

# Developing Audio Products with Cortex-M3/NuttX/C++11

Masayuki.Ishikawa@sony.com

Senior Software Engineer

Sony Video & Sound Products Inc.

# Agenda

SONY

- Product outline
- Typical software development
- Porting NuttX to MCU
- Power management and fast ELF loading
- C++11 and standard library
- Debugging with apps
- adb support and testing with adb
- Demo videos

# Product Outline

SONY

ICD-UX560



- microSDHC and microSDXC support
- Focus and wide mic mode
- Digital Pitch Control

ICD-SX2000



- LPCM recording (up to 96k/24bit)
- FLAC/LPCM playback (up to 192k/24bit)
- Wireless control with REC Remote

NW-WS410



- Water proof (salt water)
- Ambient sound mode
- Up to 12h of battery life

# Hardware Comparison

SONY

Model name	ICD-UX560	ICD-SX2000	NW-WS410
Public release	2015/10	2016/01	2016/02
RTM*	2015/09	2015/12	2015/11
CPU package	TQFP	WLP	WLP
eMMC	4GB, 8GB	16GB	4GB, 8GB
SD card	microSDHC microSDXC	microSDHC microSDXC	-
Audio CODEC	DA7213	DA7211x2 + CXD3774GF	CS47L01
Display	OLED 128x128	STN LCD 128x128	-
Serial Flash	-	Winbond 2MB	-
NFC	-	CXD2249GG (HCI)	-
Bluetooth	-	CSR8811 (HCI)	-
FM Tuner	Si4708	-	-

\*RTM = Release To Manufacturing

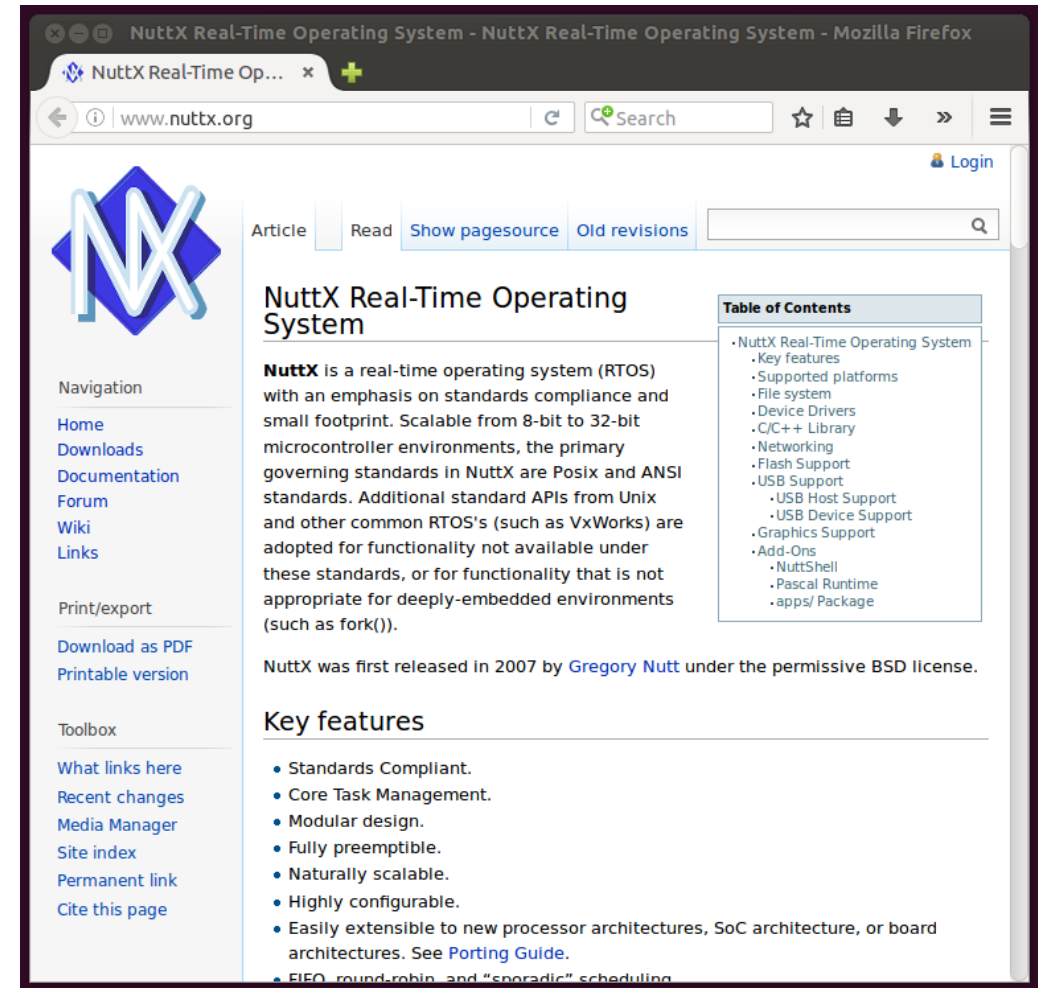
# Typical Software Development

Product models	Android based	Linux based	RTOS based
CPU	ARM Cortex-A series w/ MMU	ARM Cortex-A series w/ MMU	ARM Cortex-M series w/o MMU
Clock	1GHz -	500MHz -	100MHz –
Memory	mDDR2 512MB -	mDDR 64MB -	SRAM 128KB –
SPI Flash	Not used	Not used	Normally used
Toolchain	arm gcc (Google provides)	arm gcc (SoC vendor provides)	Provided by MCU vendor
BSP (Board Support Package)	Provided by SoC vendor	Provided by SoC vendor	Provided by MCU vendor
Programming Language	Java + native (C/C++)	C/C++	C (C++)
Debug commands	Can load dynamically	Can load dynamically	Need to link statically
Debug tools	adb, gdb + gdb server	gdb + gdb server	Commercial ICE

# Why we chose NuttX

From <http://www.nuttx.org/>

- POSIX and libc are supported
  - Can reuse existing software
  - Can reduce training costs
- ELF\* is supported
  - Can divide into small apps
- Driver framework is supported
  - Helps us implement drivers
- Has Linux-like configuration system
  - Helps us develop multiple products
- Many MCUs and boards are supported
  - Helps us port NuttX to new MCU
- BSD license is available



\* ELF = Executable and Linking Format

# Technical Challenges

SONY

- Porting NuttX to MCU
- How to use open tools such as openocd
- Need to consider small RAM size
- How to reuse existing software
- How to apply modern software development



<http://www.nuttx.org/>



**Open On-Chip Debugger**

Free and Open On-Chip Debugging, In-System Programming and Boundary-Scan Testing

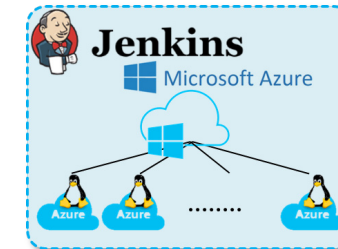
<http://openocd.org/>



<http://wiki.qemu.org/Logo>



<http://www.stroustrup.com/4th.html>



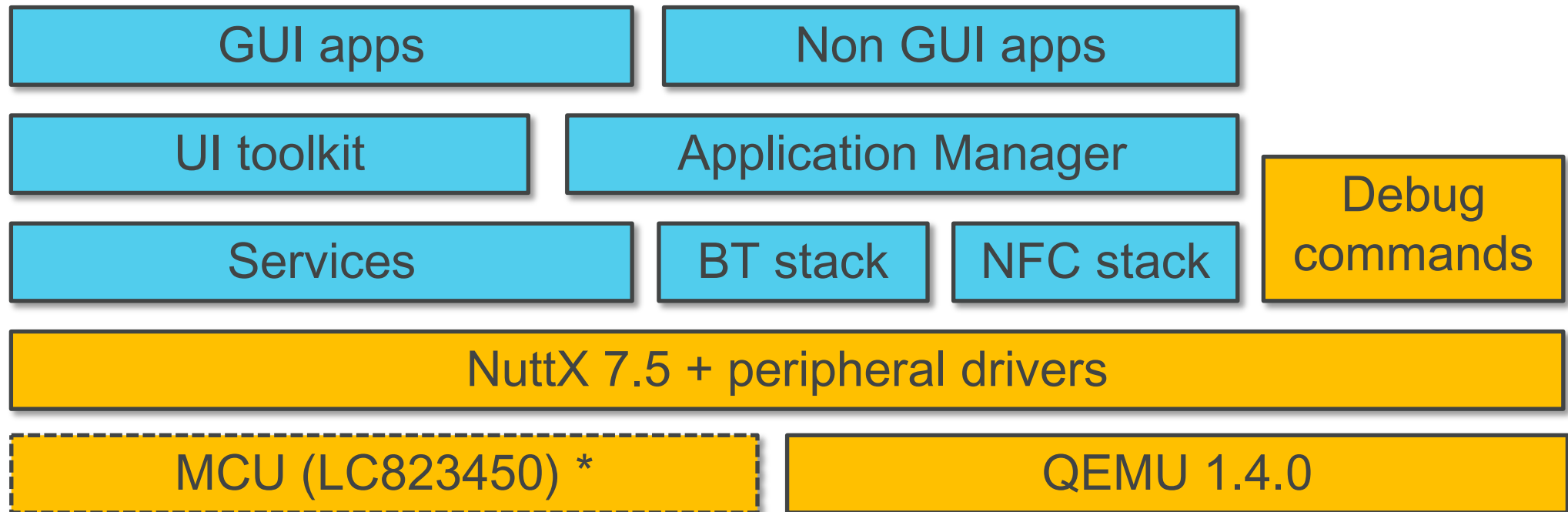
**GitHub**



<https://github.com>

# Software Stack and tools

SONY



\*MCU is not a part of software stack.

tools: gcc-arm-none-eabi-4\_8-2014q1, openocd-0.9.0-dev



# LC823450 Features

SONY

- ARM Cortex-M3 dual core
- 32bit fixed point, dual-MAC original DSP
- Internal SRAM (1656KB) for ARM and DSP
- I2S I/F with 16/24/32bit, MAX 192kHz (2chx2)
- Hard wired audio functions
  - MP3 encoder and decoder, EQ (6-band equalizer), etc.
- Integrated analog functions
  - Low-power Class D HP amplifier, system PLL
  - Dedicated audio PLL, ADC
- Various interfaces
  - USB2.0 HS device / host (not OTG), eMMC, SD card, SPI, I2C, etc.
- ARM and DSP clock max frequency
  - 160MHz at 1.2V
  - 100MHz at 1.0V



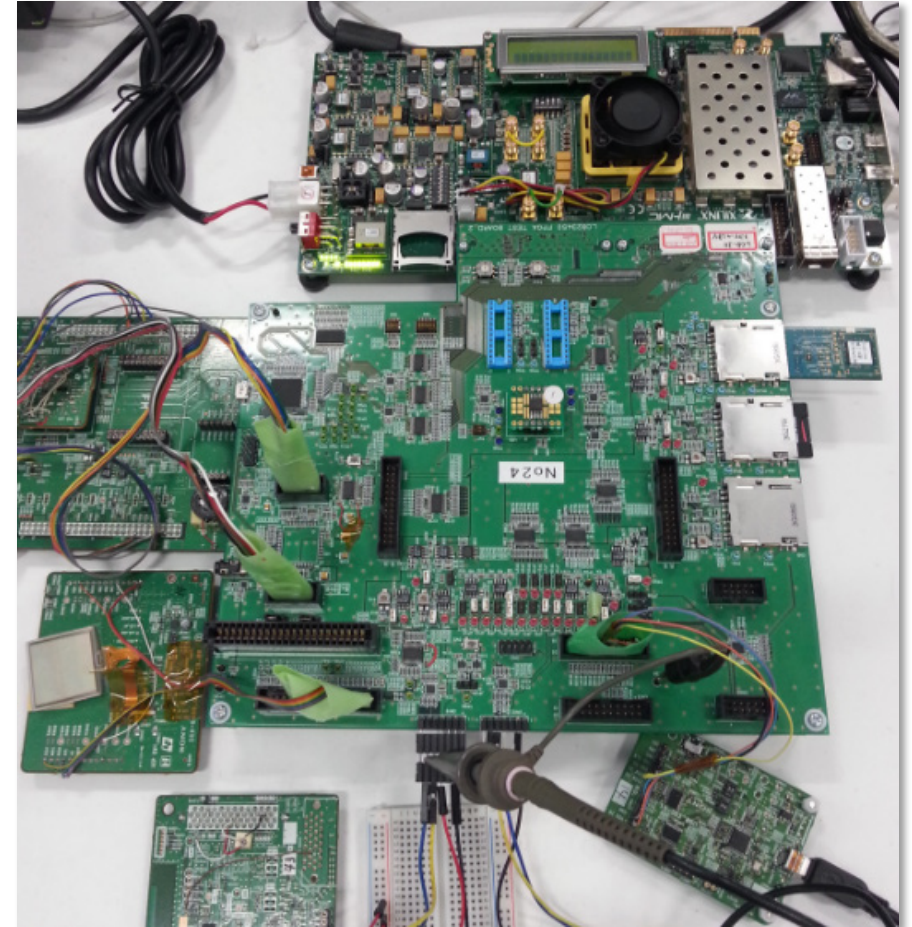
ON Semiconductor LC823450

From <http://www.onsemi.com/PowerSolutions/product.do?id=LC823450>

# Porting NuttX to MCU

Xilinx VC707 + sub boards

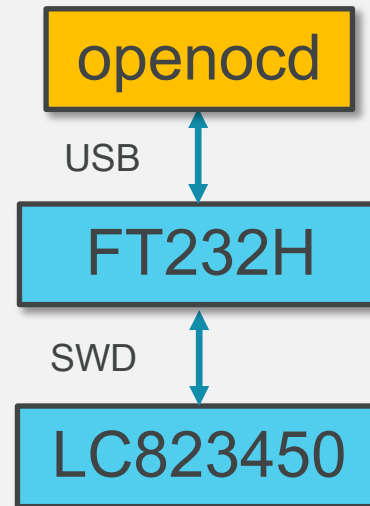
- Started with LC823450 FPGA
  - FPGA code was provided by ON Semiconductor
  - Ported NuttX-7.4 first, then merged 7.5
  - Cortex-M3 (20MHz), NVIC, Timer, UART, GPIO
  - eMMC, SD, DMA, SPI, LCD
  - I2C, I2S, Audio Buffer, Audio CODEC
  - RTC, ADC, USB
  - SPI-Flash, Bluetooth, DSP
- After LC823450 ES arrived
  - Test MAX CPU clock with PLL
  - Test eMMC boot
  - Implement power management
  - Implement suspend & resume



openocd is running on Linux host

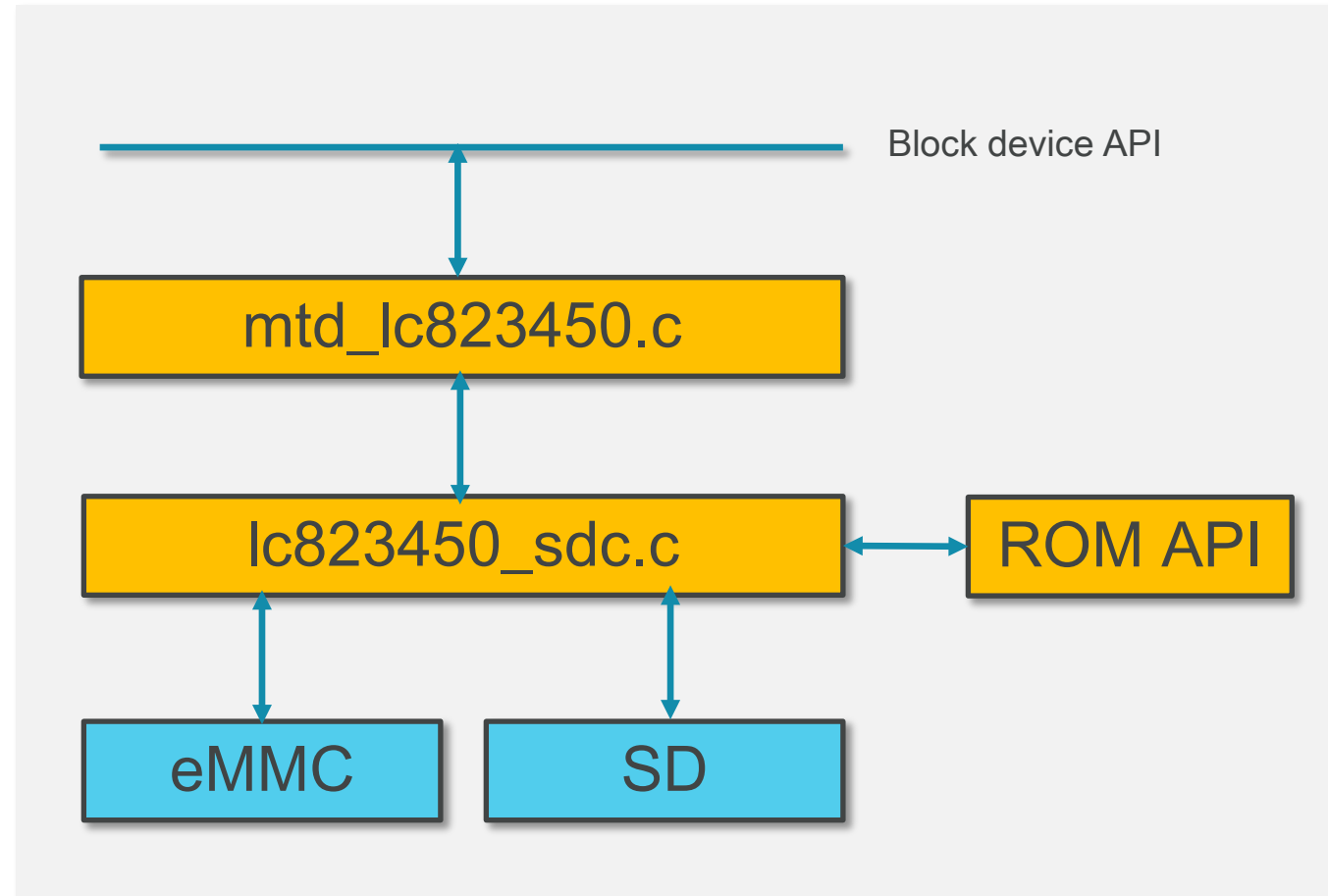
- The very first step
  - Need to prepare before porting NuttX
- Version 0.9.0-dev
  - SWD (Serial Wire Debug) supported
  - With FTDI FT232H board
- Prepare startup scripts
  - Cortex-M sysreset
  - Be careful with adaptor clock
- Load the program to SRAM
- Load to SPI-Flash
  - Implement SPI-Flash driver

```
Terminal
File Edit View Search Terminal Help
$ sudo ~/bin/openocd -s ./tcl -f ./tcl/board/lc823450_sony.cfg -c init -c "reset halt"
Open On-Chip Debugger 0.9.0-dev-dirty (2016-08-15-12:00)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.sourceforge.net/doc/doxygen/bugs.html
adapter speed: 300 kHz
Info : FTDI SWD mode enabled
cortex_m reset config sysresetreq
Info : clock speed 300 kHz
Info : SWD IDCODE 0x2ba01477
Info : lc823450.cpu: hardware has 6 breakpoints, 4 watchpoints
#### reset-start
adapter speed: 300 kHz
0x40080124: 00000000
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x02200474 msp: 0x020779a8
#### reset-end
0x40080124: 00000000
adapter speed: 6000 kHz
```



# eMMC/SD driver

- Implement as a block device
- Call ROM APIs
  - identifycard, readsector, writesector, etc.
  - instead of using eMMC driver in NuttX
- Use fixed partitions
  - due to ROM code restrictions
- Use DMA
  - to reduce CPU load
- Work with hotplug driver
  - i.e. SD card detection
  - newly introduced



# File Systems

df & mount on ICD-SX2000

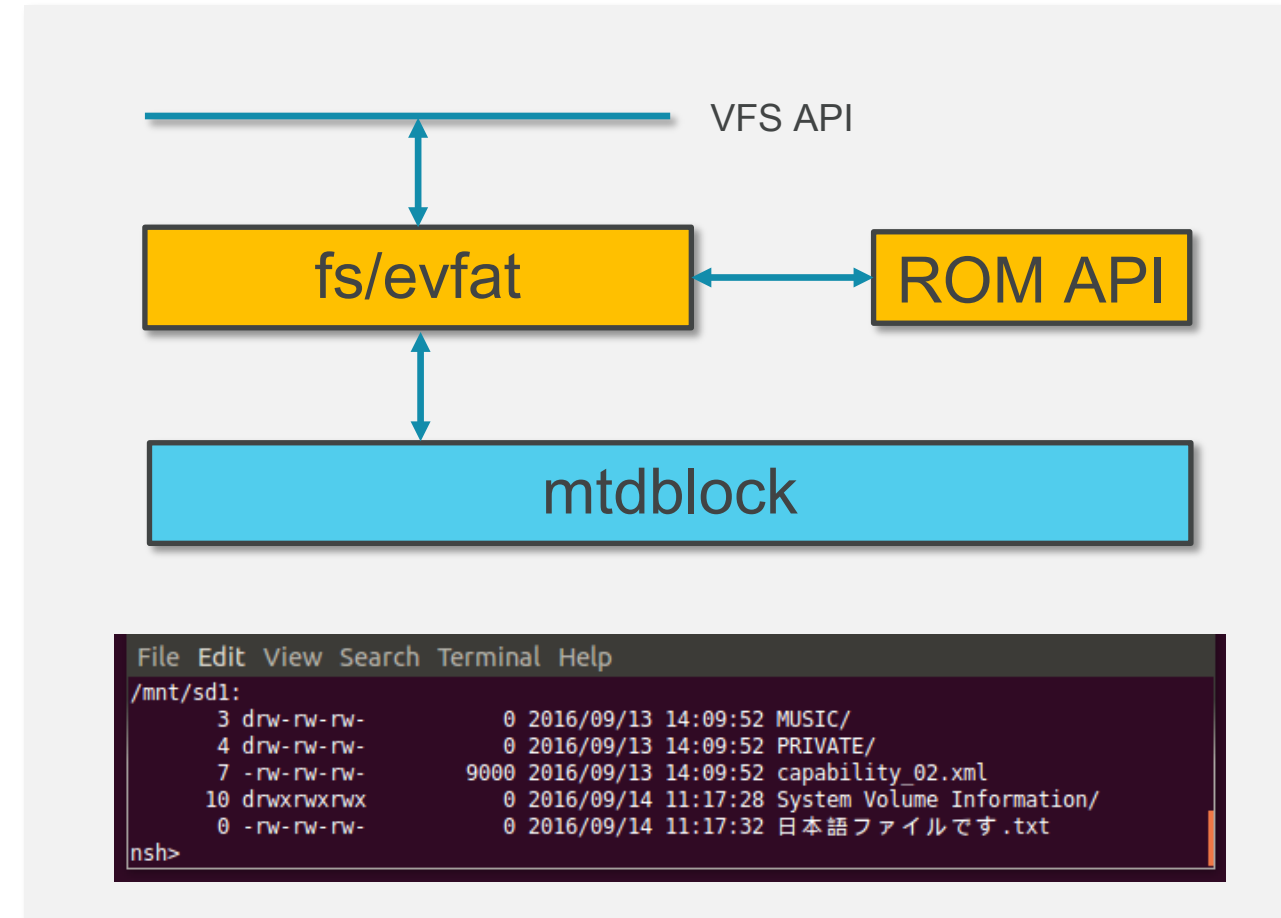
- Using NuttX file systems
  - procfs for debugging, wake\_lock, etc.
  - vfat for program files, properties, database
- Add eVFAT
  - Provided by ON Semiconductor
  - FAT32, exFAT supported
  - IC recorder specific APIs supported
  - Cache control supported
- Others
  - Add read only option
  - Add remount option

```
Terminal
File Edit View Search Terminal Help

nsh> df
  Block  Number
  Size  Blocks    Used Available Mounted on
16384   4349     791     3558 /cache
16384   8188        6     8182 /db
16384   1022        7     1015 /etc
16384   1022        4     1018 /log
32768 471232     426   470806 /mnt/sd0
131072 486992        3   486989 /mnt/sd1
      0        0        0         0 /proc
16384   4093    1024     3069 /system
nsh> mount
/cache type vfat (rw)
/db type vfat (rw)
/etc type vfat (ro)
/log type vfat (rw)
/mnt/sd0 type evfat (rw)
/mnt/sd1 type evfat (rw)
/proc type procfs (rw)
/system type vfat (ro)
nsh> 
```

# eVFAT

- Implement using NuttX VFS\* APIs
- Call ROM APIs
  - mount, open, read, write, lseek, ..., etc
- Add new IOCTLs
  - divide, ...
- Add UTF-8 from/to UTF-16 conversion
- Use a dedicated stack like IRQ
  - Because some APIs need more stacks



\* VFS = Virtual File System

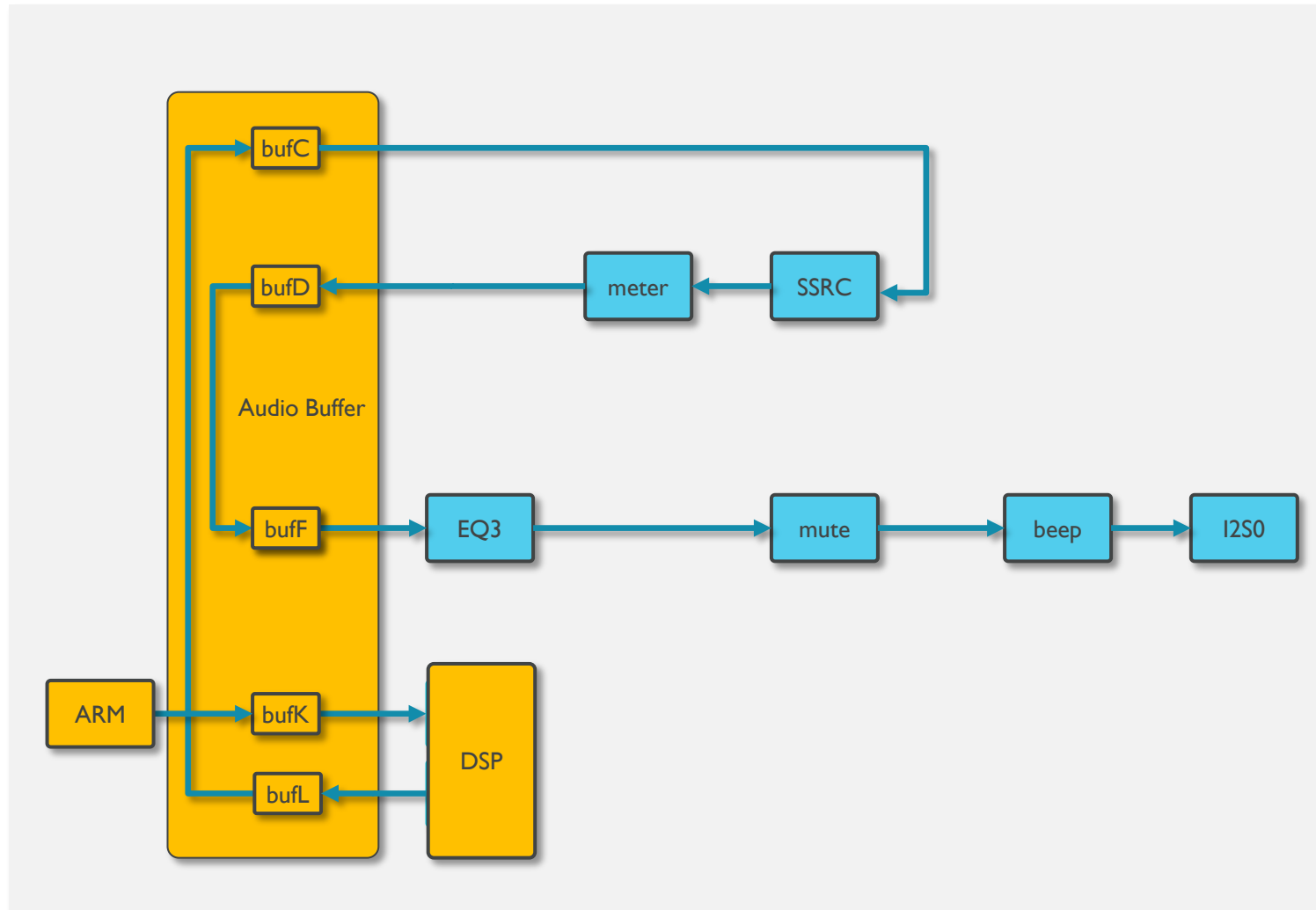


- [illegible]

#ARMTechCon

# Audio Playback Example (AAC,...)

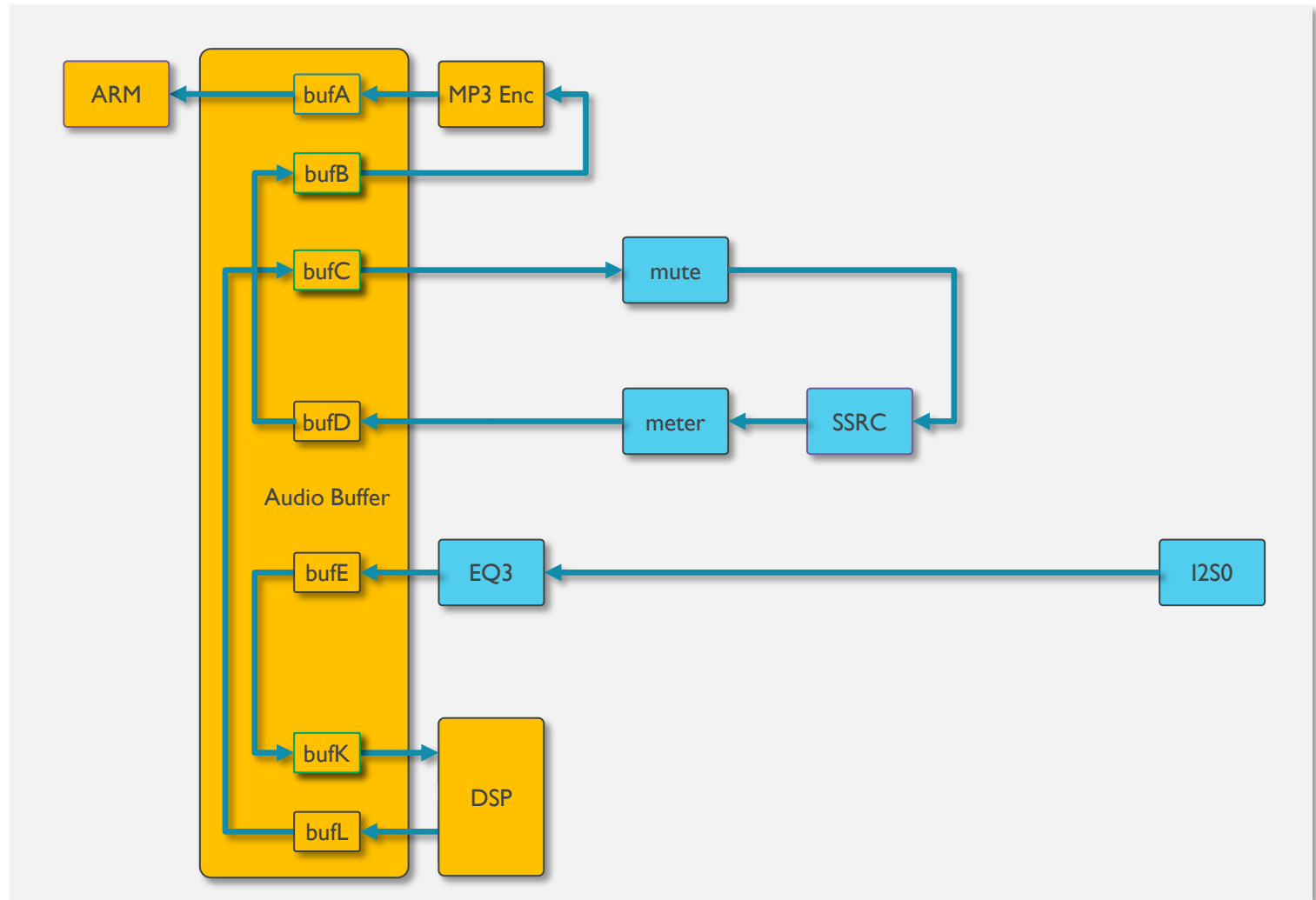
- Cortex-M3
  - Set up audio routing and buffers
  - Set up external audio CODEC
  - Load DSP code and boot
  - Read a file on eMMC/SD
  - Parse audio frame
  - Write the frame to the Audio Buffer
- DSP
  - Decode the frame
  - Do post process
  - Write PCM data to the Audio Buffer





# Audio Recording example (MP3)

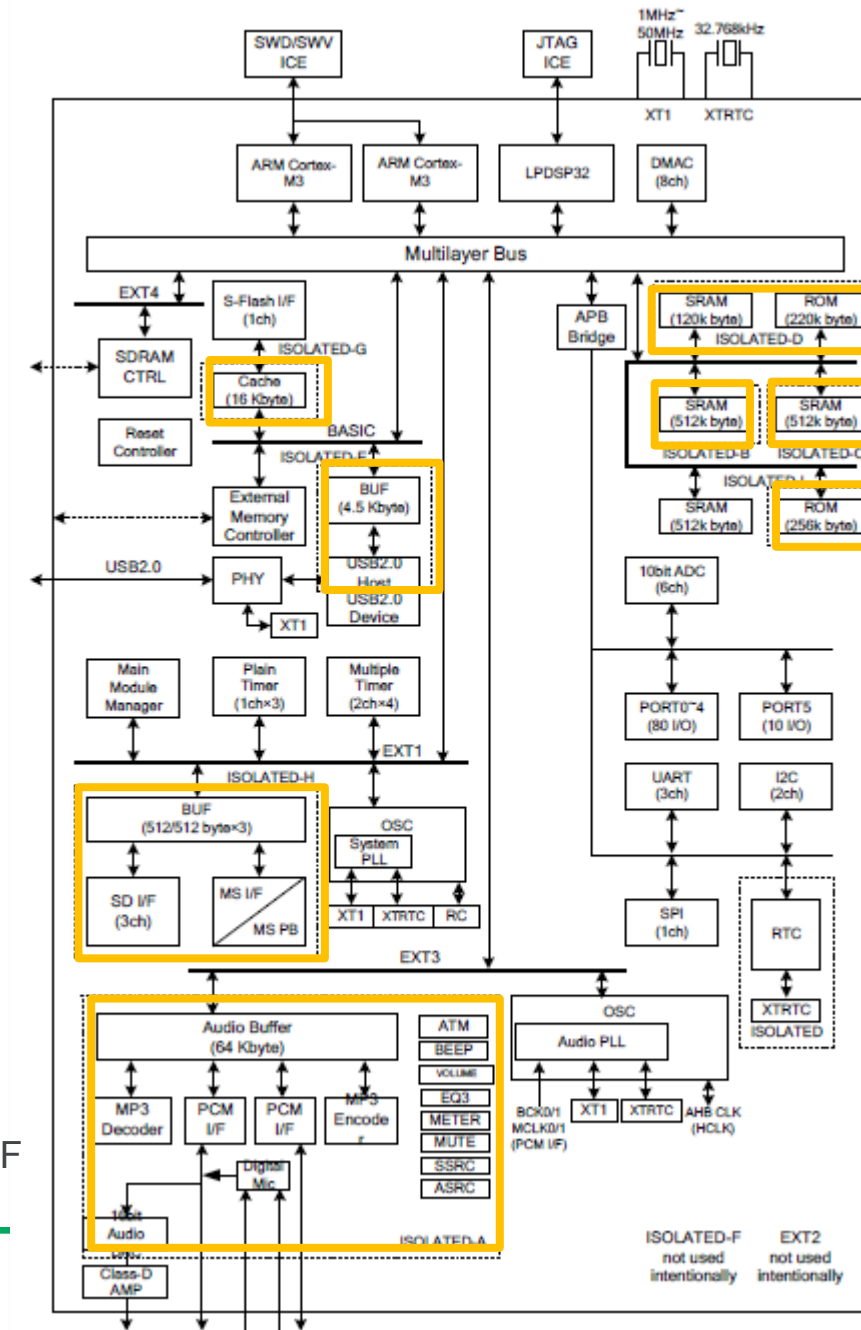
- Cortex-M3
  - Set up audio routing and buffers
  - Set up external audio CODEC
  - Load DSP code and boot
  - Wait for the buffer to be filled
  - Write the audio frame to a file
- DSP
  - Wait for the audio buffer from I2S
  - Perform preprocessing of the frame
  - Write to the Audio Buffer
- MP3 Encoder
  - Wait for the audio buffer from DSP
  - Encode the PCM data
  - Write to the Audio Buffer



# Power Management

- Clock gating
    - Disable clocks for unused blocks
  - Power gating
    - Disable power for unused blocks
  - ISOLATED-A : Audio
  - ISOLATED-B/C/D : SRAM
  - ISOLATED-E : USB Host
  - ISOLATED-G : SPI-Flash cache
- DVFS
  - Suspend & Resume

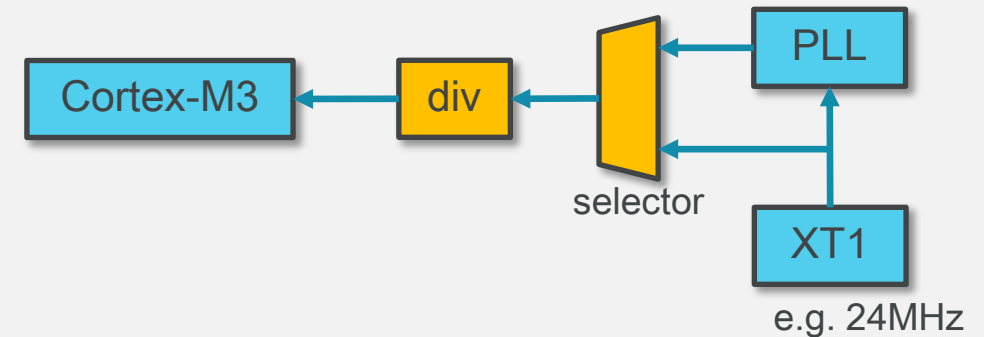
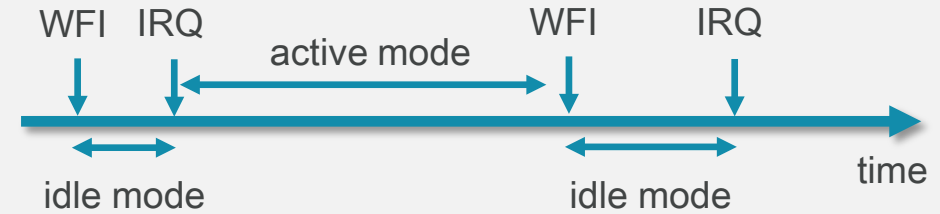
From LC823450-D.PDF



SONY

# DVFS\* (1/2)

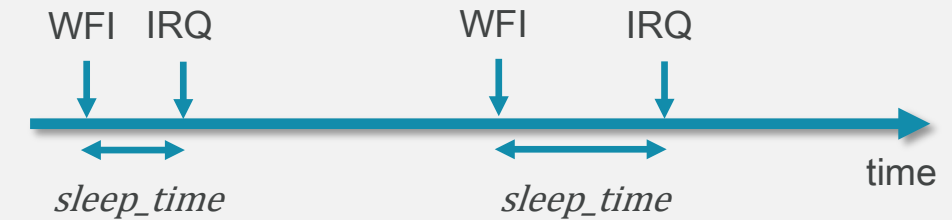
- Voltage control
  - 1.2V at 160MHz, 1.0V at 100MHz
- Clock control
  - CPU/DSP clock, AHB clock
- Clock table example
  - Active mode: 160M/80M/40M/24M
  - Idle mode: 24M/12M/6M/3M
- Autonomous control
  - Calculates the idle ratio
  - Controls divider and selector
- Boost the clock
  - when the keys are pressed
  - while loading applications



\* DVFS = Dynamic Voltage and Frequency Scaling

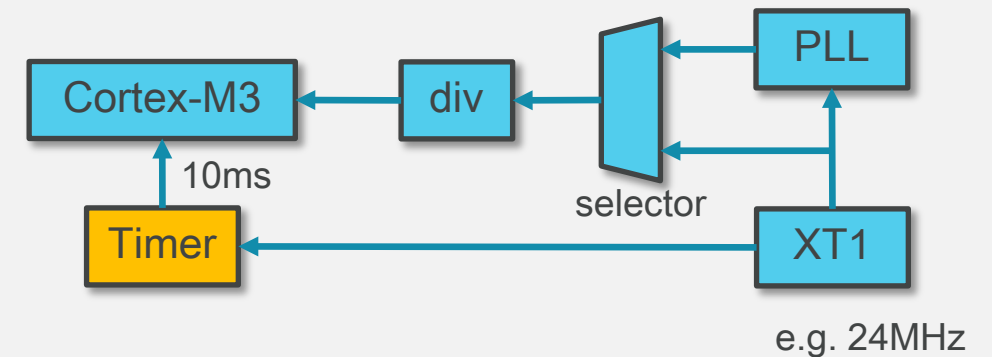
# DVFS (2/2)

- NuttX has CPU load monitoring
  - To monitor each task load
  - But the load in IRQ handler is not considered
- Need more accurate idle time
  - With simple calculation
  - Accumulate sleep time in usec during WFI
  - Calculate the idle ratio
- Use an internal H/W timer for tick
  - Instead of SYSTICK in Cortex-M3
  - As the timer is not affected by clock change
  - Results in simple calculation



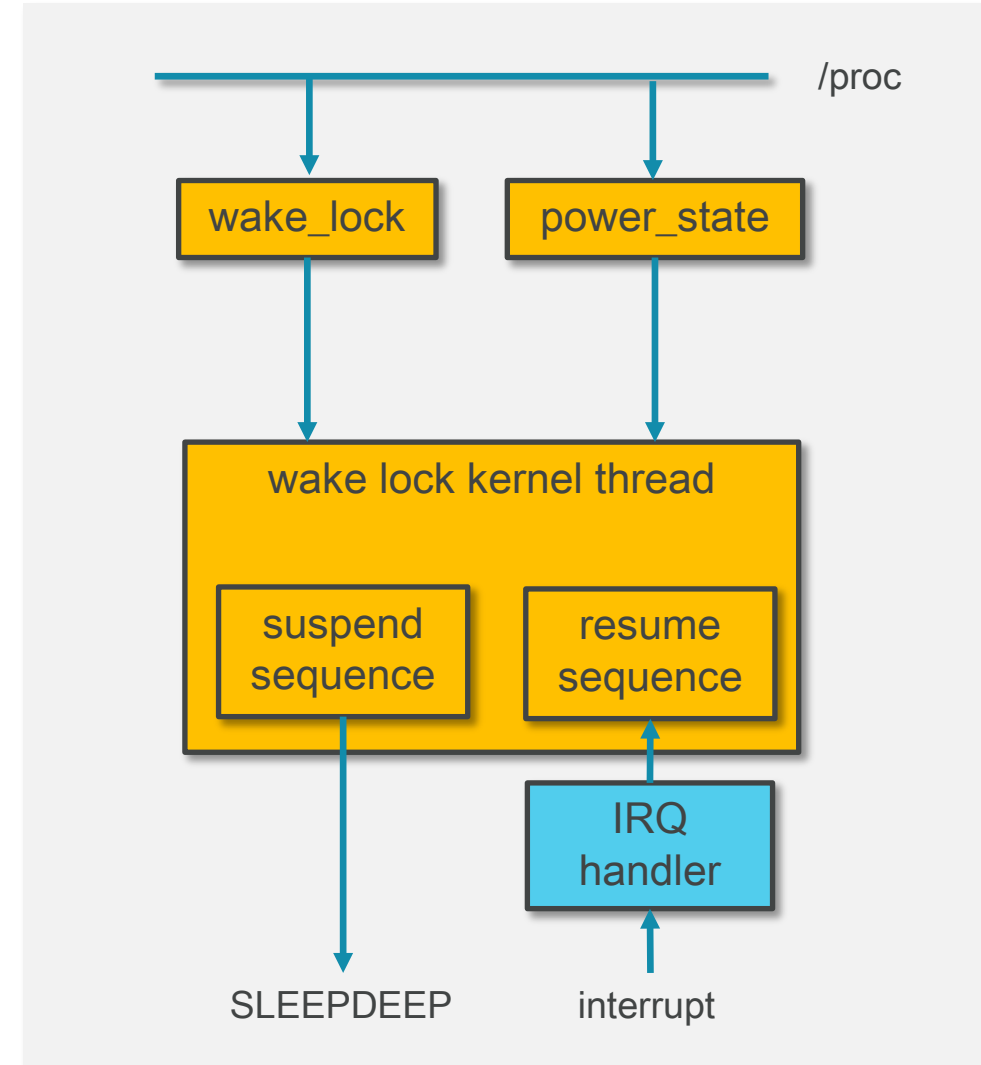
$$idle\_time = \sum sleep\_time$$

$$idle\_ratio_n = \frac{(idle\_time_n - idle\_time_{n-1})}{interval}$$



# Suspend & Resume

- Introduce wake\_lock
  - Provides APIs similar to those of Android kernel
  - If the power state is set to “mem” and no wake\_lock exists, enter to SLEEPDEEP mode
- Implementation
  - Use a kernel thread
  - Power down unused blocks
    - e.g. Audio, SD, etc...
    - USB Suspend must be considered
  - Set SLEEPDEEP flag in NVIC and issue WFI
  - Woken up by interrupt when the following are received
    - i.e. GPIO, RTC alarm, USB resume
  - Power on the blocks if needed
  - Synchronize the kernel time with RTC



- Motivation
  - To overcome limited memory
  - More flexible than overlay
- Divide into small applications
  - e.g: Home, Settings, Play, Rec, ...etc.
- Can use separate debug commands
  - without linking them to the applications.
  - e.g: ps, free, ...

```
Terminal
File Edit View Search Terminal Help

nsh> ps
PID  PPID  PRI  SCHED  TYPE  NP  STATE  STACK  USED  NAME
0    0     0    FIFO  KTHREAD  READY  656  35%  Idle Task()
1    0    192   FIFO  KTHREAD  WAITSEM 1144  35%  hpwork()
2    0     50   FIFO  KTHREAD  WAITSEM 576  42%  lpwork()
3    0    128   FIFO  KTHREAD  WAITSEM 1784  12%  wlock()
4    0    100   FIFO  TASK  WAITSIG 848  72%  init()
6    1    128   FIFO  KTHREAD  WAITSEM 1800  11%  scsid()
11   4    100   FIFO  TASK  WAITSEM 544  78%  /system/bin/system_serv()
14   1    100   FIFO  TASK  WAITSEM 608  39%  adbd()
17   4    100   FIFO  PTHREAD  WAITSEM 1496  63%  system_server(20a1d00)
18   4    100   FIFO  PTHREAD  WAITSEM 1648  59%  system_server(20a2e80)
23  11    100   FIFO  TASK  WAITSEM 1064  46%  /system/bin/headless_re(20ab778)
24  11    100   FIFO  PTHREAD  WAITSEM 496  51%  dispatch_headless_rec_r(20ab67c)
25  11    100   FIFO  PTHREAD  WAITSEM 1264  38%  headless_rec_remote_app(20adc00)
26  11    100   FIFO  PTHREAD  WAITSEM 1264  38%  headless_rec_remote_app(20ae5a0)
27   4    100   FIFO  PTHREAD  WAITSEM 2896   5%  <pthread>(2014bec)
28   4    100   FIFO  PTHREAD  WAITSEM 2800   8%  <pthread>(201cb10)
30   4    100   FIFO  PTHREAD  WAITSEM 1376  55%  <pthread>(201cafc)
31   4    100   FIFO  PTHREAD  WAITSEM 1456  52%  <pthread>(201cb38)
32   4    100   FIFO  PTHREAD  WAITSEM 2536  17%  <pthread>(201cb24)
37  11    100   FIFO  TASK  WAITSEM 1704  57%  /system/devel/ext1/home(20a7708)
38  11    100   FIFO  PTHREAD  WAITSEM 496  51%  dispatch_HomeApp(20a75d4)
39  11    100   FIFO  PTHREAD  WAITSEM 3312  19%  HomeApp(21036e0)
40  11    100   FIFO  PTHREAD  WAITSEM 3592  12%  HomeApp(2104880)
41   4    100   FIFO  TASK  RUNNING 1448  27%  /system/xbin/toolbox(20bc145)

nsh> free
MEM      : total      used      free      largest  ref(s)
[GLOBAL ]: 498800    256640    242160    225856    1283
[USREXT1]: 262144    36352    225792    225120    286
[USREXT4]: 131216     16      131200    131200     1
[USREXT5]: 0           0         0         0         0
EVFATSTACK: 4096     3616     480
INTSTACK : 2048     300      1748

nsh>
```

\* ELF = Executable and Linking Format

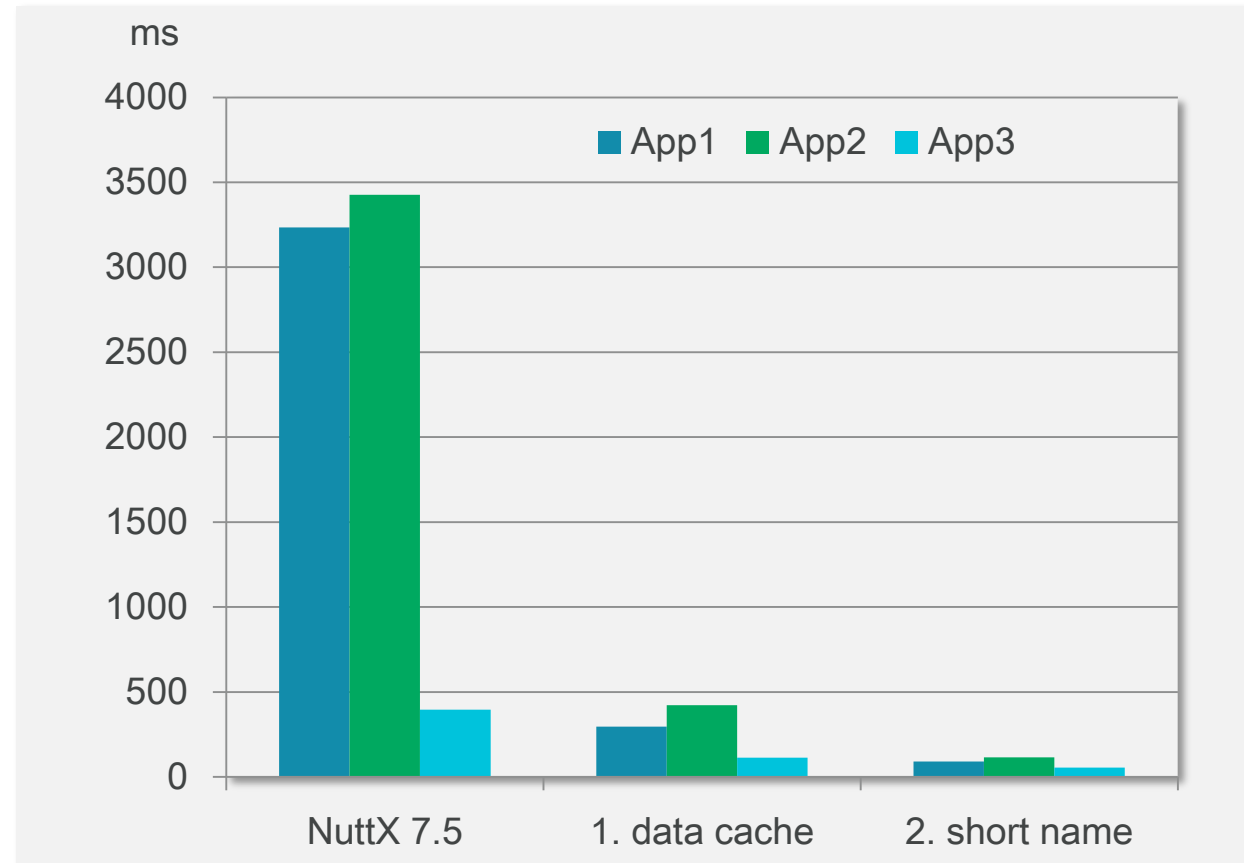
# Fast ELF loading

## 1. Section data cache

- Allocate a big heap to hold tables to reduce eMMC access.
- Use unused SRAM areas, if possible
  - e.g. DSP program & work area

## 2. Symbol name replacement

- Shorten symbols by hashing their names
- Sort A-Z and do binary-search in find-by-name
- Need to modify the build system

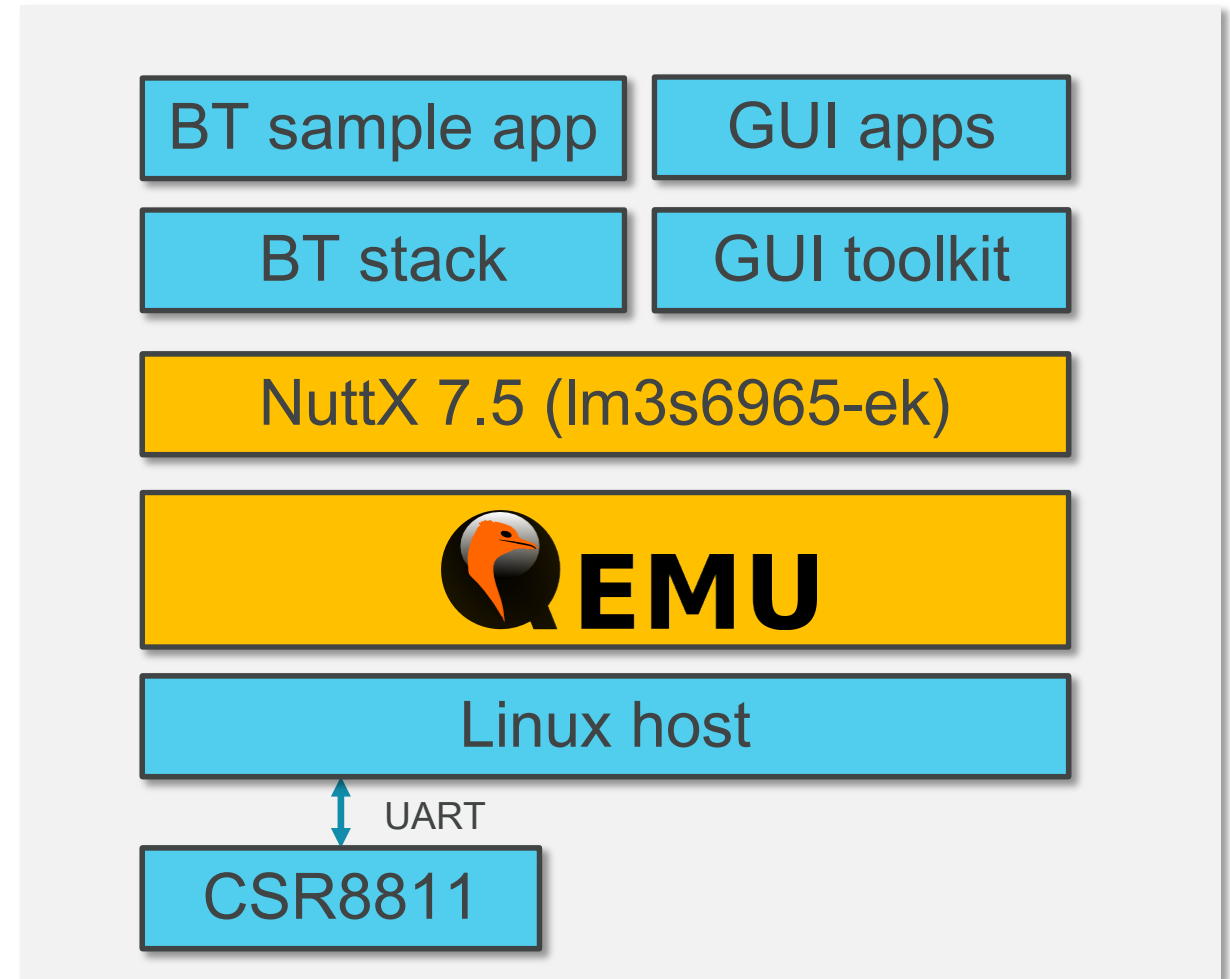


```
└─ {"pthread_condattr_setclock", &pthread_condattr_setclock},  
└─ {"a895afc", &pthread_condattr_setclock},
```

# Developing with QEMU\*

SONY

- Motivation
  - To port the Bluetooth stack
  - To port in-house GUI toolkit
  - To develop applications
- Implementation
  - Start with 1.4.0
  - Use TI Stellaris for QEMU hardware
  - Use Im3s6965-ek for NuttX
  - Increase SRAM size to 4MB
  - Fix SD driver
  - Fix NVIC issue



\*QEMU is open source CPU emulator



- Motivation

- Improve productivity
- Performance benefits

- Features

- `auto` keyword
  - the compiler determines the type
- Lambda expression to define function objects
- New smart pointer
  - to avoid memory leaks
  - introduced `std::unique_ptr<>` and `std::shared_ptr<>`
- Move semantics to optimize copying
  - introduced move constructor and assignment
  - introduced `std::move()`
- `override`, `final`, `nullptr`, `constexpr` ...

```
vector<vector<MyType>>::const_iterator it = v.begin();
```

can be replaced with


```
auto it = v.cbegin();
```

```
bool is_fuel_level_safe()
{
    return all_of(_tanks.begin(), _tanks.end(),
        [this](Tank& t) { return t.fuel_level() > _min_fuel_level; });
}
```

From <http://cpprocks.com/9-reasons-to-start-using-c11/>

# C++ Standard library

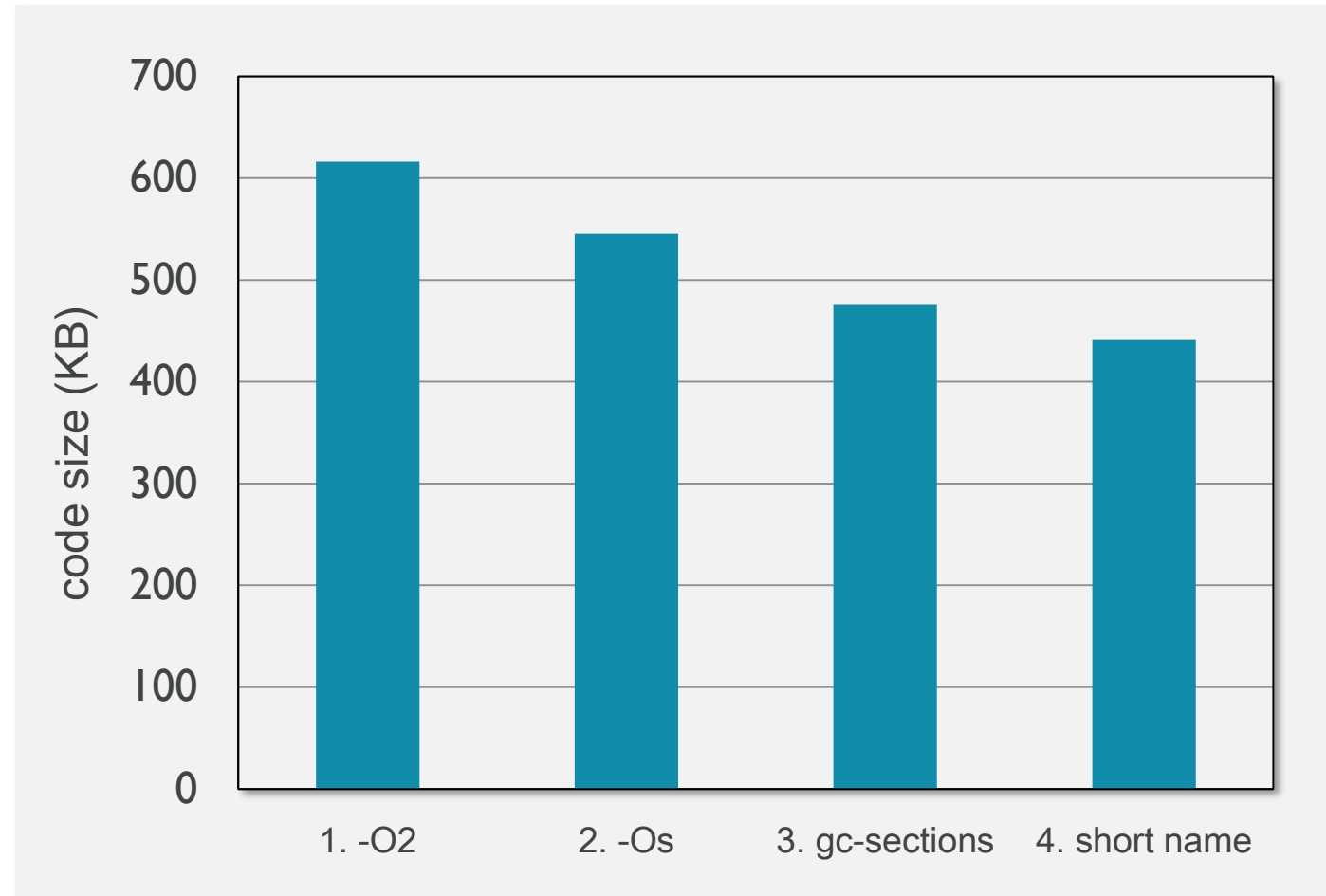
SONY

	libc++	libstdcxx	libstdc++	STLPort
Maintained by	 LLVM COMPILER INFRASTRUCTURE	APACHE	GNU	STLPort
C++11 support	Fully supported	Not supported	Fully supported	Not supported
License	MIT and UIUC (BSD-like)	Apache	GPLv3 (mainline) GPLv2 (ver 4.2)	Boris Fomitchev
Others	LLVM and Clang supported Newer codebase and easier to port	4.2.1 released in 2008/05	Tightly integrated with g++	5.2.1 released in 2008/10

From <http://libcxx.llvm.org/>

# Code size reduction

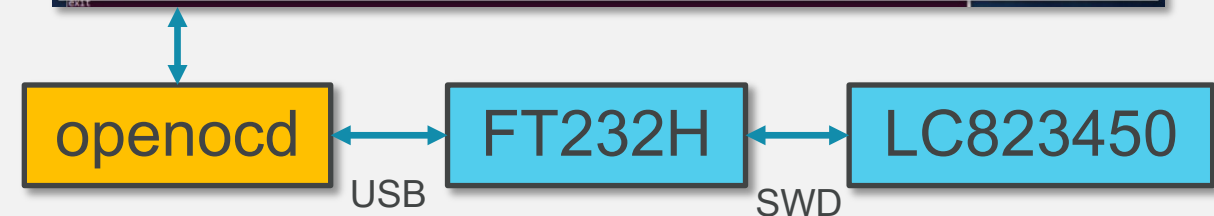
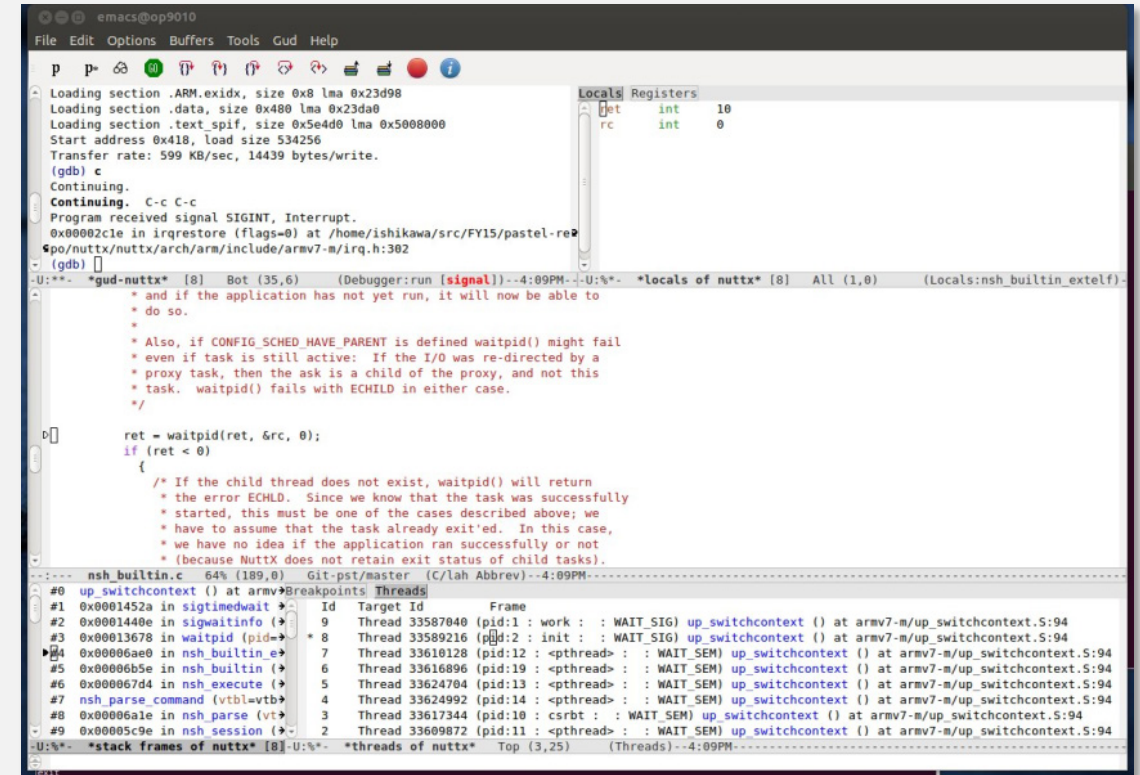
- Example
  - kernel and static libraries
- Approaches
  1. Started with '-O2'
  2. Plus Compile with '-Os'
  3. Plus GC of unused sections at link
  4. Plus Symbol name replacement



# Debugging with apps (1/2)

emacs + arm-none-eabi-gdb-7.6

- openocd supports some OSes
  - Linux, FreeRTOS, ChibiOS, ...
  - The feature is very useful to debug deadlocks
  - Unfortunately NuttX is not supported
- Implementation\*
  - Similar to other RTOSes (e.g. ChibiOS)
  - Prepare symbol list to look up
    - i.e. g\_readytorun, g\_tasklisttable, ...
  - Implement update\_threads callback
  - Fix memory corruption in rtos.c



\*The code is now available on <https://github.com/sony/openocd-nuttX>

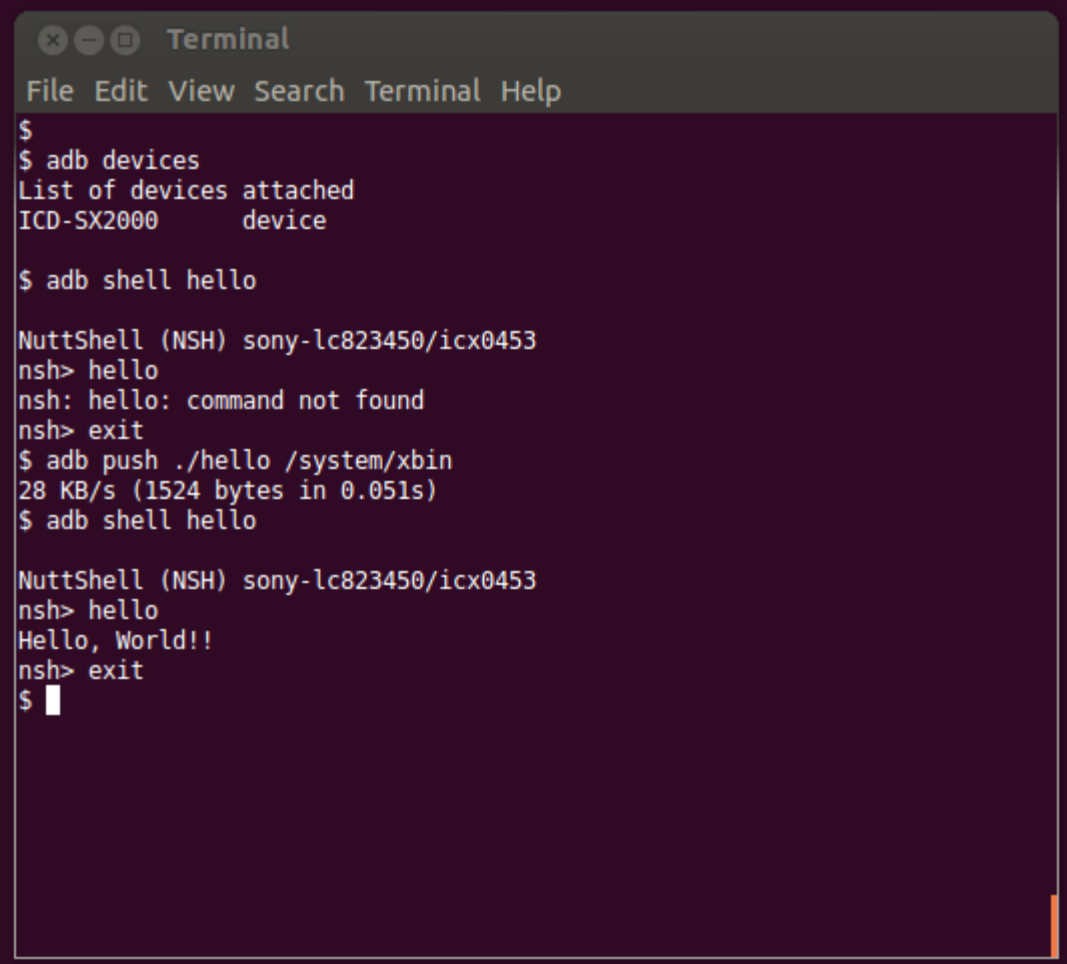
# Debugging with apps (2/2)

- Typical scenario
  - Crash occurs when testing
  - Crash logs are saved in RAM
  - Reboot by WDT
  - Save the logs to a file when booting
  - Pull the log with adb
  - Analyze the log with debug symbols

```
Terminal
File Edit View Search Terminal Help
[36.280] Assertion failed: task: system_server
[36.280] sp: 020f6800
[36.280] IRQ stack:
[36.280] base: 020060b0
[36.280] size: 00000800
[36.280] used: 00000134
[36.280] User stack:
[36.280] base: 020f6b08
[36.280] size: 00000ffc
[36.280] used: 00000a24
[36.280] DLTASKS
[36.280] name = /system/bin/system_server, baseaddr = 0x20c0010
[36.280] name = /system/lib/librecmediarecorder.so, baseaddr = 0x20a4e30
[36.280] name = /system/lib/libqdbm.so, baseaddr = 0x20f0bc0
[36.280] name = /system/devel/ext1/usb_connect_app, baseaddr = 0x2100010
[36.280] == stack trace ==: sp = 0x20f67f0, base = 0x20f6b08
[36.280] addr = 0x0204ef08 (nutttx + 0x204ef08)
[36.280] addr = 0x02048a64 (nutttx + 0x2048a64)
[36.280] addr = 0x020f1390 (libqdbm.so + 0x7d0)
[36.280] addr = 0x020d49a2 (system_server + 0x14992)
[36.280] addr = 0x020d4956 (system_server + 0x14946)
[36.280] addr = 0x020d4974 (system_server + 0x14964)
[36.280] addr = 0x020d443a (system_server + 0x1442a)
[36.280] addr = 0x020cdbac (system_server + 0xdb9c)
0x0204ef08 in up_assert at /home/hudson/workspace/workspace/nmobile10-release-ICX0455/TARGET_BUILD_TYPE/release/TARGET_PRODUCT/ix0455/label/Ubuntu12.04-FY15-ICX0455/out/target/product/ix0455/nutttx/out/sony-lc823450/ix0455/nutttx/arch/arm/src/armv7-m/up_assert.c:556
0x02048a64 in mm_free at /home/hudson/workspace/workspace/nmobile10-release-ICX0455/TARGET_BUILD_TYPE/release/TARGET_PRODUCT/ix0455/label/Ubuntu12.04-FY15-ICX0455/out/target/product/ix0455/nutttx/out/sony-lc823450/ix0455/nutttx/mm/mm_heap/mm_free.c:121 (discriminator 1)
0x020f1390 in _qdbm_munmap at /home/hudson/workspace/workspace/nmobile10-release-ICX0455/TARGET_BUILD_TYPE/release/TARGET_PRODUCT/ix0455/label/Ubuntu12.04-FY15-ICX0455/external/qdbm/myconf.c:288
0x020d49a2 in pst::recscanner::RecStoreProxy::CloseStore() at /home/hudson/workspace/workspace/nmobile10-release-ICX0455/TARGET_BUILD_TYPE/release/TARGET_PRODUCT/ix0455/label/Ubuntu12.04-FY15-ICX0455/framework/mediascanner/src/recscanner/RecStoreProxy.cc:15
0x020d4956 in ~StoreClosure at /home/hudson/workspace/workspace/nmobile10-release-ICX0455/TARGET_BUILD_TYPE/release/TARGET_PRODUCT/ix0455/label/Ubuntu12.04-FY15-ICX0455/framework/mediascanner/src/recscanner/RecScannerImpl.cc:113 (discriminator 1)
0x020d4974 in pst::recscanner::RecScannerImpl::Scan(pst::recscanner::ScanParams const&) at /home/hudson/workspace/workspace/nmobile10-release-ICX0455/TARGET_BUILD_TYPE/release/TARGET_PRODUCT/ix0455/label/Ubuntu12.04-FY15-ICX0455/framework/mediascanner/src/recscanner/RecScannerImpl.cc:62
0x020d443a in pst::recscanner::RecScanner::Scan(pst::recscanner::ScanParams const&) at /home/hudson/workspace/workspace/nmobile10-release-ICX0455/TARGET_BUILD_TYPE/release/TARGET_PRODUCT/ix0455/label/Ubuntu12.04-FY15-ICX0455/framework/mediascanner/src/recscanner/RecScanner.cc:20
0x020cdbac in pst::recservice::RecScanHandler::RecEnd(pst::core::IEventManager*, std::__1::vector<std::__1::basic_string<char, std::__1::cha
```

# adb\* support

- Motivation
  - To test the system without proprietary tools
  - To retrieve internal logs
- Features
  - push, pull and shell with a remote execution
  - The feature is disabled at the factory before shipping
- Implementation
  - Start with the NuttX USB serial driver
    - composite version
  - Change the USB descriptors
  - Implement the protocols from scratch



```
Terminal
File Edit View Search Terminal Help
$
$ adb devices
List of devices attached
ICD-SX2000      device

$ adb shell hello

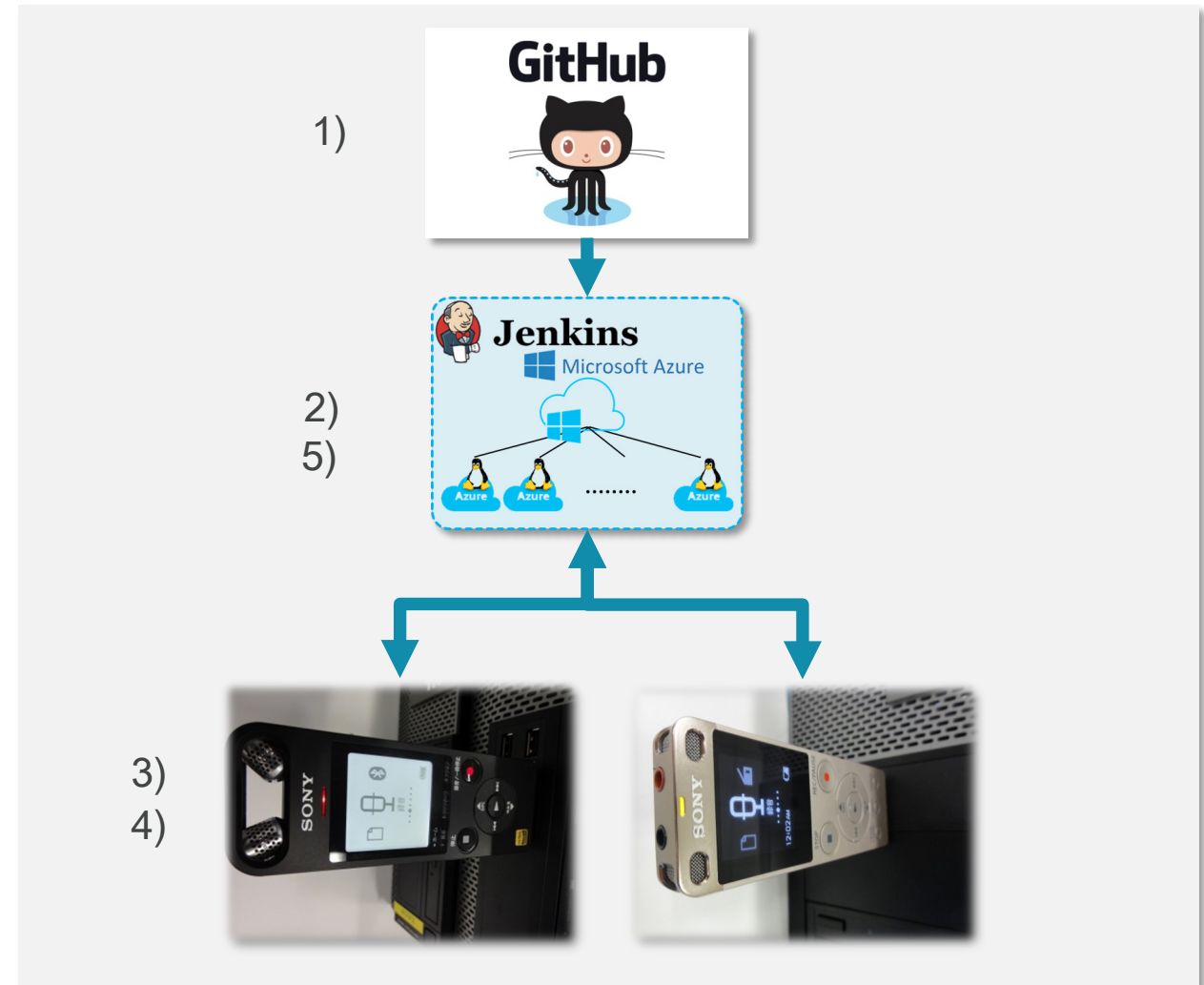
NuttShell (NSH) sony-lc823450/icx0453
nsh> hello
nsh: hello: command not found
nsh> exit
$ adb push ./hello /system/xbin
28 KB/s (1524 bytes in 0.051s)
$ adb shell hello

NuttShell (NSH) sony-lc823450/icx0453
nsh> hello
Hello, World!!
nsh> exit
$
```

\* adb = Android Debug Bridge

# Integration & testing with adb

- For development
  1. Push codes and create a pull request
  2. Build the code
  3. Deploy the software to each product
  4. Test the products with adb
  5. Store the test results with Jenkins
- At factory
  - PCB\* tests are done with adb
  - After all tests pass, adb is disabled



\*PCB = Printed Circuit Board



# Automated Unit-testing with googletest

- Google Test
  - Google's C++ testing framework
  - Port to NuttX and libc++ environment
- Motivation
  - To find bugs early
  - To clarify interfaces between modules
  - To refactor code safely
  - To make sure code works correctly on new target boards
- Executing Test
  - Transfer and execute test cases with adb
  - Faster-cycle of developing and testing

```
File Edit View Search Terminal Help
$ make
$ adb push HelloTest /system/bin
769 KB/s (207060 bytes in 0.262s)
$ adb shell HelloTest

NuttShell (NSH) sony-lc823450/icx1282
nsh> HelloTest
Running main() from gmock_main.cc
[=====] Running 2 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 2 tests from HelloTest
[ RUN      ] HelloTest.get
[          OK ] HelloTest.get (0 ms)
[ RUN      ] HelloTest.set
[          OK ] HelloTest.set (0 ms)
[-----] 2 tests from HelloTest (0 ms total)

[-----] Global test environment tear-down
[=====] 2 tests from 1 test case ran. (1 ms total)
[ PASSED   ] 2 tests.
$
```

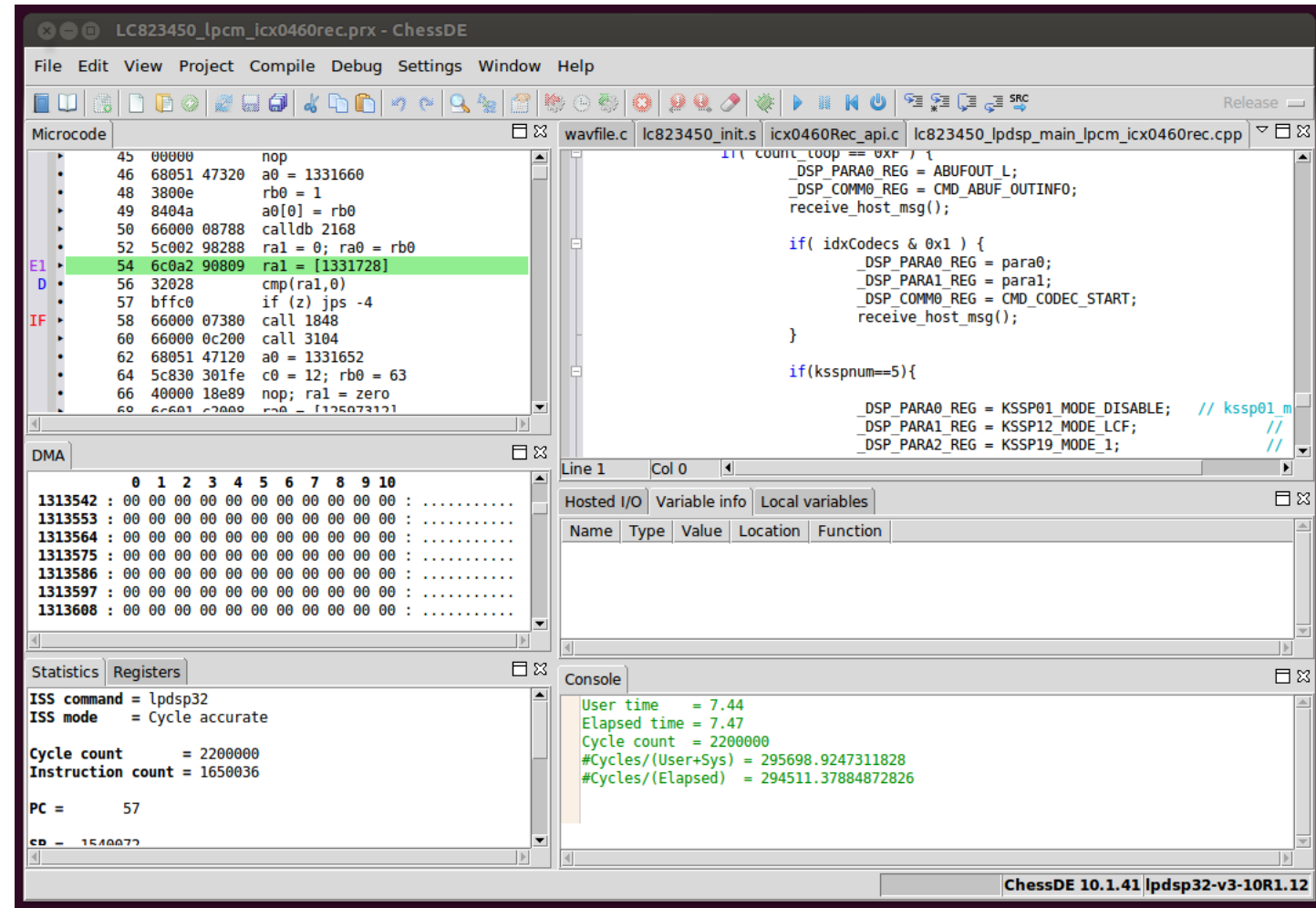


# DSP software development

SONY

- Procedure

- Develop code on the simulator
- Run the sample app on Cortex-M3 and wait for loading DSP code
- Load the DSP code via DSP-ICE then start the DSP
- Continue the app on Cortex-M3



# Demo videos

**SONY**

- Video #1 : adb, fast ELF loading, DVFS
- Video #2 : stress testing tool like Android monkey

# Thank you

ARM and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. APACHE is registered in Australia, Norway, Switzerland, Japan, Brazil, and is pending in other countries, as our house mark symbolizing our high quality community-led, volunteer built software products provided for the public good. Xilinx is a registered trademark of Xilinx, Inc. Linux is a registered trademark of Linus Torvalds. FreeRTOS is a trademark of Real Time Engineers Limited. QEMU is a trademark of Fabrice Bellard. Android is a trademark of Google Inc. POSIX is a registered trademark of the IEEE. BSD is a registered trademark of UUnet Technologies, Inc. The jenkins logo is released under the Creative Commons Attribution-ShareAlike 3.0 Unported License and created by the jenkins project (<https://jenkins.io/>) GITHUB, the GITHUB logo design, OCTOCAT and the OCTOCAT logo design are exclusive trademarks registered in the United States by GitHub, Inc. Microsoft Azure is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries. Stellaris is a registered trademark of Texas Instruments Incorporated. "WALKMAN" and "WALKMAN" logo are registered trademarks of Sony Corporation. SONY is a registered trademark of Sony Corporation.

Copyright 2016 Sony Video & Sound Products Inc.