssl rsa -in private_pkcs1.pem -out public_pkcs1.pem -pubout -RSAPublicKey_out	提取2048公钥
openssl rsa -in private_pkcs1.pem -text -noout	查看2048密钥
openssl genpkey -algorithm ED25519 -out ed25519.key	生成ed25519密钥
openssl pkey -in ed25519.key -pubout -out ed25519_pub.key	提取ed25519公钥
openssl pkey -in ed25519.key -text -noout	查看ed25519密钥
base64 -d download1.crt > download1.der	Base64 解码
openssl x509 -in download1.der -inform der -noout -text	DER格式的证书查看
openssl x509 -in download1.pem -noout -text	PEM格式的证书查看
openssl x509 -in b2.crt -noout -enddate	证书截止日期查看
Openssl speed aes	查看openssl aes算法性能

https://blog.csdn.net/dobell/article/details/54985245	
openssl speed sha256	查看openssl sha256算法性能
openssl verify -CAfile ca/ca.crt server/server.crt	校验服务器证书
openssl verify -CAfile ca/ca.crt client/client.crt	校验客户端证书
openssl dgst -sha256 filename	计算filename的sha256
https://www.jianshu.com/p/c93a993f8997	rsa密钥解析
http://bunchpost.site/%E5%AE%89%E5%85%A8%E5%8A%A0%E5%AF%86/2017/07/04/ed25519-introduction.html	Curve25519/Ed25519/X25519算法简介
https://www.ssl.com/guide/pem-der-crt-and-cer-x-509-encodings-and-conversions/	PEM和DER区别
https://zh.wikipedia.org/wiki/%E5%82%B3%E8%BC%B8%E5%B1%A4%E5%AE%89%E5%85%A8%E6%80%A7%E5%8D%94%E5%AE%9A	TLS和SSL以及HTTPS关系
http://www.openssl.org/docs/man1.0.2/man3/engine.html	openssl使用硬件加速
curl指令	
curl -v --cacert ca.pem --cert b2.crt --key b2.key https://iotdev.dfmc.com.cn:485/vcpapi	岚图H53测试证书服务器dcvp地址
curl -v --cacert ca.pem --cert b2.crt --key b2.key https://vdcvpapikpi.dfmc.com.cn/api	岚图H53生产证书服务器dcvp地址
	岚图H97测试证书服务器dcvp地址,pc测试

curl -v --cacert ./ca.pem --cert ./b2.crt --key ./b2.key https://dcvpqwpkizh.dfmc.com.cn:443/api	(10.96.0.201)
curl -v --cacert ./ca.pem --cert ./b2.crt --key ./b2.key https://dcvppki.dfmc.com.cn/api	岚图H97预生产证书服务器dcvp地址,pc测试
curl -v --cacert ./ca.pem --cert ./b2.crt --key ./b2.key https://pkigw.dcvp.dfmc.com.cn:443/api	岚图H97生产证书服务器dcvp地址,,pc测试 (10.92.1.201)
curl -uforbidden.city:AP8ApTbqxN3a8gTRVUjjZ5a feBq -T s32g3_m7_pack_mboot_img_20221026.bin http://artifactory.i-tetris.com/artifactory/kiara-s oc/monza/s32g/kiara_fw_master/s32g3_m7_ pack_mboot_img_20221026.bin	Artifactory 上传文件
iptables	
iptables -F	清除iptables配置
audit安全审计	
service auditd start	开启安全审计功能, https://www.cnblogs.com/Alanf/p/8653395.html
service auditd status	查看安全审计状态

selinux	
audit2allow	Generates policy allow rules from the audit.log file
audit2why	Describes audit.log messages and why access was denied.
avcstat	Displays the AVC statistics.
chcat	Change or remove a category from a file or user.
chcon	Changes the security context of a file.
checkmodule	Compiles base and loadable modules from source
checkpolicy	Compiles a monolithic policy from source.
fixfiles	Update / correct the security context of for filesystems that use extended attributes.
genhomedircon	Generates file configuration entries for users home directories. This command has also been built into semanage(8), therefore when using the policy store / loadable modules this does not need to be used.
getenforce	获取selinux开启状态
getsebool	Shows the state of the booleans.
load_policy	Loads a new policy into the kernel. Not required when using semanage(8) / semodule(8) commands.
matchpathcon	Show a files path and security context.
newrole	Allows users to change roles - runs a new shell with the new security context.

restorecon	Sets the security context on one or more files.
run_init	Runs an init script under the correct context.
runcon	Runs a command with the specified context.
selinuxenabled	Shows whether SELinux is enabled or not.
semanage	Used to configure various areas of a policy within a policy store.
semodule	Used to manage the installation, upgrading etc. of policy modules.
semodule_expan	Manually expand a base policy package into a kernel binary policy file.
semodule_link	Manually link a set of module packages.
semodule_package	Create a module package with various configuration files (file context etc.)
sestatus	获取selinux状态、policy状态
setenforce	设置selinux状态
setfiles	Initialise the extended attributes of filesystems.
setsebool	Sets the state of a boolean to on or off persistently across reboots or for this session only.
setenv mmcargs setenv bootargs console=ttyLF0,115200 root=/dev/mmcblk0p5 rootwait rw earlycon nohz=off coherent_pool=64M security=selinux selinux=0	uboot中关闭selinux
内核符号表	
https://developer.aliyun.com/article/53679 <pre>cat /proc/kallsyms</pre>	linux内核符号表kallsyms简介

<pre>cat /proc/sys/kernel/kptr_restrict</pre> 输出是2. https://www.cnblogs.com/hellokitty2/p/16468019.html	user版本上查看内核符号表, 地址全是0;
模块相关指令	
insmod demo.ko	加载驱动模块
rmmod demo.ko	卸载驱动模块
lsmod	查看已经加载的模块
modinfo demo.ko	查看模块信息

qnx指令

进程相关	
s32-hse -v -i200 &	启动hse的进程
pidin grep kbox	查看kbox的进程
slay 65560	杀死65560进程
hogs grep kbox&	查看kbox的内存占用情况
top	查看系统的cpu、内存占用情况
日志相关	
slog2info grep hse	查看hse打印的信息
slog2info -c	清除日志缓存

slog2info -b <process_name> -w &	查看进程的实时日志
dlt_convert -a ACORESMIP_1722_19700101-083839.dlt grep -i kbox	Dlt 查看文件中的log, 文件路径/var/log/dltlogs
dlt_receive -a localhost grep KBox-TA	Dlt通过socket接收log
tcpdump -w log.cap tcpdump -i vlan7 -w /data/vlan7.pcap &	网络抓包 网络抓包VLAN7的数据
wireshark log.cap	查看网络日志
系统相关	
shutdown	重启
cp libkbox.so /lib/libkbox.so	动态库要拷贝到lib目录下, app默认到lib目录下找库
date MMDDhhmm[[CC]YY][.SS]	设置板子时间
vi /etc/startup.sh	修改默认路由
vi /etc/resolv.conf	修改默认DNS
route show	查看路由信息
cat /etc/image_version	查看镜像版本
mount -o remount,rw /dev/emmc0.qnx6.4 /	文件系统由只读改为非只读
mkqnx6fs /dev/emmc0.qnx6.8	文件系统/data目录由10G变成26G
self-upgrade upgrade 7 mega-image-silverstonemonza-2022092	更新A核镜像
bsp-upgrade upgrade 3 1 1 mega-image-silverstonemonza-20220923070 813.rootfs.sdcard	更新A核uboot
LD_LIBRARY_PATH=/data ./voy_ut	运行时连接库路径

netstat -an	查看所有网络端口
获取物理寄存器	
xrdc_test perp read 0x40210104 4 0	读取HSE MU0 FSR物理寄存器的地址对应的值
获取版本号	
get-version readsraminfo A	获取A核实际启动分区
get-version readsraminfo M	获取M核实际启动分区
cat /etc/image_version	查看A核版本号
cat /etc/mcore_version	查看A核打包时写入的M核版本号
cat /etc/oem_version	查看A核打包时写入的岚图版本号
get-version readpartinfo A ABoot	获取A核A分区bootloader版本号
get-version readpartinfo B ABoot	获取A核B分区bootloader版本号
get-version readsram A MBoot	获取M核A分区bootloader版本号
get-version readsram B MBoot	获取M核B分区bootloader版本号
get-version readsram A MApp	获取M核A分区APP版本号
get-version readsram B MApp	获取M核B分区APP版本号
mount	A核挂载获取
sja1110 getver	switch版本号获取
sja1110 gen -r	switch查看是否开启mirror功能
/etc/upgrade/diag-switch-upgrade.sh -off	Switch 关
/etc/upgrade/diag-switch-upgrade.sh -on	Switch 开
ota环境切换	
echo test > /data/environment	

echo stg > /data/environment echo pro > /data/environment	

uboot指令

fatls mmc 0:3	查看mmc分区
run loadfdt	加载fdt
run load_qnx_image	加载qnx kernel
bootm \${fdt_addr}	从kernel启动
run	
printenv	打印环境变量
saveenv	设置环境变量
md 0x4030c090 01	读寄存器地址
mw 0x4030c090 0xAB3498FE 01	写寄存器地址
mw 0x4030c080 0x01 01	写寄存器
md 0x4030c080 01	读寄存器

vim操作

接受命令模式	
a	当前光标的下一个字符处插入
i	当前光标处插入

o	换行的第一个字符处插入
ctrl +v	可视模式操作
:	进入ex模式
h j k l	左下上右移动光标
[[移动到上一个代码块
]]	移动到下一个代码块
gg	移动到第一行
G	移动到最后一行
\$	移动到行尾
^	移动到行首
dd	剪切
yy	复制
p	粘贴
/	查找（n下翻, N上翻）
u	撤销
ctrl +R	重做
ctrl +]	跳转到定义处
ctrl + t	跳转到跳转前位置
ctrl + ww	光标从编辑窗口跳转到Tlist窗口
EX模式	
:w	保存
:q	退出
:ts main	查询main标签

:tag main	跳转到main标签
: ! ls	vim里执行指令
:Tlist	打开Tlist
:%s /123/abc/g	所有123的地方替换为abc
:vimgrep /user/g **	使用该命令可以查找当前目录下所有文件中包含 user 字符串的文件, 并跳转到第一个匹配的文件
:cw :cn :cp	打开vigrep的搜索结果
:sp file	将vim窗口横向分为2个窗口
:vsp file	将vim窗口纵向分为2个窗口
:%!xxd	16进制编辑
:%!xxd -r	退出16进制编辑
vimdiff	
] c	跳转到下一个差异点
[c	跳转到上一个差异点
d p	Diff push, 将本侧的差异覆盖另一侧对应位置
d o	Diff obtain, 从另一侧对应位置拷贝到本侧

Shell脚本

1、shell中常用指令

调试指令	
Set +e	脚本中的指令报错继续执行后续指令 https://blog.csdn.net/xiaofei125145/article/details/39345331

Set -e	脚本中的指令报错后不再继续执行后续指令
Set -v	打印脚本的内容
Set -x	打印执行过程
echo \${}	打印调试信息

2、Linux 环境变量总结

<https://www.jianshu.com/p/ac2bc0ad3d74>

Yocto

1)常见宏的意义

/\${B}

/\${S}指出source code存放的位置

/\${D}指存放编译后生成binary的地址

/\${bindir}指文件系统的 /usr/bin

install -d 指建立一个folder

install -m 指把文件收集到指定目录下, 并修改权限

2)yocto bb文件打印调试信息

1、第一是Python形式，该形式可在console上打印出来: bb.plain, bb.note, bb.warn, bb.error, bb.fatal, bb.debug

举例如下：

xxx.bb或xxx.bbapend

```
python do_listtasks() {
    bb.debug(2, "Starting to figure out the task list")
    if noteworthy_condition:
        bb.note("There are 47 tasks to run")
    bb.debug(2, "Got to point xyz")
    if warning_trigger:
        bb.warn("Detected warning_trigger, this might be a problem later.")
    if recoverable_error:
        bb.error("Hit recoverable_error, you really need to fix this!")
    if fatal_error:
        bb.fatal("fatal_error detected, unable to print the task list")
    bb.plain("The tasks present are abc")
    bb.debug(2, "Finished figuring out the tasklist")
}
```

2、第二种是bash形式，该形式会在temp目录下的log中包含，需要inherit logging(base.bbclass会包含，通常不需要特意添加): bbplain, bbnote, bbwarn, bberror, bbfatal, bbdebug

举例如下：

xxx.bb或xxx.bbapend

```
do_configure_prepend() {
    builddir=`readlink -f ${WORKDIR}/../../../../ | awk -F "/" '{ print $NF }'`
    bbplain "xxxxxxxxxxxxx..... WORKDIR = ${WORKDIR}" /*此条会打印到前台，同时会
打印到temp目录下的log文件里面*/
    bbplain "xxxxxxxxxxxxx..... bbplain = ${builddir}"
    # bb.plain "xxxxxxxxxxxxx..... bb.plain = ${builddir}"
    echo "hjkjkhkhkhkhkhkhkhkh" /*此条不会打印到前台，会打印到对应的temp目录下的log里面*/
}
```

<https://www.jianshu.com/p/4ca51e851d3a>

3)yocto 版本查看

vim Project/meta-poky/conf/distro/poky.conf

```
DISTRO = "poky"
DISTRO_NAME = "Poky (Yocto Project Reference Distro)"
DISTRO_VERSION = "3.2.4"
DISTRO_CODENAME = "gatesgarth"
SDK_VENDOR = "-pokysdk"
SDK_VERSION = "${@d.getVar('DISTRO_VERSION').replace('snapshot-${DATE}', 'snapshot')}"
MAINTAINER = "Poky <poky@lists.yoctoproject.org>"
```

makefile

1) Makefile 语法入门

https://blog.csdn.net/afei_/article/details/82696682?utm_source=app&app_version=4.12.0

2) Linux学习笔记——例说makefile 头文件查找路径

<https://blog.csdn.net/xukai871105/article/details/36476793>

3) 多个文件目录下Makefile的写法

https://blog.csdn.net/YJ1an/article/details/103677603?utm_source=app&app_version=4.12.0

4) Makefile 语法知识

一、语法规则

目标.....: 依赖.....

命令1

命令2

二、通配符

%.* #所有的.* 文件

\$@ #表示目标

\$< #表示第一个依赖文件

\$^ #表示所有的依赖文件

三、假想目标

.PHONY: clean #将clean定义为假想目标, 这样就算Makefile目录下有clean文件也不影响

四、变量

:= #即时变量赋值

= #延时变量赋值

?= #延时变量, 如果是第一次定义则赋值有效, 如果前面定义过了则忽略该句赋值

+= #附加, 即时变量还是延时变量取决前面的定义

五、函数

\$(warning string) //用来作为makefile调试

Cmake

1) Cmake和make的区别

<https://www.huaweicloud.com/articles/c43f4f15a7d47a99406f93a855ec4c7e.html>

2) Cmake语法与实战入门

<https://zhuanlan.zhihu.com/p/267803605>

3) 语法规则

a、单行注释

#

b、多行注释

[[

]]

4) 添加打印信息

```
message("hello")
```

```
message("${PROJECT_SOURCE_DIR}")
```

Linux 驱动开发

1) Linux驱动第一篇-----最简单的内核模块

<https://blog.csdn.net/u014183456/article/details/85642381>

2) Linux驱动第二篇-----内核模块解析

<https://blog.csdn.net/u014183456/article/details/85703527>

3) Linux驱动第三篇-----把驱动编译到内核中

<https://blog.csdn.net/u014183456/article/details/85705758>

4) 如何使用内核API函数proc_create

<https://www.jianshu.com/p/2ddd32527367>

5) Linux 内核中 Kconfig 文件的作用和添加 menuconfig 项的方法

<https://cloud.tencent.com/developer/article/1155242>

6) Linux内核platform_get_resource函数如何得到设备的基地址

https://blog.csdn.net/SdustLiYang/article/details/6782714?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-6.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-6.control

7) Linux 字符设备驱动开发基础(五)—— ioremap() 函数解析, 物理地址映射为内核空间的虚拟地址

https://blog.csdn.net/zqixiao_09/article/details/50859505

8) linux内存管理

<https://cloud.tencent.com/developer/news/646104>

9) linux内存管理(一): arm64内核内存布局

<https://blog.csdn.net/yhb1047818384/article/details/104621500>

10) linux设备树

11) linux中断服务如何将中断注册到内核中去

12) 打印级别管理

a) 设置打印级别

```
echo "7 4 1 7" > /proc/sys/kernel/printk
```

b) 显示当控制台打印级别

```
cat /proc/sys/kernel/printk
```

c) 不够打印级别的信息会被写到日志中可通过dmesg 命令来查看

d) printk的打印级别

```
#define KERN_EMERG      "<0>" /* system is unusable */
#define KERN_ALERT      "<1>" /* action must be taken immediately */
#define KERN_CRIT       "<2>" /* critical conditions */
#define KERN_ERR         "<3>" /* error conditions */
#define KERN_WARNING    "<4>" /* warning conditions */
#define KERN_NOTICE      "<5>" /* normal but significant condition */
#define KERN_INFO        "<6>" /* informational */
#define KERN_DEBUG       "<7>" /* debug-level messages */
```

13) 创建Linux内核函数的Man手册

https://blog.csdn.net/qz_43176116/article/details/116660033

14) ATF (arm trust firmware) 完成启动流程

<https://blog.csdn.net/u014426028/article/details/117949006>

15) 如何运用Linux内核访问另外一个模块的函数和变量？

<https://www.ofweek.com/ai/2021-03/ART-201721-11000-30488857.html>

16) Linux驱动分析之MMC子系统框架

<https://developer.aliyun.com/article/1203842>

QNX 驱动开发

1) 查看内核日志指令

slog2info |grep hse

2) devctl的使用

http://www.qnx.com/developers/docs/qnxcar2/index.jsp?topic=%2Fcom.qnx.doc.neutrino.getting_started%2Ftopic%2Fs1_resmgr_devctl_simple.html

3) makefile 结构

https://www.qnx.com/developers/docs/7.0.0/#com.qnx.doc.neutrino.prog/topic/make_convent.html

4) BSP 结构

https://www.qnx.com/developers/docs/7.0.0/#com.qnx.doc.neutrino.building/topic/bsp/bsp_structure.html

5) IPC

6) 信号量

7) resmanager

<http://xilinx.eetrend.com/blog/2018/100016522.html>

8) 查看二进制文件相关信息

比如 devb编译完后生成devb-sdmmc-s32v二进制文件，可以使用use -i 查看。

```
# use -i devb-sdmmc-s32v
QNX_BUILDID=(GNU)4bafaebcc16193a83cfed1918651881f
NAME=devb-sdmmc-s32v
```

```
DESCRIPTION=sd/mmc disk driver
DATE=2020/07/03-10:30:30-EDT
STATE=experimental
HOST=bsp-d3
USER=builder
TAGID=BSP_nxp-s32g-evb_br-700_be-700_919132_17
```

9)查看emmc驱动日志
slog2info

Android驱动开发

1)adb工具使用

https://blog.csdn.net/weixin_36667844/article/details/53513566

2)scrcpy工具使用

<https://blog.csdn.net/was172/article/details/99705855>

3)logcat日志打印工具

https://blog.csdn.net/chen245250566/article/details/105381593?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522165000184516780271514662%2522%252C%2522scm%2522%253A%25220140713.130102334..%2522%257D&request_id=165000184516780271514662&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-105381593.142^v9^pc_search_result_control_group.157^v4^new_style&utm_term=logcat&spm=1018.2226.3001.4187

代码流程规范

1)gerrit 目录规范

https://docs.google.com/spreadsheets/d/1RtysJoRr7N_azNV9U4FtHJbdpDsG1Wa7cZy-1VASOCY/edit#gid=85255605

2) jira使用规范

https://docs.google.com/document/d/1tb9hNj9L0cS_RmXy8NDtC8QLEm9GNKH5KKE-8JFLO7k/edit

3) gerrit代码组织规范

https://docs.google.com/document/d/1f5ZMNR7hPbjbtewgYMoMIsTXWcJyEEbF4_ozDoRtMcY/edit#heading=h.w3z7rr5fjgme

<https://docs.google.com/document/d/1eoYhG3QCC56lUPxPFghtUnBc4R8Np8uT1jK4pXa0EnA/edit#heading=h.imshlmtk00qq>

4) 工程流程规范

<https://docs.google.com/document/d/1ivuBH0dqFIMR5vSKFIT3AGTsY8XjxF7vpWy6vI3p3Hg/edit>

<https://docs.google.com/document/d/1MjgmIRXuG5SiP0hNsFwsF2Nf6k4tc7gSDQiBwELtDZ8/edit#heading=h.1cc3dkwtgm1m>

<https://drive.google.com/drive/folders/1mLJ2jtEvVgaK-jGXkNOmIGPMnMbWAq9Q>

5) 文档规范

<https://drive.google.com/drive/folders/1nDFGEwzqSibmx8ZI4GwQKmCx8gNfdOEn>

6) coverity代码静态扫描

8155在jenkins上的扫描: icapp:

https://jenkins.i-tetris.com/view/coverity/job/kiara_icapps_8155_coverity/

s32g os在jenkins上的扫描:

https://jenkins.i-tetris.com/view/coverity/job/kiara_os_s32g_coverity/

可以手动在Jenkins上起编译, 编译结束后, 在如下网址查看结果:

<https://coverity.i-tetris.com/reports.htm#v10038/p10010>

用户名: reporter 密码: 123456

S32G coverity调试

<https://docs.google.com/document/d/14gm2vTfwyAgaN8g-RWsaUp8jmvylITTHmiVmP6CUZCCg/edit#>

7) 标签使用规范

<https://docs.google.com/document/d/1NSEH7LeaQen-979OEZbhhYy646dYcWH2C9mIIbVL9D4/edit#>

8) MPU 软件开发规范

<https://docs.google.com/document/d/14pfQi0UdSJ8Z2yFhRep4U1kfHRI07SbpdckH1TEPXwM/edit#heading=h.nw1vd4qqmqg8>

GIT使用方法

Git 教程

<https://docs.google.com/document/d/1HpseQl8zAbYtIfxWmACN8IZfAMVRm9NBVL0fwSQ8ILQ/edit>

Git 教程

<https://www.liaoxuefeng.com/wiki/896043488029600>

配置git

git config --global user.name yourname

git config --global user.email youremail

生成以及安装ssh公钥git

ssh-keygen -t rsa -C youremail

cat ~/.ssh/id_rsa.pub Copy the key content and add to Gerrit.

修改 ~/.netrc 增加一行

machine artifactory.i-tetris.com login bingbing.cheng password Cbb123456!

1) 下载代码

git clone "ssh://bingbing.cheng@git.i-tetris.com:29418/apps/kiara/kbox"

2) 查看与切换分支

git branch -a

//查看所有分支(远程+本地)

git branch

//查看本地分支

git branch -vv

//查看本地分支与远程分支对应关系

git branch -m oldName newName

//修改本地分支名称

git checkout name

//切换到name分支

git checkout -b branch-name origin/branch-name

//在本地创建和远程分支对应的分支

3)修改代码

增加一个test.c文件

git status 查看当前修改的文件，红色部分为工作区的修改文件

4)增加代码到本地

git add file.a file.b ...

//增加文件

git add .

//增加当前目录所有文件

5)commit 代码到当前分支

git commit -s

//新建commit

git commit --amend

//修改当前的commit

6)查看提交记录

git log

7)将这个修改推送到远程服务器gerrit进行代码review

git push local remote

git push mega HEAD:refs/for/devel%topic=#1

git push mega HEAD:refs/for/devel=#1

git push origin HEAD:refs/for/dev_1.0

8)拉取服务器最新的代码与本地合并, 并合并冲突(TBD)

git pull origin (远程分支名) --rebase

9)建立本地开发分支

git checkout -b 本地分支名 远程分支名

10)重新回到某次commit

git reset --hard \${commit_id}

11)修改当前代码基于新的分支

git rebase -i \${commit_id}

12)gerrit解冲突

a)git pull到最新代码;

b)在gerrit上找到已经提交代码的commit id, 复制

c)使用git rebase -i \${commit_id}命令

d)git push

13)删除未跟踪的文件

git clean -fd 未跟踪的文件和目录一起删掉

<https://www.cnblogs.com/lsgxeva/p/8540476.html>

14)生成patch

```
git format-patch -1 $(commit_id)
```

15)在本地项目目录创建本地仓库

```
git init
```

输入完命令后项目目录会有一个隐藏的.git文件夹

REPO使用方法

1)首次拉取整个项目

```
repo sync
```

2)只更新本地

```
repo sync --local-only
```

Doxygen使用方法

```
doxygen config-hal
```

```
sed -i "s/Generated by Doxygen\(\ $(doxygen -v)\)*/Megatronic/g"  
doc/latex/refman.tex
```

```
cd ./doc/latex
```

```
make
```

```
cd -
```

```
cp ./doc/latex/refman.pdf ./kiara_hal_api.pdf
```

```
rm doc -rf
```

Docker使用方法

<code>docker image ls</code>	显示加载了的docker镜像
<code>docker ps -a</code>	显示运行的docker容器
<code>Docker kill <容器ID></code>	杀掉运行的docker容器

Docker remove <容器ID>	删除docker容器
docker save -o <tar包名>.tar <镜像名>:<tag>	保存镜像
docker load -i <tar包名>.tar	装载镜像

1)Get the docker

```
git clone "ssh://git.i-tetris.com:29418/cores/kiara/tools/build_dockers"
```

```
cd kiara_docker_ubuntu-16.04
```

2)Build the docker

```
./build-image.sh
```

3)Start docker

```
./start_docker.sh
```

4)Enter the docker./start_docker.sh

```
docker exec -it kiara_build bash
```

劳德巴赫使用方法

1、连接劳德巴赫到jtag口

2、打开trace32软件

3、点击startup开始连接jtag

4、SYStem.CPU 选择CPU

5、在SYStem.Attach之前输入密码

```
SYStem.Option KEYCODE %Byte 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A  
0x0B 0x0C 0x0D 0x0E 0x0F
```

6、SYStem.Attach开始调试

7、break后即可查看CPU以及外设的寄存器

杂七杂八安全小知识

1)Java混淆工具ProGuard

<https://baike.baidu.com/item/ProGuard/5627019?fr=aladdin>

ProGuard是一个压缩、优化和混淆Java字节码文件的免费工具，它可以删除无用的类、字段、方法和属性。可以删除没用的注释，最大限度地优化字节码文件。它还可以使用简短的无意义的名称来重命名已经存在的类、字段、方法和属性。常用于Android开发用于混淆最终的项目，增加项目被反编译的难度。

2)KASLR-内核地址空间布局随机化

<https://blog.csdn.net/y13182588139/article/details/125827111>

QNX小知识

1)log_launch和echo的区别

echo是打印到标准输出，不会留在文件中；log_launch打印到/dev/bmetrics中，并记录了从上电到执行该指令的时间(us)。

```
cat /dev/bmetrics
1041341[us]: ifs1_entry
1162740[us]: init_parti_end
1164571[us]: ifs2_load_start
1194906[us]: cpu1_psci_start
1195455[us]: cpu1_psci_start_done
1195455[us]: cpu2_psci_start
1195821[us]: cpu2_psci_start_done
1195821[us]: cpu3_psci_start
1196187[us]: cpu3_psci_start_done
1196187[us]: cpu4_psci_start
1197683[us]: cpu4_psci_start_done
1197683[us]: cpu5_psci_start
1198781[us]: cpu5_psci_start_done
1198781[us]: cpu6_psci_start
1199971[us]: cpu6_psci_start_done
1199971[us]: cpu7_psci_start
1201711[us]: cpu7_psci_start_done
1201894[us]: ifs1_exit
1271261[us]: bmetrics_service took - 0.000275 seconds,( 1.270986 , 1.271261
): INTERMEDIATE
1277547[us]: DRIVER UART Ready:/dev/ser: INTERMEDIATE
1304250[us]: init_loader_and_launcher: INTERMEDIATE
```

2)qnx查看进程及权限

pidin -p 进程名 user

这个看用户

pidin -p 进程名 -f_

这个看类型

secpol -f capability -t 类型名

查这个类型有哪些能力

3)启动或关闭secpolgenerate进程(qnx侧的权限检查功能)

hlos_dev_qnx/apps/qnx_ap/target/hypervisor/host/build_files/mifs.build.tmpl

中增加: on -T secpolgen_t -u SECPOLGEN_UID:SECPOLGEN_GID secpolgenerate -l

或者在qnx中直接执行:

```
on -T secpolgen_t -u SECPOLGEN_UID:SECPOLGEN_GID secpolgenerate -l &
```

或

```
on -T secpolgen_t -u 20:20 secpolgenerate -l &
```

NXP代码下载

xiaobo.he@megatronix.co @Anxp135792023 软件:

<https://www.nxp.com/webapp/swlicensing/swlicensingIntermediate.sp>

CURL

1)可使用ip地址直接替换url

<https://superuser.com/questions/1587335/replace-server-address-by-ip-in-curl>

如:

```
curl_easy_setopt(curl, CURLOPT_URL, "https://vdcvpapikpi.dfmc.com.cn/api");  
可写成url对应的ip地址  
curl_easy_setopt(curl, CURLOPT_URL, "https://10.92.89.8/api");
```

samba配置方法

https://blog.csdn.net/qg_43516928/article/details/120636842

s32g2_linux_lmc_niagara_cgw

0)参考文档

https://docs.google.com/document/d/1_akhPejM5pYjYEBQgA6efn-UcbneFtpqCNNKU93tS4/edit#heading=h.sz4fc2dbz5oq

a)Download the Repo,

```
repo init -u "ssh://git.i-tetris.com:29418/nxp/s32g/yocto/manifest" -b  
bsp28_master -m lmc.xml --repo-url=ssh://git.i-tetris.com:29418/repo  
--no-repo-verify
```

b)Extract S32G Downloads Cache to your working directory or make a soft link to it.
smb://smb.i-tetris.com/sambashare/Software/S32G/downloads.tgz

c) copy docker to working directory

下载docker

<https://drive.google.com/file/d/1hjh4AlGfiSdW3wyuvoEKxusgOMpQwEYh/view?usp=sharing>

```
cp ~/Downloads/s32g.docker.tar ~/code/s32g_qnx
```

d) exec docker init, run docker

```
./docker/init.sh
```

e)run cmd in docker
source meta-lmc/env.sh
./build.sh all

1)下载S32G_LMC_hse代码并且编译

a)Copy mcore images to the directory below,this file contains hse image
/home/bingbingcheng/code/s32g_lmc/build_silverstonelmc/tmp/deploy/images/silverstonelmc/LMC_CGW_App/Debug/LMC_CGW_App_img_mboot.bin

b)modify /home/bingbingcheng/code/s32g_lmc/meta-mega/recipes-mcu/mcu/mcu.bb
Delete 34 35 36 three lines

c)this file is used for merging mcore_file to uboot_file(line 135 to line 145)
/home/bingbingcheng/code/s32g_lmc/meta-mega/recipes-bsp/u-boot/u-boot-s32_2020.04.bb
append

d)this file is used for merging uboot_file to rootfs_file (line 427)
/home/bingbingcheng/code/s32g_lmc/meta-mega/classes/image_types_fsl_sdcard_mega.bbclass

e)bitbake -c cleanall fsl-image-base

bitbake fsl-image-base

2) 代码烧录

a、tftp环境配置

<https://blog.csdn.net/lr131425/article/details/73917481>

b、boot烧录, 使用windows s32g flashtool下载M核的boot
刷机

bootmode0-1 off
bootmode0-2 off

bootmode1-1 off
bootmode1-2 off

debugmode-1 on
debugmode-2 off

刷机后正常开机

bootmode0-1 on
bootmode0-2 off

bootmode1-1 off
bootmode1-2 off

debugmode-1 off
debugmode-2 off

C、rootfs烧录, 使用uboot里的 tftp工具下载到开发板

3)应用app开发

a)增加代码到该目录下

~/code/s32g_lmc/src/factory/factory_service/tester/
该目录下的makefile可能未生效, 需要再研究一下

b)修改

~/code/s32g_lmc/src/factory/factory_service 目录下的CMakeLists.txt

c)进入docker 用 docker里的./build.sh XXX编译, 编译结果在下面目录下

~/code/s32g_lmc/src/factory/factory_service 目录下的CMakeLists.txt
cd

~/code/s32g_lmc/build_silverstonelmc/tmp/work/aarch64-fsl-linux/factory-tools/1.0-r0/factory-too
ls-1.0/

d)拷贝到开发板

scp ut root@172.28.4.1:/home/root/data/

e)开发板运行程序

./ut -i 0

注意: 如果ut用到了动态库so, 要将动态库的路径写到/etc/ld.so.conf后执行ldconfig将
使所有的库文件都被缓存到文件/etc/ld.so.cache中, 如果没做可能会找不到刚安装的库。

s32g3_mboot_voy_h53_ccu

1)代码编译

A、在windows平台安装multiide集成开发环境、git bash

B、powershell

执行编译脚本

```
E:\S32G_Bootloader_Ghs_Flash_To_Windows\code\framework\realtime\swc\bootloader\platforms\S32G3XX\build> .\build_cmd.bat
```

C、bash执行打包脚本

```
/e/S32G_Bootloader_Ghs_Flash_To_Windows/code/framework/realtime/swc/bootloader/platforms/S32G3XX/build
```

```
$ ./build_pack.sh
```

s32g3_uboot_voy_h53_ccu

1) 编译

2) 调试

3) 从emmc启动

```
=> fatls mmc 0:3
```

```
9687020 ifs-s32g-evb.ui
```

```
43809 silverstone-s32g.dtb
```

```
160976 s32g_pfe_class.fw
```

```
=> fatload mmc 0:3 0x80080000 ifs-s32g-evb.ui
```

```
9687020 bytes read in 234 ms (39.5 MiB/s)
```

```
=> fatload mmc 0:3 0x83e00000 silverstone-s32g.dtb
```

```
43809 bytes read in 14 ms (3 MiB/s)
```

```
=> bootm 0x80080000 - 0x83e00000
```

4) 从tftp server启动

Legacy Image:

```
=> tftp 0x83E00000 silverstone-s32g.dtb
```

```
=> tftp 0x80080000 ifs-s32g-evb.ui
```

```
=> bootm ${loadaddr} - ${fdt_addr}
```

FIT image:

```
=> tftp ${fdt_addr} ifs-s32g-evb.image.itb
=> bootm ${fdt_addr}
```

5) uboot启动kernel验签

参考手册

https://drive.google.com/drive/u/0/folders/11Gmt25z2zmp7Md8oJBA5iH5OsMj6_W71

1. Create a directory in which to store the generated private key and certificate

```
mkdir kernel_keys
```

2. Generate the RSA2048 key pair

```
openssl genrsa -out kernel_keys/boot_key.key 2048
```

```
openssl req -batch -new -x509 -key kernel_keys/boot_key.key -out kernel_keys/boot_key.crt
```

3. Create an .its file for Linux

4. Build the dtb tools

From the U-Boot directory, run:

```
sudo make
```

```
CROSS_COMPILE=/home/bingbingcheng/Downloads/gcc-arm-10.2-2020.11-x86_64-aarch64-n  
one-linux-gnu/bin/aarch64-none-linux-gnu- tools
```

5. Compile the dts into a dtb

```
sudo make
```

```
CROSS_COMPILE=/home/bingbingcheng/Downloads/gcc-arm-10.2-2020.11-x86_64-aarch64-n  
one-linux-gnu/bin/aarch64-none-linux-gnu- silverstone_s32g_defconfig
```

```
sudo make
```

```
CROSS_COMPILE=/home/bingbingcheng/Downloads/gcc-arm-10.2-2020.11-x86_64-aarch64-n  
one-linux-gnu/bin/aarch64-none-linux-gnu- menuconfig
```

```
sudo make
```

```
CROSS_COMPILE=/home/bingbingcheng/Downloads/gcc-arm-10.2-2020.11-x86_64-aarch64-n  
one-linux-gnu/bin/aarch64-none-linux-gnu- dtbs
```

安装设备树编译器：

```
sudo apt install device-tree-compiler
```

首先根据dts文件生成dtb文件：

```
dtc fsl-s32g274a-sec-boot.dts -O dtb -o fsl-s32g274a-sec-boot.dtb
```

6. Build the Linux Kernel

挂载编译出来的文件

```
sudo mount boot.img /tmp/test
```

此处需要将legacy image文件ifs-s32g-evb.ui头64个字节去掉变成ifs-s32g-evb.image才能用来生成fit image

7. Sign the kernel image and pack it into a FIT image

```
sudo tools/mkimage -f security_boot/sign-images.its -K arch/arm/dts/fsl-s32g274aevb.dtb -k security_boot/kernel_keys/ -r security_boot/ifs-s32g-evb.ui.signature.itb
```

```
sudo tools/fit_check_sign -f security_boot/ifs-s32g-evb.ui.itb -k arch/arm/dts/fsl-s32g274aevb.dtb
```

```
sudo tools/mkimage -f security_boot/images.its security_boot/ifs-s32g-evb.ui.itb
```

此处如果报错

```
Verifying Hash Integrity for node 'conf-1'... Verified OK, loading images
## Loading kernel from FIT Image at 7f7bb87dc000 ...
  Using 'conf-1' configuration
  Verifying Hash Integrity ...
OK

  Trying 'kernel' kernel subimage
    Description: unavailable
    Created:     Fri Aug  5 16:14:41 2022
    Type:        Kernel Image
    Compression: uncompressed
    Data Size:   9686980 Bytes = 9459.94 KiB = 9.24 MiB
    Architecture: AArch64
    OS:          Linux
    Load Address: 0x84080000
    Entry Point:  0x84080000
    Sign algo:    sha1,rsa2048:boot_key
    Sign value:   14648c6fc87674ec4b03086fe1b8429fc6d8c1d7f370a5099c940096e3ed4046f27807ac43f69ddbbff398a2664375d
f4ede1fe2eedf6686847ffb69d75274ed4fc7f4a4374b8d0b87e65e84439629cd2efb07790b0a12a7e6ca0d84e32f9646885c231de6fe9f36b
    Timestamp:   Fri Aug  5 16:14:41 2022
    Verifying Hash Integrity ...
sha1,rsa2048:boot_key+
OK

  Loading Kernel Image
Image too large: increase CONFIG_SYS_BOOTM_LEN
Must RESET board to recover
## Loading fdt from FIT Image at 7f7bb87dc000 ...
  Using 'conf-1' configuration
  Verifying Hash Integrity ...
OK

  Trying 'fdt-1' fdt subimage
    Description: snow
    Created:     Fri Aug  5 16:14:41 2022
    Type:        Flat Device Tree
    Compression: uncompressed
    Data Size:   11553 Bytes = 11.28 KiB = 0.01 MiB
    Architecture: AArch64
    Sign algo:    sha1,rsa2048:boot_key
    Sign value:   1d9c4ea8db725197adf1c539c96786f4c9f08fa7118115f2936a47bd8ee99b86de25bdd92c289a1feb06645df385b29
de2e8e61facd6f5d35717726d4b49dc45ff610895687aea623846ffb1bee046251b6bba1b9c1e2118778682c5a83ad7241de408a7683a0e7d4
    Timestamp:   Fri Aug  5 16:14:41 2022
    Verifying Hash Integrity ...
sha1,rsa2048:boot_key+
OK

  Loading Flat Device Tree
## Loading ramdisk from FIT Image at 7f7bb87dc000 ...
  Using 'conf-1' configuration
  Verifying Hash Integrity ...
OK

Could not find subimage node type 'ramdisk'
Signature check Bad (error 1)
```

vim common/bootm.c修改CONFIG_SYS_BOOTM_LEN后

sudo make

CROSS_COMPILE=/home/bingbingcheng/Downloads/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu- dtbs

8. Build U-Boot with an external dtb

To build using the dtb containing the public key, from the u-boot folder, run:

sudo make

CROSS_COMPILE=/home/bingbingcheng/Downloads/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu- EXT_DTB=security_boot/fsl-s32g274a-sec-boot.dtb

9. Boot the board and stop at U-Boot command line

tftp 0x83E00000 ifs-s32g-evb.ui.itb

bootm 0x83E00000

6) Uboot中调试norflash

<https://docs.google.com/document/d/1QOCBYzZ7bAru3h0O2MxubCn6AJSnwrDkii8XXvXDvDU/edit#heading=h.wm4m6k70rcei>

s32g3_qnx7.0_voy_h53_ccu

0) 参考文档、开发指导

<https://docs.google.com/document/d/1zvdaqZ-TKSuGHb5wnxSS6HbbLKGzqQqrIW7QQFlxra3A/edit#>

<https://docs.google.com/document/d/1pYdywq6lXzprBiE99dcIYpINOBt95Svr8m67qNabw4M/edit#heading=h.rx5rgkeymg3z>

1) OS代码编译

a) Download the Repo,

```
repo init -u "ssh://git.i-tetris.com:29418/nxp/s32g/qnx/manifest" -m monza.xml  
--repo-url=ssh://git.i-tetris.com:29418/repo --no-repo-verify
```

b) Extract S32G Downloads Cache to your working directory or make a soft link to it.

smb://smb.i-tetris.com/sambashare/Software/S32G/downloads.tgz

c) copy docker to working directory

下载docker

<https://drive.google.com/file/d/1hjh4AlGfiSdW3wyuvoEKxusgOMpQwEYh/view?usp=sharing>

```
cp ~/Downloads/s32g.docker.tar ~/code/s32g_qnx
```

d) exec docker init, run docker
./docker/init.sh

e)run cmd in docker
source meta-monza/env.sh
./build.sh all

MACHINE_TYPE=silverstonemonza BOARD_TYPE=s32g ENABLE_FACTORY_TEST=false
PRODUCT=s32g QNX_VER=7.0 BUILD_TYPE=debug ./build.sh all

2) 代码烧录

<https://docs.google.com/document/d/1saYKnbx7Cmy29RKWWVpMkfluo8ycRNOP/edit>

进入uboot后执行：

```
tftp 0x80080000 ${image}  
mmc write 0x80080000 0 0x14b800  
Reboot
```

快速执行

```
Cp image ~/tftp_server/Image  
run upgradetftpimage
```

注意：

a)pc不能插2个以上网卡开机，否则只能识别一个网卡。如想外接其他网卡，需开机后，再将外部网卡插入pc！

b) 如果a核跑的是旧的版本,switch已经升级到最新，则a核qnx应用中需要创建vlan4

```
ifconfig vlan4 create  
ifconfig vlan4 vlan 4 vlanif pfe0  
ifconfig vlan4 172.16.104.11 netmask 255.255.255.0 up
```

c)如果a核跑的是旧的版本,switch已经升级到最新，则uboot 中需要配置vlan4

```
setenv serverip 172.16.104.100  
setenv ipaddr 172.16.104.11  
setenv vlan 4  
saveenv
```

d)如果代码是从旧的版本升级到新的版本，tftp不能用了，则需要升级switch，升级步骤如下

https://docs.google.com/document/d/1DhU308zMOuInVU9VIQPgrNk3IzsjFEQmWFU-_UCW_I8/edit

e)针对 acore_img_aboot.bin 中间的2核变4核, 会导致U-Boot tftp下载过程卡死, 暂定对策是增加一个刷写步骤。

1、取得master分支的 acore_img_aboot.bin 文件。

2、刷写master分支的acore_img_aboot.bin

```
tftp 0x80080000 acore_img_aboot.bin
```

```
mmc write 0x80080000 0x4852 0x1000
```

```
mmc write 0x80080000 0x5852 0x1000
```

```
reset
```

3) 单个应用app编译

http://www.qnx.com/developers/docs/6.6.0_anm11_wf10/index.html#com.qnx.doc.neutrino.prog/topic/devel_USINGLIB.html(参考手册)

http://www.qnx.com/developers/docs/qnxcar2/index.jsp?topic=%2Fcom.qnx.doc.neutrino.getting_started%2Ftopic%2Fs1_resmgr_io_devctl.html(参考手册)

开发环境 (linux) 指令:

```
patch
```

//打补丁指令

```
vim images/evb.build
```

//修改开机启动的driver

```
vim images/rootfs.bld
```

//修改打包进rootfs的文件

```
addvariant aarch le
```

//增加编译可执行文件目录

```
addvariant aarch so.le
```

//增加编译so库指令

4) 去掉libkbox.so的符号表

<https://www.technovelty.org/linux/stripping-shared-libraries.html>

```
readelf --syms libkbox.so
```

```
aarch64-unknown-nto-qnx7.0.0-strip --strip-unneeded libkbox.so
```

```
readelf --syms libkbox.so
```

5) 将hse的镜像放到mcore原始镜像中

a.进入到Mapp所在目录

```
cd ~/code/monza-s32g/meta-monza/recipes-mcu/mcu/silverstonemonza
```

b.将ivt拷贝到0x1000(4096)的位置, ivt的大小为256字节

```
dd if=ivt.bin of=S32G_Mcore_Test_Image.bin bs=1 seek=4096 count=256
```

```
dd if=S32G_Mcore_Test_Image.bin.bak of=S32G_Mcore_Test_Image.bin bs=1 seek=4352  
count=9473824
```

c.将Mbootloader拷贝到0xa000(40960)的位置, ivt的大小为524288字节

```
dd if=Mbootloader.bin of=S32G_Mcore_Test_Image.bin bs=1 seek=40960 count=524288
```

```
dd if=S32G_Mcore_Test_Image.bin.bak of=S32G_Mcore_Test_Image.bin bs=1 seek=565248 count=8912928
```

d.将hse的镜像放倒0x00D0A800(13674496)的位置, hse的V0.1.0.0版本镜像大小0x51160(332128)

```
dd if=s32g2xx_hse_fw_0.1.0_1.0.0_pb211015.bin.pink of=S32G_Mcore_Test_Image.bin bs=1 seek=13674496 count=332128
```

```
dd if=S32G_Mcore_Test_Image.bin.bak of=S32G_Mcore_Test_Image.bin bs=1 skip=14006624 seek=14006624 count=808096
```

e.将hse的镜像放倒0x00D95800(14243840)的位置, hse的V0.1.0.0版本镜像大小0x51160(332128)

```
dd if=s32g2xx_hse_fw_0.1.0_1.0.0_pb211015.bin.pink of=S32G_Mcore_Test_Image.bin bs=1 seek=14243840 count=332128
```

```
dd if=S32G_Mcore_Test_Image.bin.bak of=S32G_Mcore_Test_Image.bin bs=1 skip=14575968 seek=14575968 count=238752
```

f.制作成功后可以通过下面两条指令查看mcore的镜像是否成功合入hse镜像

```
hexdump -s 0x00d0a800 -n 64 S32G_Mcore_Test_Image.bin
```

```
hexdump -s 0x00d95800 -n 64 S32G_Mcore_Test_Image.bin
```

g.制作成功后通过下面指令查看ivt是否更新

```
hexdump -s 0x00001000 -n 64 S32G_Mcore_Test_Image.bin
```

6) 如何将hse的驱动自启动

kbox.ko放到 prebuilt/aarch64le/lib/目录下编译到qnx镜像中

```
Vim ~/code/monza-s32g/src/qnx/src/hardware/startup/boards/s32g/s32g_init_raminfo.c
```

修改 s32g_init_raminfo.c

拷贝 startup-s32g-evb

```
cp
```

```
~/code/monza-s32g/src/qnx/src/hardware/startup/boards/s32g/evb/aarch64/le/startup-s32g-evb
```

```
~/code/monza-s32g/src/qnx/prebuilt/aarch64le/boot/sys/startup-s32g-evb
```

修改 evb.build 的内容(必须修改, 否则startup-s32g-evb的修改没法编译到镜像中去)

```
vim ~/code/monza-s32g/src/qnx/images/evb.build
```

编译出来的镜像位置: cd

```
~/code/monza-s32g/build_silverstonemonza/tmp/deploy/images/silverstonemonza
```

7) S32G安全启动

https://docs.google.com/document/d/1R0rrlgLiwUkNMLXN9rhCmv0tlfIB4knNpDJK_iiUPq4/edit#heading=h.8sg_vn7_my5

加入签名脚本到

/home/bingbingcheng/code/monza_s32g_qnx7.1/meta-monza/recipes-images/images/mega-image.bb

8) S32G 通过路由器连接外网

<http://jira.i-tetris.com/browse/MON-2225>

a、连接网线，将路由器、开发板和电脑配到一个网段，路由器WAN口连接外网，LAN口连接开发板和电脑

b、设置路由器，

WAN口连接类型：自动获取IP地址

LAN口IP设置：手动

IP地址：172.16.104.200

子网掩码：255.255.255.0

c、连接路由器，WAN口连接公司网络，LAN口连接S32G，S32G双绞线连接右下角的诊断口，S32G板子ip 172.16.104.11

d、笔记本连接路由器的另外一个LAN口，ping 172.16.104.11 能通，笔记本也能上外网

e、然后修改S32G /etc/startup.sh，将/sbin/route 增加一行/sbin/route add default 172.16.104.200，查看板子默认路由route show

f、然后修改S32G /etc/resolv.conf，修改板子的默认DNS，增加一个 nameserver 172.16.104.200

9) S32G通过TBOX连接外网

a、连接TBOX电源，将TBOX的网线连到OIB-B2开发板的右上角第一个口；

b、在板子上的QNX系统ping TBOX的ip地址172.16.107.20，看能否ping通；

c、在板子上修改/etc/resolv.conf 增加一个百度的DNS服务器nameserver 180.76.76.76；

d、ping www.baidu.com

E、连接tbox查看tbox版本号确认是联通tbox还是电信tbox，

ssh root@172.16.104.20

cat /oemapp/version.txt

岚图H53项目 S32G A核上网连接管理

<https://docs.google.com/document/d/1FZ1lclX2vdB7r3CReda9U3jNFyZKIQ5ziXJMZ8sc3vU/edit#>

S32G通过本地linux电脑上网

A、连接开发版的网线到本地电脑的usb网卡

B、配置开发板的ip: 172.16.104.11,

查看开发板的路由表

```
# route show
Routing tables

Internet:
Destination      Gateway           Flags
default          172.16.104.100   UG
10.92.2.0/24     172.16.106.20    UG
```

C、配置本地电脑的usb网卡:

172.16.104.100

配置本地电脑的路由规则所有网络数据都从eno1转发出去:

```
sudo iptables -A FORWARD -i enx000ec65d5dd4 -o eno1 -j ACCEPT
```

```
sudo iptables -A FORWARD -i eno1 -o enx000ec65d5dd4 -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

```
sudo iptables -t nat -A POSTROUTING -o eno1 -j MASQUERADE
```

查看本地的路由表:

route

```
bingbingcheng@HP-EliteDesk:~/Downloads/PKI/h53_mega_cgw_s32g3_qnx7.1/S32G3_import_root_cert$ route
Kernel IP routing table
Destination      Gateway           Genmask           Flags Metric Ref    Use Iface
default          bogon             0.0.0.0           UG    100    0        0 eno1
10.25.8.0        0.0.0.0           255.255.255.0     U     100    0        0 eno1
link-local       0.0.0.0           255.255.0.0       U     1000   0        0 eno1
172.16.104.0     0.0.0.0           255.255.255.0     U     101    0        0 enx000ec65d5dd4
172.17.0.0       0.0.0.0           255.255.0.0       U      0      0        0 docker0
```

D、在开发板上ping百度

ping www.baidu.com

S32g通过本地windows电脑上网

A、连接开发版的网线到本地电脑的有线网卡

B、配置开发板的ip:

```
ifconfig vlan4 192.168.137.11 netmask 0xfffff00 up
```

配置开发板的默认网关：

```
route delete default
```

```
route add default 192.168.137.1
```

配置开发板的默认DNS,证书申请完成后还原：

```
Cp /etc/resolve.conf /data/resolve.conf.bak
```

```
Echo nameserver 114.114.114.114 > /etc/resolve.conf
```

```
Cp /data/resolve.conf.bak /etc/resolve.conf
```

C、配置无线网卡共享给有线网卡

查看本地有线网卡的ip为192.168.137.1

D、在开发板上ping百度

```
ping www.baidu.com
```

10) OIB-2 使用说明书

<https://docs.google.com/document/d/1HqjIBuSuigsxKtP0ndinR8MW9bnboRcL/edit#>

11)H53证书申请

<https://docs.google.com/document/d/1GqNOK452-cNwmJBFSmOk5j9FzCnF5u9myScZQqXFVrQ/edit>

12)H53证书服务器地址

<https://docs.google.com/document/d/1gRfOtuWathlxByzwwl4ouWFhHgJWQqRh/edit#heading=h.1x6nvabdqbc>

12)使用GDB调试core文件

<https://cloud.tencent.com/developer/article/1177442>

a.拷贝异常文件

```
scp qnxuser@172.16.104.11:/dev/shmem/kbox.ko.core
```

```
/home/bingbingcheng/Downloads/kbox异常退出
```

```
scp qnxuser@172.16.104.11:/data/kbox.ko /home/bingbingcheng/Downloads/kbox异常退出
```

b.输入gdb指令和需要调试的文件

```
~/code/s32g3_qnx710_monza_sop6/src/qnx/sdp/files/qnx700/host/linux/x86_64/usr/bin/ntoaarc  
h64-gdb kbox.ko
```

c.导入可执行文件的core文件

```
(gdb) core-file kbox.ko.core
```

d.输入where定位coredump的位置
(gdb) where

e.查看dump位置
(gdb) up

13)RVDC测试和OTA测试

a.确认已经连接外部网络和系统时间已经修改

b.测试VIN

<https://docs.google.com/spreadsheets/d/1fgrGmfdhbmXdMUPobChpvZV0JWOifeMB/edit#gid=881949956>

c.模拟高低压、OTA前置

https://drive.google.com/drive/folders/1WghpyXFuRQUnmUoLi_NPEiUY1Pw9PQr

d.相关指令

```
mount -o remount,rw /dev/emmc0.qnx6.4 /
```

1、将dev_vin_sn.conf放到/data/目录下

```
cp /data/mqtt_config.json /usr/bin/apps/cloud_agent/etc/mqtt_config.json
```

```
cp /data/vsm_config.json /usr/bin/apps/vsm_manager/etc/vsm_config.json
```

```
cp /data/resolv.conf /etc/resolv.conf
```

```
cp /data/startup.sh /etc/startup.sh
```

```
cd /data/simulate_precondition/
```

```
./vsm_publisher&
```

```
dlt_receive -a localhost | grep -E
```

```
"CMD|DCVP|PERU|ota_manger|rvdc_manger|DCVP|FLS|PCON|PKIU|CHPC|mega_cloud|mega_rvdc|OTX|DIAG"
```

e.云端下发任务名称：

bingbing

VIUML

VIUMR

14)修改soname以及动态库链接问题

修改soname可以应用程序不用重新编译

patchelf --set-soname libnewname.so.3.4.5 path/to/libmylibrary.so.1.2.3
<https://blog.csdn.net/qazw9600/article/details/117997996>

15) 内存泄漏排查

a、通过hogs和top指令可以排查是否有内存泄漏

hogs |grep kbox&

top

B、通过coverity排查内存泄漏点

C、hogs指令简介

在QNX 系统的 hogs 命令:显示进程对系统CPU的占用情况。

1	# hogs							
2	PID	NAME	MSEC	PIDS	SYS	MEMORY		
3	1234	xxx1	1	5%	5%	204k	0%	
4	1345	xxx2	1	6%	6%	2640k	0%	
5	7005	xxx3	1	7%	7%	1144k	0%	

PID: 进程ID

NAME: 进程名

MSEC: 自上次迭代以来此进程运行的毫秒数。

PIDS: 进程在此迭代中,占有进程运行时间的百分比。

SYS: 进程在此迭代中,CPU的百分比。

MEMORY: 内存使用情况

s32g3_qnx7.0_voy_h53_ccu(app)

1) app build

a. 代码下载

repo init -u "ssh://git.i-tetris.com:29418/nxp/s32g/qnx/projects/voyah/h53/manifest" -m default.xml --repo-url=ssh://git.i-tetris.com:29418/repo --no-repo-verify

- b. 代码同步
repo sync
- c. 代码编译
./build.sh all

s32g3_qnx7.1_voy_h53_ccu

0) OS编译烧录开发指导

https://docs.google.com/document/d/1gwbO-H2UH57ZGnb1q6wgeIBF9qSf7_LChFGxZDt7gfs/e/dit#heading=h.1z8l8k9ig7e7

1)代码下载

```
repo init -u ssh://auth-agent@git.i-tetris.com:29418/nxp/s32g/qnx/manifest -b master -m monza.xml --repo-url=ssh://auth-agent@git.i-tetris.com/repo --no-repo-verify
```

2)编译

检查编译环境

```
./meta-alb/scripts/host-prepare.sh
```

QNX 7.1 编译命令

```
MACHINE_TYPE=silverstonemonza BOARD_TYPE=s32g3 ENABLE_FACTORY_TEST=false  
PRODUCT=s32g QNX_VER=7.1 BUILD_TYPE=debug ./build.sh all
```

注意事项

a) 请copy smb服务器上的downloads到工程目录【samba:Software\S32G\qnx_downloads.tar.gz】

b) 编译 报错问题修改

```
vim meta-monza/recipes-mcu/mcu/mcu.bbappend
```

3) SDK集成

8155_android_voy_h53_cdc

https://docs.google.com/document/d/10HmopLEcyJT1MuUv0WT2KgErzZXwbiD_3FNvbfqYE/edit#heading=h.lqag9t8dkt2t

1) 代码编译

a) 环境配置

配置swap大小为32G

```
dd if=/dev/zero of=~/.media/swapfile.img bs=1024 count=32M
```

```
mkswap ~/.media/swapfile.img
```

Add this line to /etc/fstab

```
~/.media/swapfile.img swap swap sw 0 0
```

swapon ~/.code/swapfile.img(编译前查看下swap大小, 如果不是32G就启动下该文件)

配置git ssh repo库

b) 代码下载

```
repo init -u ssh://git.i-tetris.com:29418/qcom/sa/platform/manifest -b dev_hqx.1.2.1 -m platform.xml --no-repo-verify --repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

repo sync

c) 分支代码下载

以[release/monza_hqx.1.2.1_20220221](#)分支为例,

```
repo init -u ssh://git.i-tetris.com/qcom/sa/platform/manifest -b release/monza_hqx.1.2.1_20220221 -m monza_MEGAOS.xml --no-repo-verify --repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

repo sync -d

repo init 用的megaos.xml的, 都用**repo sync -d**拉代码!!!

20220411分支

```
repo init -u ssh://git.i-tetris.com/qcom/sa/platform/manifest -b release/monza_hqx.1.2.1_20220411 -m monza_MEGAOS.xml --no-repo-verify --repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

```
repo sync -d -c --no-tags --force-sync
c)编译
./docker/init.sh
export HQX_TARGET_DEVICE=monza
make all (全部编译)
make clean_all(全部清除)
make android (编译android)
make qnx (编译qnx)
```

```
d)单独编译qnx
cd qnx
source build/envsetup.sh monza
make
```

```
e)单独编译android
cd android
source build/envsetup.sh
lunch monza-userdebug
make -j16
```

2) fastboot代码烧录

a、下载8155上android以及qnx镜像
http://artifactory.i-tetris.com/artifactory/webapp/#/artifacts/browse/tree/General/sa8155-hqx/daily/build/dev_hqx.1.2.1/monza_megaos/H53-OS-dev_hqx.1.2.1-1.130.81.0-20220228115358.tgz

B、板子进入fastboot烧录, 如果板子能进入qnx系统则在qnx系统中
reset -f

c、刷android和qnx所有镜像
cd ~/code/sa8155-hqx1.2.1/burn/monza/qfi_bin/common/build
./fastboot_8155.sh

D、单独刷qnx
cd ~/code/sa8155-hqx1.2.1/qnx/hlos_dev_qnx/apps/qnx_ap/target/hypervisor/host
./flash_qnx.sh

e)单独刷android
cd ~/code/sa8155-hqx1.2.1/android/out/target/product/monza

./flash-android.sh

2) QFIL刷写

<https://docs.google.com/presentation/d/1oShJbvg4bsEM6s1SucFvZ9GSAqCoQ2re/edit#slide=id.p12>

3) 从PC向8155中QNX拷贝文件

- a)USB接电脑
- b)进入cmd界面(先不要进adb)
- c)adb root (这是对8155的android操作)
- d)adb push 文件 文件存放在android路径(将PC上的文件拷贝到android中)
- e)adb shell (进入android)
- f)cp 文件存放在android路径 /ota (将文件拷贝到android根目录下ota中)
- g)通过串口进入minicom查看qnx下/ota/android中的文件,可以发现上一步拷贝的文件就在这里。你可以使用cp命令将该文件拷贝到qnx中任何你想放置的地方。

4) 单独编译qnx的某个模块

- a) ./docker/init.sh
- b) export HQX_TARGET_DEVICE=monza
- c) cd qnx
- d) source build/envsetup.sh monza
- e) pushd mega/apps/kbox_if
- f) make clean
- g) make install

5) 单独编译android某个模块

- a) ./docker/init.sh
- b) export HQX_TARGET_DEVICE=monza
- c) 进入android目录: cd android
- d) 执行: source build/envsetup.sh
- e) 执行: lunch monza-userdebug
- f) cd vendor/mega/system/caVoyTee/ 进入模块内, 执行: mm
- g) 推送编译出来的文件到板子上:
cd ~/code/sa8155-hqx1.2.1
adb push ./android/out/target/product/monza/vendor/bin/voytee_ut /vendor/bin/
adb push ./android/out/target/product/monza/vendor/lib64/libvoytee.so /vendor/lib64

6) 在android中执行可执行文件

- adb wait-for-device ;
- adb root;
- adb remount ;
- adb disable-verity ;

```
adb reboot ;
adb wait-for-device ;
adb root ;
adb remount ;
```

将要执行的文件adb push到/vendor/bin下, 执行 ./voytee_ut 。

注意: 岚图8155-Android编译出的voytee的ut叫 **voytee_ut**

7) 在android中通过wifi连接外网

8155 Android通过办公网络连接外网:

a、开发环境安装scrcpy

```
sudo apt-get install scrcpy
```

b、开发环境安装完毕后执行scrcpy

```
scrcpy
```

c、在模拟的屏幕上通过设置连接wifi

d、连接wifi后板子上可以ping通百度

```
ping www.baidu.com
```

e、板子上ping证书服务器ping不通, 修改板子上/system/bin/oem-iprules-init.sh去掉最后两行默认路由的设置, 再ping证书服务器可以ping通

```
adb push oem-iprules-init.sh /system/bin/
```

```
ping 58.49.84.116
```

8) 8155的qnx环境下导入根证书

Pre.拷贝证书到安装路径下

```
adb connect 172.16.104.40:5555
```

```
adb root
```

```
adb connect 172.16.104.40:5555
```

```
adb push <ca.pem> /data/
```

```
adb shell
```

```
cp /data/ca.pem /ota
```

```
telnet cdc-qnx
```

```
root
```

```
cp /ota/android/ca.pem /data/B2_8155-old/
```

a、在板子上运行

```
./factoryservice
```

b、在开发环境运行

```
./factory_host_tool -i 172.16.104.30 -t write_certificate_test -p 'ca.pem'
```

```
./factory_host_tool -i 172.16.104.30 -t read_certificate_test -p 'ca.pem'
```

9) 在8155的android和qnx之间测试message center

Message_center详细设计

https://docs.google.com/document/d/1BGwNTZ_iTRTwQm6JhCz_iwgeZcVA_dn4-Vvvi7Op3hg/edit#heading=h.ehrslyah7zlh

android侧接收:

```
msg_center_test -r "pki"
```

qnx侧发送:

```
on -T msg_center_t -u msg_center msg_center_test -t "pki" "test-data"
```

通信协议

<https://docs.google.com/document/d/1yHxr-wYIUW86mwed2xd8EKQtXZbEWCVq/edit>

Android侧接收:

```
./remote_certificate
```

Qnx侧订阅TOPIC: (将megaipc_test拷贝成megaipc_test_p到当前路径下)

```
on -T doip_server_t -u doip_server ./megaipc_test_p -s  
s:routineControlOption/request/reply -s s:routineControlOption/result/reply  
-s s:routineControlOption/request -s s:routineControlOption/result -s  
b:uartrpc_svc/uds/req
```

另开一个qnx侧的终端, 模拟M核给A核发送指令:

```
on -T doip_server_t -u doip_server megaipc_test -p uartrpc_svc/uds/resp  
b:030100aabbccdd3101648101(申请证书)
```

```
b:030100aabbccdd3101648102(申请更新)
b:030100aabbccdd3101648103(申请售后)
b:030100aabbccdd3103648101 (查询结果)
```

直接使用msg_center 在qnx侧发送topic测试。注意命令中的sn需要用新的, 即之前用过的sn会失败。vin需要用固定的, 这个vin在吉大的服务端都有注册, 如果用其他的vin会报错。

1) 证书请求:

```
on -T msg_center_t -u msg_center msg_center_test -t "routineControlOption/request"
"{\"function\": \"cert_apply\", \"vin\": \"LDP31B960HG980033\", \"sn\": \"0123a1\"}"
```

查询结果

```
on -T msg_center_t -u msg_center msg_center_test -t "routineControlOption/result"
"{\"function\": \"cert_apply\"}"
```

2) 证书更新:

```
on -T msg_center_t -u msg_center msg_center_test -t "routineControlOption/request"
"{\"function\": \"cert_update\", \"vin\": \"LDP31B960HG980033\", \"sn\": \"0123a1\"}"
```

3) 证书售后:

```
on -T msg_center_t -u msg_center msg_center_test -t "routineControlOption/request"
"{\"function\": \"cert_after_sale\", \"vin\": \"LDP31B960HG980033\", \"sn\": \"0123a1\"}"
```

查询结果:

```
on -T msg_center_t -u msg_center msg_center_test -t "routineControlOption/result"
"{\"function\": \"cert_after_sale\"}"
```

每条指令执行后, 都可以在android侧看到相应的日志打印。

10) IDPS QNX侧更新

可以在qnx侧直接执行: idps_update_trigger

打印出: set idps ota state result:0

表示成功执行了更新, 至于更新是否成功, 还取决于吉大的库

11) IDPS android侧更新

在android侧设置setprop

setprop sys.update.state 24

12) scrcpy不能用

进入qnx, 切换电源状态

```
# megapm_switch_state.sh
1) Switch2ScreenOn      4) Switch2Driving      7) Switch2Exception
2) Switch2ScreenOff     5) Switch2OTA          8) ExitException
3) Switch2Convenient    6) Switch2Abandon      9) break
Select a func to execute: 1
```

```
- - - - - start set all screen to Screen_on - - - - -
-
send data success:
{"value":{"type":0,"mode":1,"priority":0,"reason":"QNX:MEGAPM:Switch"}}
- - - - - set all screen to Screen_on end - - - - -
Select a func to execute:
```

13)android上自启动程序tbox_log_service.rc

1)在 /home/bingbingcheng/code/sa8155_hqx1.2.1/android/device/mega/common/init.mega.rc
中添加自启动tbox_log_service

2) 修改selinux的域

~/code/sa8155_hqx1.2.1/android/device/mega/common/sepolicy/file_contexts

3)修改文件的权限

~/code/sa8155_hqx1.2.1/android/device/mega/common/sepolicy/tbox_log_service.te

查看selinux报的权限问题

adb logcat | grep -iE "tbox_log_service|avc"

代码永久关闭selinux

<https://blog.csdn.net/lwz622/article/details/122014647>

8155_android_changan_c385_cdc

1)刷写镜像

先执行命令: enter_fastboot_mode.sh 停止心跳检测

再执行 fastboot_8155.sh

远程下载: wget --user XXX --password XXX 下载地址

2)下载代码(release分支)

```
repo init -u ssh://git.i-tetris.com/qcom/sa/platform/manifest -b
release/bigsur_hqx.1.2.1_20220217 -m bigsur_MEGAOS.xml --no-repo-verify
--repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

repo sync -d

sop/bigsur_hqx.1.2.1_20230130分支

```
repo init -u ssh://git.i-tetris.com/qcom/sa/platform/manifest -b sop/bigsur_hqx.1.2.1_20230130
-m bigsur_MEGAOS.xml --no-repo-verify --repo-url=ssh://git.i-tetris.com:29418/repo
--repo-branch=repo-1
```

3)adb 长期有效

长安项目，U盘切换 adb 的功能仅限当次系统启动，如果想默认保留 adb 的配置，可以设置如下指令

```
adb shell setprop persist.vendor.usb.mode device
```

4)OTA升级

```
adb root
```

```
adb push 升级包 /data/ota_package/ota.zip
```

触发升级：

```
adb shell
```

```
msg_center_test -t update/action/req
```

```
'{"filename":"/data/ota_package/ota.zip","action":7,"force":true}'
```

监听升级进度及结果

```
msg_center_test -r update/state/notify
```

5)user版本qnx登录密码

```
@mega#cdc!
```

6)镜像版本

http://artifactory.i-tetris.com/artifactory/webapp/#/artifacts/browse/tree/General/backup_version

7)上电开机打印的第一条信息

```
Daemon launched
```

8155_android_changan_c673_cdc

1)下载代码 (release分支)

```
repo init -u ssh://git.i-tetris.com/qcom/sa/platform/manifest -b
release/solvang_hqx.1.2.1_20220907 -m solvang_MEGAOS.xml --no-repo-verify
--repo-url=ssh://git.i-tetris.com/repo --repo-branch=repo-1
```

```
repo sync -d
```

8155_android_chery_t18p_cdc

1)下载代码 (release分支)

```
repo init -u ssh://git.i-tetris.com/qcom/sa/platform/manifest -b  
release/sequoia_hqx.1.2.1_20220517 -m gladius.xml --no-repo-verify  
--repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

```
repo sync -d -c --no-tags --force-sync
```

2)编译

```
./docker/init.sh  
export HQX_TARGET_DEVICE=sequoia  
export HEXAGON_ROOT=/opt/Qualcomm/Hexagon  
make all
```

8155_android_dongfeng_m18_cdc

1)下载代码(主线)

```
repo init -u ssh://git.i-tetris.com:29418/qcom/sa/platform/manifest -b dev_hqx.1.2.1 -m  
acadia.xml --no-repo-verify --repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

```
repo init -u ssh://git.i-tetris.com:29418/qcom/sa/platform/manifest -b  
dev_hqx.1.2.1 -m acadia.xml --no-repo-verify  
--repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

```
repo sync -d -c
```

2)编译

```
./docker/init.sh  
export HQX_TARGET_DEVICE=acadia  
make all
```

3)长安项目-集成HUD

相关jira: [SLV-367](#) [BGS-30767](#) [BGS-35702](#)

功能说明:

主要工作: 集成

相关人员: 钟其江

张志军, 水晶光电对接人

曹斌鑫, HUD通信

后续工作: HUD更新时, 更新下集成文件

文件地址: smb://smb.i-tetris.com/sambashare/peak.wang/长安项目/sjgd_hud

8155_android_mega_hqx_cdc

1)BSP 代码下载

```
repo init -u ssh://git.i-tetris.com:29418/qcom/sa/platform/manifest -b dev_hqx.1.2.1 -m  
platform.xml --no-repo-verify --repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1
```

repo sync -d -c

编译user版本加上: export USER_BUILD=true

若编译过user版本, 需要执行下面命令删除相关路径, 才能重新编译debug版本

```
rm -rf kernel/prebuilts
rm -rf kernel/ship_prebuilt
rm -rf out/target/product/bigsur/obj/kernel/msm-5.4
```

Napa代码下载

repo init -u ssh://git.i-tetris.com:29418/qcom/sa/platform/manifest -b dev_hqx.1.2.1 -m napa.xml
--no-repo-verify --repo-url=ssh://git.i-tetris.com:29418/repo --repo-branch=repo-1

repo sync -c -d

2)查看版本号

Android 下执行命令: `getprop | grep build`

3)查看secpol权限

Qnx下查看 `cat /dev/secpolgenerate/error`

Android下查看 `logcat | grep avc`

4)项目名称及对应代号

长安C385-----bigsur

长安C673-----solvang

岚图H53-----monza

岚图H37-----separang

5)android启动app

可以使用命令启动app, 具体做法:

a.使用adb install 命令安装app;

b.再apk包中找到文件名, 用如下命令启动: `com.jit.ssltest/.MainActivity`为app名称

```
monza:/data/app/~ym9ZTu7h0jwvKV-k4q8LPg== # am start
com.jit.ssltest/.MainActivity
```

6)android telnet qnx问题

```
monza:/ # telnet cdc-qnx
telnet: connect: No route to host
```

原因: 需要满足以下条件才能telnet连接上。

1. Android的ip地址必须和qnx的ip地址在一个网段;

2. android侧/etc/hosts中cdc-qnx/cdc-android ip地址必须和qnx侧的/etc/hosts中的信息一致；
3. android和qnx的ip地址需要和/etc/hosts中ip一致。

7)查看8155 rpmb是否开启

在qnx侧执行下面指令：

```
QSEECOMAPI_Sample -v smplap64 14 1
```

输入2, 如果返回0, 表示已经开启; 返回ffffffff表示未开启

8)pcan停止心跳指令

接CAN1

报文ID 7C2 05 31 01 03 0A 01 00 00

选中后, 按空格发送。会收到04 71 01 03 0A AA AA AA

9)Android查看内核版本号

```
cat /proc/version
```

10)android原生自带的是boringsssl, 不是openssl

11)单独编译android内核

待补充

12)联通tbox发送唤醒pcan指令

CAN-ID:664h 64 10 00 00 00 00 00 00

13)单独编译android内核和刷内核

```
make kernel
```

```
fastboot flash la_boot_a boot.img
```

14)本地打包整体升级包

生成 8155 升级包

```
cd burn/monza/qfi_bin/common/build/ota/fullota
```

```
./ota_pack.sh monza
```

生成升级包位置在 burn/monza/*.tgz

打包整体升级包

```
./build_ota_full.py -s 升级包
```

Build_ota_full.py在 d:workspace

Llinux上在 workspace/my

15)本地ota包升级

用命令升级吧

```
adb root
```

```
adb push 升级包 /data/ota_package/ota.zip
```

触发升级

adb shell

msg_center_test -t update/action/req

'{"filename":"/data/ota_package/ota.zip","action":7,"force":true}'

监听升级进度及结果

msg_center_test -r update/state/notify

16)网盘地址

<https://mega-disk.megatronix.com/>

RYUpgTkmyYYabih4

Liang.cheng

17)qnx gdb 调试(以solvang为例)

a)进入docker环境,并resouce环境变量

./docker/init.sh

cd qnx/

source build/envsetup.sh solvang

b)执行gdb

ntoarch64-gdb mega/apps/solvang/sjgd/vendor/cr/bin/C673

mega/apps/solvang/sjgd/vendor/cr/bin/C673.core

第一个参数是进程文件,第二个参数是对应的core文件

c)查看进程崩溃地方

bt

d)添加库路径

Bt后发现有些库缺失,用下面命令添加库路径

set sysroot //设置系统库路径

set solib-search-path 库路径 //设置库路径

注意:在本地代码替换掉老的库

cp mega_ipc/lib/lib_mega_ipc.so iceoryx/lib/lib_mega_ipc.so

set solib-search-path

mega/prebuilts/common/iceoryx/lib/:mega/apps/solvang/sjgd/vendor/cr/lib/:mega/apps/solvang/sjgd/vendor/cr/shjgd/lib/:mega/prebuilts/solvang/ic/lib/:./Qt/prebuilt/lib/:./Qt/prebuilt/plugins/platforms/

set solib-search-path

./SLV_coredump/install/aarch64le/vendor/cr/lib/:./SLV_coredump/install/aarch64le/usr/lib/graphics/

```
s/qc:/SLV_coredump/install/aarch64le/lib:/Qt/prebuilt/lib:/Qt/prebuilt/plugins/platforms:/Qt/prebuilt/plugins/imageformats:/Qt/prebuilt/qml/QtQuick/Extras:/Qt/prebuilt/qml/QtQuick.2/
```

set solib-search-path

```
./BGS_HUD_coredump/install/aarch64le/usr/lib/graphics/qc:/BGS_HUD_coredump/install/aarch64le/vendor/cr/lib:/BGS_HUD_coredump/install/aarch64le/lib:/Qt/prebuilt/lib:/Qt/prebuilt/plugins/imageformats:/Qt/prebuilt/qml/QtQuick.2:/Qt/prebuilt/qml/QtQuick/Extras:/Qt/prebuilt/plugins/platforms/
```

e)查看所有依赖的动态库状态

info sharedlibrary

18) selinux工具

https://blog.csdn.net/wnw_jackie/article/details/119794665

https://blog.csdn.net/weixin_44021334/article/details/122976023

audit2allow 工具路径是 external/selinux/prebuilts/bin/audit2allow

```
audit2allow -i avc_log.txt
```

```
avc: denied { read write } for name="teefe" dev="tmpfs" ino=30652 scontext=u:r:system_app:s0  
tcontext=u:object_r:teefe_device:s0 tclass=chr_file permissive=0
```

19) qfil 高通9008驱动

<https://gsmusbdrivers.com/download/qualcomm-hs-usb-qdloader-9008-driver-64-bit-windows/>

20) 挂载qlog

在android侧执行

Init.qlog.sh

21) 武汉artifactory服务器

<http://wh-artifactory.i-tetris.com/artifactory/webapp/#/artifacts/browse/tree/General/amberex>

22) qnx侧查看配置字

ecu_config_utility -d

https://docs.google.com/spreadsheets/d/1BZ-BZYKoyzanakDLss_hP7b-E6NXVjAR/edit?usp=sharing&oid=100413765119254899684&rtpof=true&sd=true

跟配置子表格做对比即可

51	HUD	HUD类型	bit0	0x0=无 0x1=W-HUD 0x2=AR-HUD 0x3=内置HUD	0x0	0
52			bit1			0
53	NAP	小憩模式	bit2	0x0=无 0x1=有	0x0	1
54	BACK_CARE	后排关怀模式	bit3	0x0=无 0x1=有	0x0	0
55	CAMPING	露营模式	bit4	0x0=无 0x1=有	0x0	1
56	AUTO_WASH	自动洗车模式	bit5	0x0=无 0x1=有	0x0	0
57	SMOKING	抽烟模式	bit6	0x0=无 0x1=有	0x0	0
58	WHIRL_SCREEN	旋转屏	bit7	0x0=无 0x1=有	0x0	0
59	MIC_NUMBER	声源定位 (麦克风数量)	bit0	0x0= 2 MIC 0x1= 4 MIC	0x0	0

IV:0x52 0x4b 0x70 0x57 0x86 0xf0 0xdc 0x1c 0x49 0x14 0x08 0x22 0x78 0xe9 0x4d 0xc6

密钥: 0x3b 0x68 0x57 0x95 0x6a 0x2d 0x02 0xea 0x0a 0x75 0x2f 0xe6 0x8c 0xfb 0x10 0x9d
0x8f 0xd0 0x05 0xbe 0xa6 0x0d 0x7b 0x2e 0xa3 0x65 0x3e 0xaa 0x14 0xa2 0x67 0x54

文件 共享盘:

fanmin.meng\h53\812c41dc463f8d2ec5fd17fab31efe064f4b55c19304d6a7963ba4911d1b95c7.
enc.full

date;./voy_ut -d;date;cat /dev/pdbg/qcore/tzdiag/qseecom

23)dl日志查看某进程启动时间

```
grep -nr "[us\]" |grep dltlog //启动dltlog的时间, 单位us
```

log_000022_20230416-124449.txt:172528:104070 2023/04/16 12:48:08.233038 1407918418
032 ECU1 QSYM QSLA log info V 6 [250 console.12291 out 1 5 [61496] 4083425[us]: dltlog:
LAUNCH]

如这里表示从启动到dltlog启动时间是4083425 us

注意这里的**2023/04/16 12:48:08.233038** 并没有什么具体意义, 这个时间只是cat的时间

```
grep -nri "Daemon 1" //时间同步后打印的第一条日志
```

log_000022_20230416-124449.txt:116841:68706 **2023/04/16 12:47:48.230170** 31995 000
ECU1 DLTD INTM log info V 1 [Daemon launched. Starting to output traces...]

如这里打印的时间是 2023/04/16 12:47:48.230170,

然后查看对应进程的时间, 如我这里查看HUD, HUD日志打印时间比北京时间多8小时

20230416-20:48:07:651 INFO crinit file:crinit.cpp fun:main line:697 SOC Start

20230416-20:48:07:654 INFO crinit file:crinit.cpp fun:main line:698 version:1.2.0.230209

这样HUD启动时间就是 $4s + (12:48:07:651 - 12:47:48.230170) = 24s$

8155 kbox-安全存储

kbox包含安全存储功能, 原理可以参考: [KBox+TrustZone平台安全存储详细设计文档](#)。

特性: 1. 保存在RPMB 和 SFS中, 互为备份。

2. RPMB需要用指令开启, 开启RPMB后, 无论那种刷机方式都不会擦除掉已写入的文件。

3. 安全存储被擦除的几种情况:


未开启RPMB, qfil或fastboot刷机;


未开启RPMB, 写入文件后, 在开启RPMB(因为开启RPMB会擦除SFS中的文件);


使用我自己编译的工具擦除(这个无论是否开启RPMB都会擦除), 但是工具不会集成到镜像中, 需要手动推进去, 这个工具目前只做测试使用。


4. 存储文件数量上限是60个。目前还没有哪个项目超过这个限制。

用途: 因为读写可能是不同的人或程序进行的, 各项目都有一个表格来维护安全存储的文件名。比如出厂阶段需要写入证书, 出厂后才读取证书使用, 所以必须统一读写的证书名字一致, 确保能够找到。

岚图项目:  岚图 安全存储 使用列表

长安项目:  长安 安全存储 使用列表.xlsx

奇瑞项目:  奇瑞 安全存储 使用列表.xlsx

大众项目:  大众 安全存储 使用列表.xlsx

打开RPMB命令:

在8155 qnx侧执行下面指令:

```
QSEECOMAPI_Sample -v smplap64 14 1
```

输入2, 如果返回0, 表示已经开启; 返回ffffffff表示未开启

输入1, 开启RPMB, 开启后最好再检查下

岚图项目-集成吉大正元PKI SDK

相关jira: [MON-5683](#) [MON-5684](#) [MON-5725](#) [MON-5969](#) [MON-5972](#) [MON-6518](#)

集成文档: [岚图8155-Android集成PKI SDK说明文档](#)

功能说明: 实现一些加解密功能/证书解析功能等。

主要工作: 集成, 适配回调函数接口, 使其能够使用硬件加密接口(kbox)。

相关人员: 无

后续工作: 无


岚图项目-诊断申请证书

相关jira: [PRS-1682](#) [PRS-3015](#) [PRS-3223](#) [PRS-3335](#)

功能说明: 实现诊断申请证书

主要工作: 通过message center发送topic, 申请证书

测试文档: 参考[PRS-3335](#)

相关文档:  SA8155_岚图_证书申请API参考手册

 岚图8155-QNX证书功能.docx

相关人员: 王久远: doip, 发送诊断请求

后续工作: 无

文件地址: smb://smb.i-tetris.com/sambashare/peak.wang/岚图项目/certificate

岚图项目-双向TLS

相关jira:[MON-23815](#) [PRS-4691](#) [MON-16762](#)

功能说明: 实现双向TLS, 岚图一些APP在用, OTA也在用。

主要工作: 增加selinux权限, 使app双向认证的时候可以访问到kbox。

相关人员: 岚图汪涛, 负责统筹

后续工作: 无

岚图项目-获取证书状态

相关jira:[MON-25465](#)

功能说明: 通过message center返回证书状态

相关文档:  岚图8155-QNX证书功能.docx

相关人员: 王久远, doip, 发送诊断请求

岚图项目-集成信大捷安IDPS

相关jira: [MON-4341](#) <https://git.i-tetris.com/q/idps>

功能说明: 集成信大捷安的idps(入侵防御检测)

主要工作: 集成

相关文档:  SA8155_岚图_集成IDPS参考手册

相关人员: 岚图汪涛, 负责统筹

后续工作: idps更新时, 更新一下集成文件即可。

文件地址: smb://smb.i-tetris.com/sambashare/peak.wang/岚图项目/xdja_idps

8155 kbox

主要内容: 使用TrustZone的安全存储和加解密接口

设计文档: [KBox+TrustZone平台安全存储详细设计文档](#)

[KBox+TrustZone平台安全加解密详细设计文档](#)

对外接口文档: [8155平台信息安全模块\(kbox\)客户API参考手册](#)

TrustZone相关文档：

https://drive.google.com/drive/folders/13rnJuurlSJm-UK6e1O2iPR2c_wgRUTcw

说明：

1. 整个kbox功能由两个库的代码组成，分别是：`cores/kiara/kbox`：
<https://git.i-tetris.com/q/project:cores%252Fkiara%252Fkbox> ;
`qcom/sa/quic/la/vendor/mega/system/tee_app`：
https://git.i-tetris.com/admin/repos/qcom/sa/quic/la/vendor/mega/system/tee_app
两者的关系：需要手动拉取`cores/kiara/kbox`代码，本地编译，编译方法见文档：
[高通8155 tee编译及推送方法](#)。编译完成后，需要将编译出来的`takbox.*`和`so`库手动复制到`qcom/sa/quic/la/vendor/mega/system/tee_app`中。具体操作见文档：
： 总之，`cores/kiara/kbox`源码不对外释放，
`qcom/sa/quic/la/vendor/mega/system/tee_app`源码可以对外释放。
`qcom/sa/quic/la/vendor/mega/system/tee_app`在代码中的路径为：
`android/vendor/mega/system/caVoyTee`和`qnx/mega/apps/kbox_if`。其中
`qnx`下的路径是`android`路径的链接。
2. `kbox`设计之初的期望是这样：`cores/kiara/kbox`中的代码基本实现基本功能，然后每个项目都有自己的路径，对于不同项目的需求分别封装`cores/kiara/kbox`中的接口实现，比如现在8155上 `android/vendor/mega/system/caVoyTee`这个路径就是岚图项目的具体实现。但是由于一些原因，现在8155所有项目都用这个名字，如果修改，影响范围较广，不建议修改。这个问题在8295上修改。
3. `kbox`在设计之处，期望能统一不同的硬件平台，对外提供统一的接口。在`cores/kiara/kbox`中的代码基本实现了这个期望。但是在实际使用过程中，由于各平台各项目的需求不同和管理的问题，`kbox`中代码越来越难以实现统一，不同平台的修改会互相影响，比如S32G平台的修改会导致在8155平台中编译不通过。所以手动编译的时候，可能需要修改或屏蔽掉不相关的代码才能编译通过。好在`kbox`代码已经稳定，一般不需要重新编译。
4. 考虑到线程安全问题，`kbox`使用阻塞访问，具体看：[kbox service 实现方法及接口](#)
5. `qcom/sa/quic/la/vendor/mega/system/tee_app`中不只包含了`kbox`的功能，还有一些其他的需求，这点在后面展开说。

8295_android_mega_kiara_cdc

1) 代码下载及编译

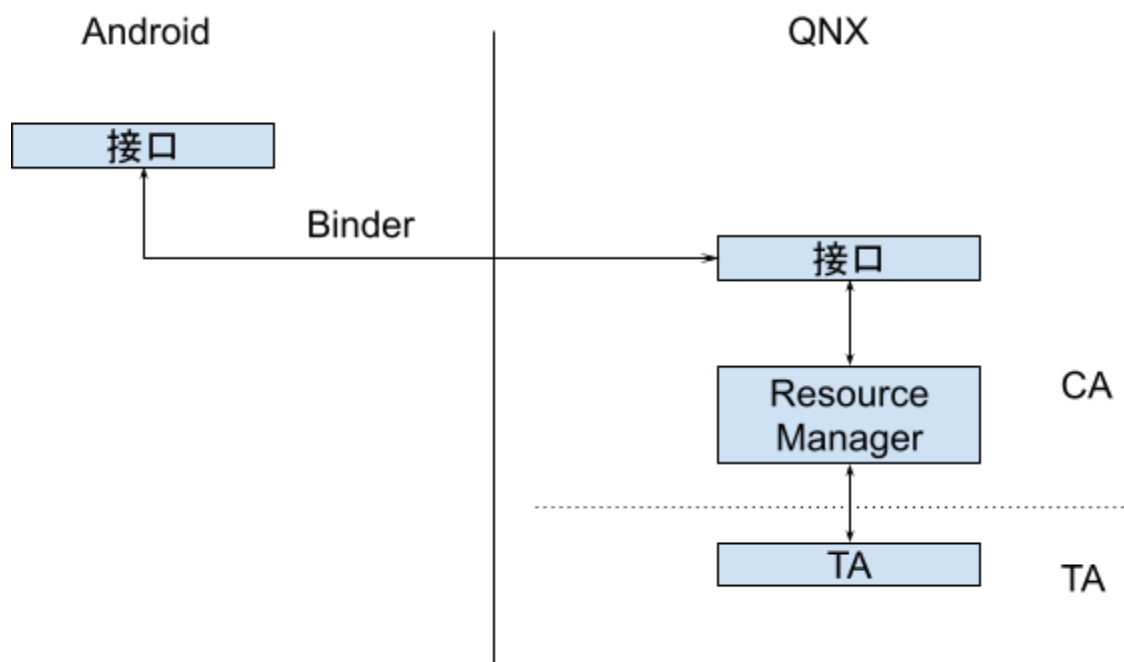
<https://docs.google.com/document/d/1jVB7II3f9GdCHWMzckHSL0M6UPcDjXXwi6XRttVrsM/edit#>

2) 刷机

<https://docs.google.com/document/d/1D7a225JMV07vQPC7fPwmzlyNbmgHd5yZ7KEIhXBVRg/edit#heading=h.1z8l8k9ig7e7>

3) `kbox`设计

a)安全存储作为一个单独的功能和ta。代码层面，在qnx侧实现，ta在tz/trustzone_images/ssg/securemsm/trustzone/qsapps下做一个软连接，编译ta;ca直接在qnx侧编译。代码都在主线库上，不再单独存放。这部分代码不向客户开放。



b)移植之前的8155上的kbox, 实现其功能。

4)编译ta

python build_all.py -b TZ.XF.5.0 CHIPSET=makena --cbt="tasecstorage" --cnb

5)8295 官方文档

<https://createpoint.qti.qualcomm.com/chipcenter/#solution/1489445>

6)向qnx推文件

使用filezilla工具, 若没有, 请使用sudo apt-get install filezilla 命令安装。

a)使用串口查看qnx侧的ip地址, 这里是192.168.1.1

```
# ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33136
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
pflog0: flags=0 mtu 33136
emac0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    capabilities rx=1f<IP4CSUM,TCP4CSUM,UDP4CSUM,TCP6CSUM,UDP6CSUM>
    capabilities
tx=7f<IP4CSUM,TCP4CSUM,UDP4CSUM,TCP6CSUM,UDP6CSUM,TSO4,TSO6>
```



```
enabled=0
address: 00:55:7b:b5:7d:f9
media: Ethernet none
inet 192.168.1.1 netmask 0xffffffff broadcast 192.168.1.255
inet6 fe80::255:7bff:feb5:7df9%emac0 prefixlen 64 scopeid 0x21
```

b)将电脑与开发板网线相连，注意开发板连接到下面那个网口。

c)修改电脑ip地址为和开发板在同一个网段，我的设置如下：

取消(C) 有线 应用(A)

详细信息 身份 **IPv4** IPv6 安全

IPv4 方式

☐ 自动 (DHCP) ☐ 仅本地链路

☒ 手动 ☐ 禁用

☐ 已与其他计算机共享

地址

地址	子网掩码	网关	
192.168.1.109	255.255.255.0	192.168.1.2	

d)在电脑上ping一下开发板的网址，正常应该可以ping通。然后打开filezilla软件，配置如下：
Password是root

sftp://root@192.168.1.1 - FileZilla

File Edit View Transfer Server Bookmarks Help

Host: sftp://192.168.1.1 Username: root Password: Port: Quickconnect

Status: Retrieving directory listing of /ifs/usr/bin ...

Status: Listing directory /ifs/usr/bin

点击快速连接，连接成功。

7)单独编译android模块

./docker/init.sh

cd android

source build/envsetup.sh

lunch lunch msmnile_gvmq-userdebug

cd 到对应模块

mm

8) 8295 kbox

相关jira:[ER-20](#) [ER-22](#) [ER-47](#) [ER-68](#) [ER-70](#) [ER-71](#)

功能说明: 见8155 kbox

主要工作: 安全存储, 密钥管理, 加解密, 签名验签, 随机数等

相关文档: 暂无

后续工作: 密钥管理和加密算法

8155_android_voy_h97_cdc

1、开发环境提供libjitcarapi.so和liboytee.so的库以及根证书

a) ./docker/init.sh

b) export HQX_TARGET_DEVICE=monza

c) 进入android目录: cd android

d) 执行: source build/envsetup.sh

e) 执行: lunch monza-userdebug

f) cd vendor/mega/system/caVoyTee/ 进入模块内, 执行: mm

g) 推送编译出来的文件到板子上:

```
cd ~/code/sa8155_hqx1.2.1/android/out/target/product/monza/vendor/lib64
```

```
scp libvoytee.so qnxuser@10.58.1.61:/home/qnxuser/bingbingcheng
```

2、上位机推送ibjitcarapi.so和liboytee.so的库以及根证书

H97的IVI上位机:

```
ssh qnxuser@10.58.1.61
```

```
adb -s 1234567 push libvoytee.so /data/ryl/H56/
```

```
adb -s 1234567 push ca.pem /data/B2_8155-old/
```

```
adb -s 1234567 push b2.crt /data/B2_8155-old/
```

```
adb -s 1234567 push b2.key /data/B2_8155-old/
```

3、开发板

H97的IVI上位机:

ssh qnxuser@10.58.1.61

H97的IVI开发板:

adb -s 1234567 shell

Cd /data/B2_8155-old/

LD_LIBRARY_PATH=/data/ryl/H56 ./pki_cert LDF2YB3D1GF141581 12345

```
f0505h:/data/B2_8155-old # ./pki_cert
voy_pki_cert_apply:753,ret = 9
voy_pki_cert_apply(LDF2YB3D1GF141585,840022
) failed 9
f0505h:/data/B2_8155-old #
```

```
f0505h:/data/B2_8155-old #
f0505h:/data/B2_8155-old # cat info_log.txt
2022-10-18T15:01:34 jitiot_set_path failed error_code=0x0009.
f0505h:/data/B2_8155-old #
```

确认跟证书的路径是否正确, 是否有读写权限, 确认selinux是否配置, 可以先关闭selinux
setenforce 0

4、OTA应用测试

8155_android_voy_h97c_cdc

1.H97C app build下载

2.H97C app编译

```
cd /home/bingbingcheng/code/monza_h97c_app
source set_env.sh
./build.sh
cd /home/bingbingcheng/code/monza_h97c_app/build/caVoyTee
sudo make VERBOSE=1
scp libvoytee.so qnxuser@10.58.2.103:/home/qnxuser/bingbingcheng
```

3.开发板调试

Cd system/bin/dlt/

./dlt-receive -a localhost&

stop fota_downloader

start fota_downloader

4、业务证书申请

<https://docs.google.com/document/d/1V-3TMYw-jZi3RPwOdkfRdMhgIUH4gGKkgDXFsWVbXw/edit#>

8155_android_faw_tahoe_cdc

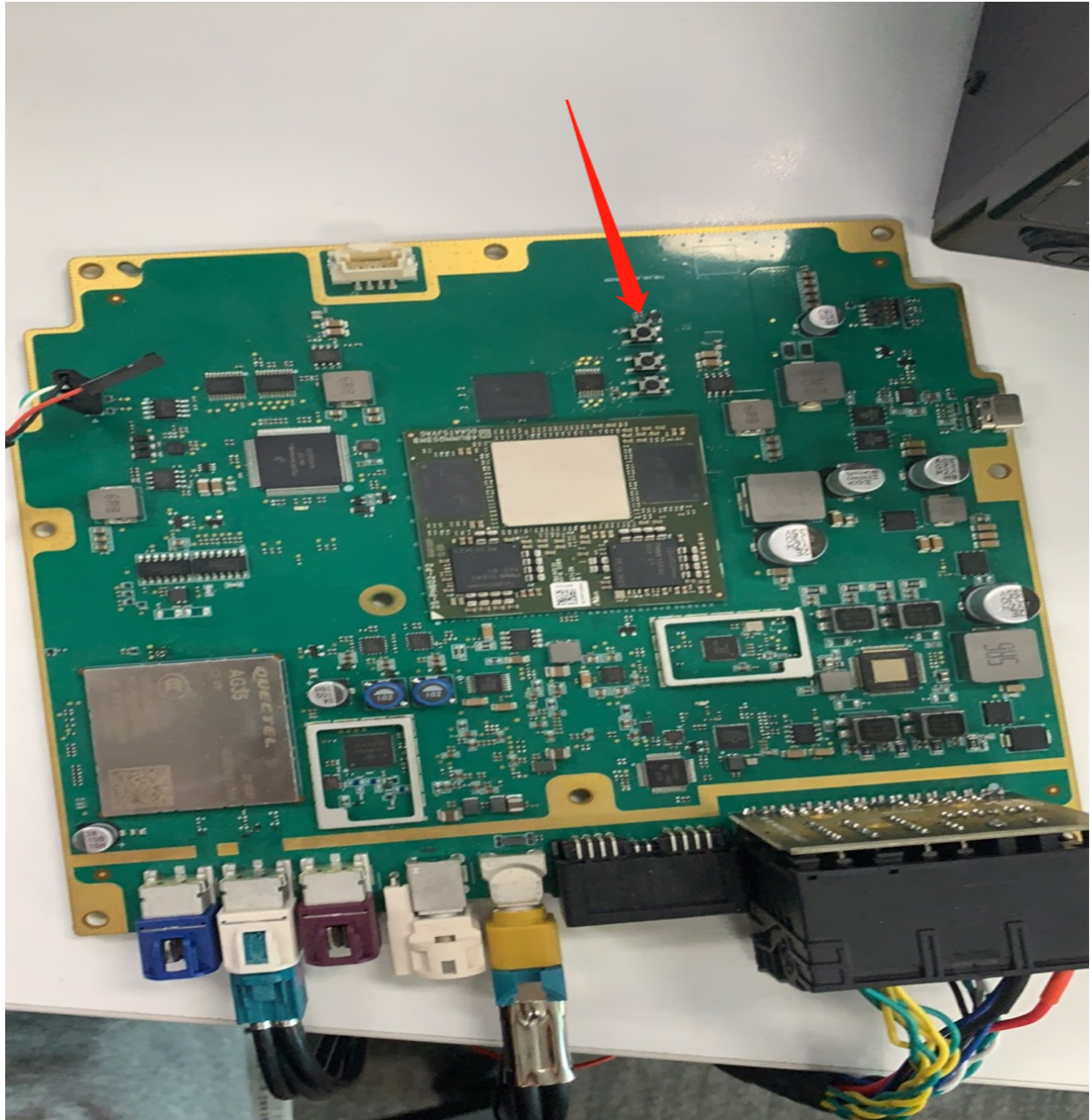
1、8155的镜像下载

http://artifactory.i-tetris.com/artifactory/webapp/#/artifacts/browse/tree/General/faw-vk/tahoe/cdc/dev_hqx.1.2.1/daily/soc/full/tahoe-cdc-8155-dev_hqx.1.2.1-soc_full-1.131.165.0-20230207101702.tgz

2、8155的镜像刷写

按住箭头指向的按钮上电进入9008模式用qfile刷写

<https://docs.google.com/presentation/d/1oShJbvg4bsEM6s1SucFvZ9GSAqCoQ2re/edit#slide=id.p1>



3、下载AG35镜像文件OTA刷写测试

<https://docs.google.com/document/d/1ovuDFdwgd3jgT2Xb2XGUNb8xP1zwQ4LtmdknaCdZRAo/edit#heading=h.nyf1bbkeup04>

1) pc上执行指令

```
adb push IVI_TBOX_VW016000204AE.tgz /sdcard
```

2) 在android上将文件放到/ota/package的目录下

```
cp /sdcard/IVI_TBOX_VW016000204AE.tgz /ota/package/
```

3) 在android上执行刷写测试指令

```
msg_center_test -t "update/tbox/start/req" "{\"ecu_name\":\"tbox\", \"ecu_childdid\":\"1\",  
\"ecu_ota_file\":\"/ota/android/load/IVI_TBOX_VW016000204AN.tgz\",  
\"ecu_sha256\":\"90fc59fc2c5c2936765557687658fc09cf11a23515c42e745d32484a327f19b2\"}"
```

```
msg_center_test -t "update/tbox/start/req" "{\"ecu_name\":\"tbox\", \"ecu_childdid\":\"1\",  
\"ecu_ota_file\":\"/ota/android/load/IVI_TBOX_VW016000204AO.tgz\",  
\"ecu_sha256\":\"e2aef9b22b996585417a801197e4c32d478f361238b477c09c25ff031f636ba4\"}"
```

```
msg_center_test -t "update/tbox/start/req" "{\"ecu_name\":\"tbox\", \"ecu_childdid\":\"1\",  
\"ecu_ota_file\":\"/ota/android/load/IVI_TBOX_VW016000204AZ.tgz\",  
\"ecu_sha256\":\"35b7ca8af77a1fd0b41ad64926c82405fa3184b4d73bdeb1f08689e27881d64e\"}"
```

```
msg_center_test -t "update/tbox/start/req" "{\"ecu_name\":\"tbox\", \"ecu_childdid\":\"1\",  
\"ecu_ota_file\":\"/ota/android/load/IVI_TBOX_VW016000204AY.tgz\",  
\"ecu_sha256\":\"3a5291cde74d9259fc040aeaaf0553d6fe204020cb3b9054793155b71d762a95\"  
}"
```

```
msg_center_test -t "update/tbox/start/req" "{\"ecu_name\":\"tbox\", \"ecu_childdid\":\"1\",  
\"ecu_ota_file\":\"/ota/android/load/IVI_TBOX_VW016000204AX.tgz\",  
\"ecu_sha256\":\"4725473579d8ee5df77c8aebf84744544687302bd92174f9fd2089470f7c5dee\"  
}"
```

```
msg_center_test -t "update/tbox/start/req" "{\"ecu_name\":\"tbox\", \"ecu_childdid\":\"1\",  
\"ecu_ota_file\":\"/ota/android/load/IVI_TBOX_VW016000204AW.tgz\",  
\"ecu_sha256\":\"fc599953947d1cce648a27e796585c884f3af4f097eda72d45ebe4190ef7b73b\"}"
```

```
msg_center_test -t "update/tbox/start/req" "{\"ecu_name\":\"tbox\", \"ecu_childdid\":\"1\",  
\"ecu_ota_file\":\"/ota/android/load/update.zip\",  
\"ecu_sha256\":\"372a7ebb212c808d7d4285be444ab3378e435ac62699c3a0033ed60f7342795f\"}"
```

4) 查看ag35的下载程序日志

```
8155_adb -s f4cc34bd logcat |grep -E "TAHOE" |grep -E  
"mtd|mdm9607|mega_ota_progress|UPDATE|ERROR|system_flag|boot_flag|CMD_FOTA|sha2  
56"
```

5) 8155的android系统上查看android的下载程序日志
logcat | grep -iE "UpdateSer|tbox_client"

```
cat logcat.log.036_2023_06_29_16_36_53 |grep -iE "UpdateService|tbox_client|boot_p"
```

6) 8155的android系统上查看ftp server的端口
netstat -anpt

4、查询tbox版本测试

在android上执行查询版本号指令

```
msg_center_test -t "update/tbox/version/req" "{}"
```

mh5000_linux_voy_h97_tbox

1、连接到开发板

H97的tbox上位机：

```
ssh qnxuser@10.58.2.210  
qnxuser
```

H97的Tbox开发板：

```
adb shell
```

mt2737_linux_voy_h97_tbox

1、连接到开发板

H97的tbox上位机：

```
ssh qnxuser@10.58.2.210  
qnxuser
```

H97的Tbox开发板：

```
adb shell
```

```
adb -s 0123456789ABCDEF push libvoytee.so /mnt/hirain_app/mega/lib
```

```
LD_LIBRARY_PATH=/mnt/hirain_app/mega/lib ./pki_cert LDF2YB3D1GF141581 012345
```

ag551_linux_voy_h97_tbox

1) app build

```
repo init -u ssh://git.i-tetris.com:29418/quectel/ag551/projects/voyah/h97/manifest -b master
--no-repo-verify --depth=1 -m default.xml --repo-url=ssh://git.i-tetris.com:29418/repo
--repo-branch=repo-1
```

2) SDK集成

王涛 5-25 18:34:58

<https://git.i-tetris.com/c/quectel/ag551/prebuilt/ql-ol-sdk/+85076>

程升升 5-25 18:35:05

@孟凡民 对的，复制到src 里面，和其他的app在同一级目录，然后执行biuld.sh all就可以了

王涛 5-25 18:35:08

这个提交已经把kbox的都提交了

王涛 5-25 18:35:53

H97 编译的库可以提交到git clone

"ssh://mars.wang@git.i-tetris.com:29418/quectel/ag551/prebuilt/ql-ol-sdk" 这个repo下的
H97/mega/target/usr/ 下面

王涛 5-25 18:36:29

H97/下面有个脚本可以创建OS build的SDK

王涛 5-25 18:36:42

create_app_sdk.sh

王涛 5-25 18:37:25

把sdk push 到artifactory, 然后更新app build的conf/sdk_download_cfg文件就行

```
curl -uforbidden.city:AP8ApTbqxN3a8gTRVUjjZ5afeBq -T os_sdk_aarch64_h97_1.1.sh
http://artifactory.i-tetris.com/artifactory/vfs-mpu/os/h97-tbox/ag551/os_sdk_aarch64_h97_1.1.sh
```

/opt/ql_crosstools/ql-ag550qcn-le20-gcc820-v1-toolchain/ql-sysroots/usr/lib

kiara@10.25.11.167:/home/kiara/bingbing

3)连接开发板, 推送文件, 执行程序

陶老师的上位机:

ssh kiara@10.25.11.167

pwd: kiara123

scp voy_ut kiara@10.25.11.167:/home/kiara/bingbing

adb shell

LD_LIBRARY_PATH=/mnt/hirain_app/mega/usr/lib

程升升的上位机

ssh qnxuser@10.58.1.239

密码 qnxuser

scp libvoytee.so qnxuser@10.58.1.239:/home/qnxuser

LD_LIBRARY_PATH=/mnt/ota/lib ./pki_cert apply LDF2YB3D0GF141999 000001

ag35_linux_faw_tahoe_tbox

1、推送文件到AG35

1) 推送文件到8155

adb push example_at sdcard

2) 登陆8155终端

adb shell

3) 拷贝数据到/data下

cp /sdcard/example_at /data/

4) 推送到AG35

./8155-adb -s f4cc34bd push example_at data

2、OS代码repo下载

repo init -u "ssh://git.i-tetris.com:29418/quectel/ag35/manifest"

--repo-url=ssh://git.i-tetris.com:29418/repo --no-repo-verify

3、OS代码编译

1)Get the docker

```
git clone "ssh://git.i-tetris.com:29418/cores/kiara/tools/build_dockers"  
cd kiara_docker_ubuntu-16.04
```

2)Build the docker

```
./build-image.sh
```

3)Start docker

```
./start_docker.sh
```

4)Enter the docker./start_docker.sh

```
docker exec -it kiara_build bash
```

5)PIP Install bsdiff4 lxml

```
pip install bsdiff4 lxml
```

6)compile os

```
PRODUCT=ag35-tahoe
```

```
DOWNLOADS=/home/bingbingcheng/code/ag35_linux_tahoe_master/downloads
```

```
OEM_VERSION=VW016000204AZ ./meta/meta-mega/build-quectel-image.sh
```

4、OS镜像下载

http://artifactory.i-tetris.com/artifactory/webapp/#/artifacts/browse/tree/General/faw-vk/tahoe/tbox/master/daily/soc/full/tahoe-tbox-ag35-master-soc_full-243-20230203030455/factory-AG35CEV_AR08A07T4G_OCPU-master-0.1.0.243-20230203030455.tgz

5、OS镜像烧录到ag35芯片

https://docs.google.com/document/d/1E2qYn9m2bCLiGmWXthz_E_HdvVBklxBI-Ke_wswEyZQ/edit#heading=h.dlnhv2knmtt9

6、单个app编译

1)在docker下进入到编译目录

```
cd
```

```
~/code/tahoe_ag35_os/build/tmp/work/armv7a-vfp-poky-linux-gnueabi/soc-service/1.0-r0/temp
```

2)执行编译脚本

```
./run.do_compile
```

3)编译的二进制文件所在目录

```
cd
~/code/tahoe_ag35_os/build/tmp/work/armv7a-vfp-poky-linux-gnueabi/soc-service/1.0-r0/build
```

7、日志收集方案

1)通过8155-adb logcat收集, 大众项目上8155_adb不允许使用

2)rlog方案, s32g采取该方案, 在s32g上通过udp协议传输控制指令, 通过tftp传输数据内容, 传输日志, 在s32g上打包并压缩日志文件发给8155, 大众项目可以采取该方案,S32G和8155的实现可参考下面文档

<https://docs.google.com/document/d/1zbwhTxpbi9trHjBAclpszdWtJO07BaFR/edit>

<https://docs.google.com/document/d/1XADqXdCTloUVDjbGh0whmyMlfsqbgS92ZdiOI0dPY0/edit>

8、Ag35侧增加vlan

```
#vconfig add bridge0 2
#ifconfig bridge0.2 172.28.2.60 netmask 255.255.255.0 up
#ping 172.28.2.10 // rlog server
cd /var/volatile/log/
tftp -p -l ql_manager_server.log 172.28.2.10 58021
```

8155Android侧

```
toybox_vendor vconfig add eth1 4
ifconfig eth1.4 172.28.4.10 netmask 255.255.255.0 up
```

测试

```
/vendor/bin/tftpd -c /log 172.28.2.10 58021 223
tftp -p -l ql_manager_server.log 172.28.2.10 58021
```

9、ag35指令

fotainfo --set-package "/usrdata" --set-type 1	设置刷写参数
ab_full_update "start-full-fota" 2>&1	AB分区刷写
fotainfo --get-activepart	查看当前分区
ql_cmd at AT+QENG="neighbourcell"	查看信号质量等
ql_cmd at AT+CFUN=?	查看飞行模式

ql_cmd at AT+CSUB	查看sub edition
ql_cmd at AT+QCCID	查看ICCID
ftpget -u admin -p 123456 -P 2121 172.28.2.3 IVI_TBOX_VW016000204AA.z ip	通过ftp从8155的/ota/package目录下载文件

s32g3_linux_gac_x4c_ccu

1)需求对外

https://docs.google.com/spreadsheets/d/14yY3SmJJU7Gy2Cs7FzvYcyB_-8ox-qan/edit#gid=1047555187

2)linux方案

<https://docs.google.com/presentation/d/1ogD7vhUvpK2HHu8VYM3aeg7umEUMvD298ZjLcipN8dA/edit#slide=id.p1>

https://docs.google.com/presentation/d/1acLxb81fXGft-tY5sUV9TDz351IQADiV_QDsH_Tp6cc/edit#slide=id.p1

3)IPCF性能以及log方案

<https://docs.google.com/presentation/d/1NHeCSziNiPvgZkum8eA5SgyKhfu-xR5YYZR6vPtmMg/edit#slide=id.p1>

4)SOR

https://docs.google.com/document/d/1f-RLHMMHHRbOY1GUZSJH31yGr_XwXmGv/edit

验收规格书

<https://docs.google.com/spreadsheets/d/1bwlzux6nIMyH7zbZDmXFQSPAZrtiZrRPMVzh059h1E/edit#gid=0>

5)代码下载

```
repo init -u "ssh://git.i-tetris.com:29418/nxp/s32g/yocto/manifest" -m
dev_bsp34.xml --repo-url=ssh://git.i-tetris.com:29418/repo --no-repo-verify
```

6)代码编译

```
MACHINE_TYPE=silverstonemonza BOARD_TYPE=s32g3 ENABLE_FACTORY_TEST=false
PRODUCT=s32g BUILD_TYPE=debug ./build.sh all
```

7)yocto集成meta-selinux

https://blog.51cto.com/u_15127616/4039784

集成步骤:

1.clone selinux源码到sources路径

```
git clone git://git.yoctoproject.org/meta-selinux
```

理论上是应该用sumo分支的, 但是实际上sumo分支编译不过。报以下错误:

NOTE: Running task 352 of 2707

(virtual:native:/sources/poky/meta/recipes-devtools/e2fsprogs/e2fsprogs_1.43.8.bb:do_patch)

NOTE: recipe e2fsprogs-native-1.43.8-r0: task do_patch: Started

NOTE: Running task 1413 of 2707

(/sources/poky/meta/recipes-devtools/e2fsprogs/e2fsprogs_1.43.8.bb:do_patch)

NOTE: recipe e2fsprogs-1.43.8-r0: task do_patch: Started

ERROR: e2fsprogs-native-1.43.8-r0 do_patch: Command Error: 'quilt --quiltrc

Applying patch misc_create_inode.c-label_rootfs.patch

patching file misc/create_inode.c

Hunk #1 FAILED at 979.

Hunk #2 FAILED at 987.

google搜到维护人员的回复, 让使用主分支, commit
id=78eca8242ea5397c4dc0654d62244453b4260151的版本。

2.切换到对应的commit id=8ecad12b2ccb612fdf4906392d26fa6bfae20460

```
git reset --hard 8ecad12b2ccb612fdf4906392d26fa6bfae20460
```

理论上讲要切换到78eca8242ea5397c4dc0654d62244453b4260151, 这个维护人员建议的分支, 但是实测还是报上述错误。

继续回退版本, 发现8ecad12b2ccb612fdf4906392d26fa6bfae20460可用。

3.在build/conf/bblayers.conf结尾加入selinux层

```
BBLAYERS += " ${BSPDIR}/sources/meta-selinux "
```

4.在build/conf/local.conf结尾加入配置

```
DISTRO_FEATURES_append = "acl xattr pam selinux"
```

```
PREFERRED_PROVIDER_virtual/refpolicy ?= "refpolicy-minimum"
```

```
PREFERRED_VERSION_refpolicy-minimum = "2.20170204"
```

```
PREFERRED_VERSION_refpolicy = "2.20170204"
```

```
DISTRO_FEATURES_remove = " sysvinit"
```

```
DISTRO_FEATURES_append = " systemd"
```

```
VIRTUAL-RUNTIME_init_manager = "systemd"
```

```
DISTRO_FEATURES_BACKFILL_CONSIDERED = ""
```

这些信息是在meta-selinux的README, FAQ文件中找到的。

5.修改生成image的bb或者bbappend文件, 把selinux文件打包进镜像

可以通过find sources -name core-image-base*找到image相关的bb或者bbappend文件

其中sources为层所在的文件夹, core-image-base为要集成selinux的镜像的名称。

sources/meta-myir/meta-myir-bsp/recipes-myir/images/core-image-base.bbappend文件中增加如下2行

```
packagegroup-core-full-cmdline
```

```
packagegroup-core-selinux
```

sources/poky/meta/recipes-core/images/core-image-base.bb文件中增加 selinux-image

6.修改内核, 在“General setup“ 和 ”Security options“ 中开启以下功能

```
CONFIG_AUDIT=y
```

```
CONFIG_SECURITYFS=y
```

CONFIG_SECURITY_NETWORK=y

CONFIG_SECURITY_SELINUX=y

CONFIG_SECURITY_SELINUX_BOOTPARAM=y

CONFIG_SECURITY_SELINUX_DISABLE=y

CONFIG_SECURITY_SELINUX_DEVELOP=y

CONFIG_SECURITY_SELINUX_AVC_STATS=y

CONFIG_SECURITY_SELINUX_CHECKREQPROT_VALUE=0

CONFIG_DEFAULT_SECURITY_SELINUX=y

CONFIG_EXT4_FS_SECURITY=y

实验发现CONFIG_EXT4_FS_SECURITY=y不打开，会报以下错误：

[17.955632] SELinux: (dev mmcblk0p26, type ext4) has no security xattr handler

selinux在初始化的时候执行/usr/bin/selinux-init.sh会导致系统shutdown

进一步分析发现脚本中执行/usr/bin/chcon system_u:object_r:root_t:s0 / 时报错

chcon: failed to change context of '/' to 'system_u:object_r:root_t:s0': Operation not supported

7.将生成的镜像烧录进设备，启动系统，修改/etc/selinux/config，然后sync

默认selinux是未开启的，需要在selinux未开启前改为permissive模式，否则启动不了

myd-imx8mm:~# cat /etc/selinux/config

This file controls the state of SELinux on the system.

SELINUX= can take one of these three values:

enforcing - SELinux security policy is enforced.

permissive - SELinux prints warnings instead of enforcing.

disabled - No SELinux policy is loaded.

SELINUX=permissive

SELINUXTYPE= can take one of these values:

minimum - Minimum Security protection.

standard - Standard Security protection.

mls - Multi Level Security protection.

targeted - Targeted processes are protected.

mcs - Multi Category Security protection.

SELINUXTYPE=minimum

8.重启, 修改u-boot环境变量, 启动参数中增加security=selinux selinux=1

然后saveenv, 用boot命令启动系统, 会看到selinux的输出信息

登陆系统, 执行/usr/sbin/sestatus查看状态

至此selinux集成工作已经完成。

8)集成selinux后开启了selinux如果有avc报错通过增加patch的方法解决

1、将错误的日志放入~/Downloads/02_sloution/selinux/avc_log.txt文件中

2、通过audit2allow工具找出错误

audit2allow -i ~/Downloads/02_sloution/selinux/avc_log.txt -p

~/code/s32g3_linux_gac_bsp34/build_silverstonemonza/tmp/work/silverstonemonza-fsl-linux/m
ega-image/1.0-r0/rootfs/etc/selinux/targeted/policy/policy.31

3、拷贝编译目录下的源码

(policy的源码是编译的时候从git上下载下来的, 然后编译的时候打的patch, patch的目录在yocto目录下, 参考步骤6)

Cp

~/code/s32g3_linux_gac_bsp34/build_silverstonemonza/tmp/work/silverstonemonza-fsl-linux/ref
policy-targeted/2.20200229+gitAUTOINC+613708cad6-r0/refpolicy

~/code/s32g3_linux_gac_bsp34/src/refpolicy/

4、通过本地的refpolicy库修改te文件

Cd ~/code/s32g3_linux_gac_bsp34/src/refpolicy/

5、提交修改te文件之后的commit作出patch文件


```
git format-patch -1 $(commit_id)
```

6、将patch文件推送到yocto目录下

```
mv 0001-modify-policy-to-adapt-selinux-enable.patch
```

```
/home/bingbingcheng/code/s32g3_linux_gac_bsp34/meta-selinux/recipes-security/refpolicy/refpolicy/0082-policy-modules-modify-policy-to-adapt-selinux-enable.patch
```

7、重新编译OS

s32g2_qnx7.1_chery_e0x_vcc

1)代码下载

```
repo init -u "ssh://git.i-tetris.com:29418/nxp/s32g/qnx/manifest" -m redwood.xml  
--repo-url=ssh://git.i-tetris.com:29418/repo --no-repo-verify --depth=1
```

2)代码编译

```
MACHINE_TYPE=silverstoneredwood BOARD_TYPE=s32g ENABLE_FACTORY_TEST=false  
PRODUCT=s32g QNX_VER=7.1 BUILD_TYPE=debug ./build.sh all
```

3)代码刷写

注意：1、设置uboot配置pfe的晶振为外部，岚图的pfe的晶振为内部

2、设置uboot配置vlan为空，岚图的vlan为4

3、确认下以太网转换器模式为模式0后

```
=> setenv hwconfig
```

```
"pcie0:mode=rc,clock=ext;pcie1:mode=sgmii,clock=ext,fmhz=125,xpcs_mode=2G5"
```

```
=> setenv vlan
```

```
=> saveenv
```

```
Cp image ~/tftp_server/Image
```

```
=>run upgradetftpimage
```

4)进入系统后配置不用VLAN, 用pfe0的网卡

```
cd ~/code/s32g2_qnx710/meta-monza/recipes-qnx/qnx/qnx-sdp/silverstonemonza
```

```
vim ifs.driver-script_s32g2_7.1.build
```

mt2635_linux_mega_bluetoothkey_ccu

1、开发计划

<https://docs.google.com/spreadsheets/d/17Rumv6D539KUQFOS8xX77nPbKxWR3Y-skQ58o8E6hil/edit#gid=456655134>

2、串口波特率

961200

3、车端方案:

<https://docs.google.com/document/d/1bBb1ciz6lydWXGUXiB8NxXFEZOVHZ-WmoW7OTgF7vCs/edit>

Ag35_linux_changan-mazda_Huron_tbo

X

1、OS代码repo下载

```
repo init -u "ssh://git.i-tetris.com:29418/quectel/ag35/manifest"
--repo-url=ssh://git.i-tetris.com:29418/repo --no-repo-verify
```

2、OS代码编译, 进入docker

Docker load -i

1)Get the docker

```
git clone "ssh://git.i-tetris.com:29418/cores/kiara/tools/build_dockers"
cd kiara_docker_ubuntu-16.04
```

2)Build the docker

```
./build-image.sh
```

以上两步可以省略, 直接装载本地的docker镜像

```
Docker load -i ~/Downloads/02_sloution/docker_images/kiara16.04.tar
```

3)Start docker

```
./start_docker.sh
```

4)Enter the docker./start_docker.sh
docker exec -it kiara_build bash

5)PIP Install bsdiff4 lxml
pip install bsdiff4 lxml

6)compile os

PRODUCT=ag35-tahoe
DOWNLOADS=/home/bingbingcheng/code/ag35_linux_huron_master/downloads
OEM_VERSION=HUR20230831AA ./meta/meta-mega/build-quectel-image.sh

3、音频功能调试

https://embedded.blog.csdn.net/article/details/126071060?spm=1001.2101.3001.6650.8&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-8-126071060-blog-104065481.235%5Ev38%5Epc_relevant_sort&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-8-126071060-blog-104065481.235%5Ev38%5Epc_relevant_sort&utm_relevant_index=14

1)查看kernel的 defaultconfig配置文件(makefile的78行)

vim

~/code/ag35_linux_huron_master/build/tmp/work/armv7a-vfp-poky-linux-gnueabi/generate-quectel/1.0-r0/ql-ol-sdk/Makefile

2)查看kernel config网站需要配置哪些参数

https://www.kernelconfig.io/config_snd_usb_audio?arch=x86&kernelversion=6.3.12

3)修改配置文件增加相关配置选项

Vim

~/code/ag35_linux_huron_master/build/tmp/work/armv7a-vfp-poky-linux-gnueabi/generate-quectel/1.0-r0/ql-ol-sdk/**ql-ol-kernel/arch/arm/configs/mdm9607_defconfig**

4)编译完内核后查看相关配置选项是否生效

Vim

~/code/ag35_linux_huron_master/build/tmp/work/armv7a-vfp-poky-linux-gnueabi/image-quectel/1.0-r0/ql-ol-sdk/ql-ol-kernel/build/**ql-ol-kernel/build/.config**

