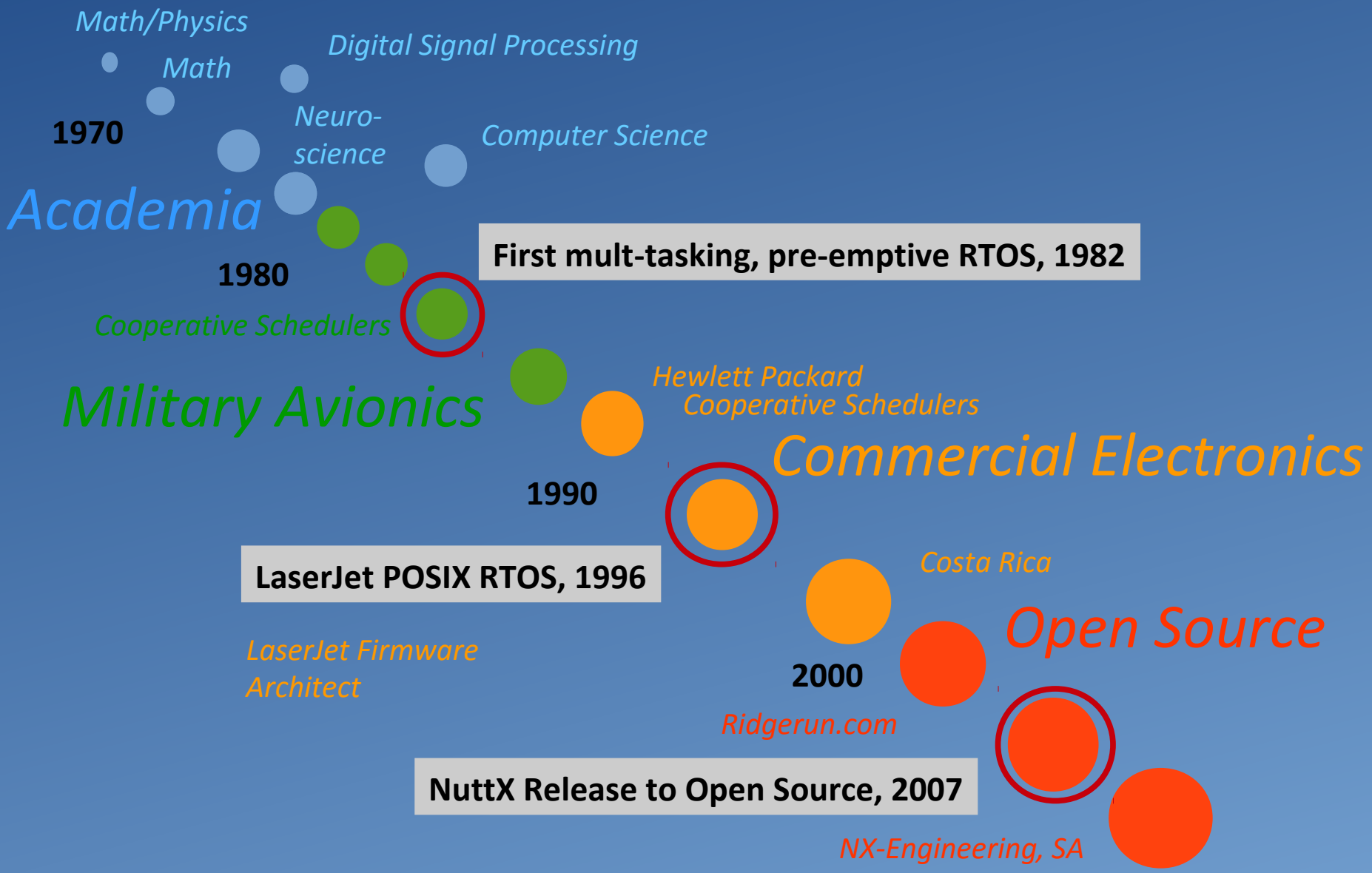
NuttX RTOS

Overview

Gregory Nutt

About Me


Math/Physics
Math
Digital Signal Processing
1970
Neuro-
science
Computer Science
Academia
First mult-tasking, pre-emptive RTOS, 1982
1980
Cooperative Schedulers
Military Avionics
Hewlett Packard
Cooperative Schedulers
Commercial Electronics
1990
LaserJet POSIX RTOS, 1996
Costa Rica
Open Source
LaserJet Firmware
Architect
2000
Ridgerun.com
NuttX Release to Open Source, 2007
NX-Engineering, SA

sourceforge

Search

SOLUTION CENTERS Go Parallel Resources

http://sourceforge.net/projects/nuttx

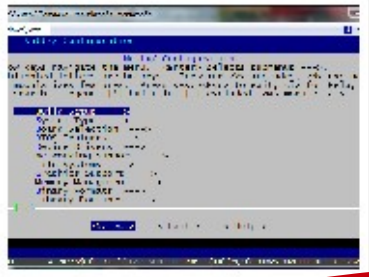
Summary | Files | Reviews | Support | Mailing Lists

★ 5.0 Stars (14)
179 Downloads (This Week)
Last Update: 2015-08-13

Download
nuttx-7.11.tar.gz

Browse All Files

Tweet 3 G+1 7 Like 10



User Ratings

5.0 OUT OF 5 STARS

★★★★★ 14
★★★★ 0
★★★ 0
★★ 0
★ 0

Ease 4 / 5
Features 4 / 5
Design 4 / 5
Support 4 / 5

Description

Nuttx is a real time embedded operating system (RTOS). Its goals are: (1) small footprint usable in deeply embedded, resource constrained environments, (2) fully scalable from tiny (8-bit) to moderate (32-bit), (3) standards compliance, (4) real time, and (5) totally open. Think "Tiny Linux".

NuttX Web Site ›

Categories
Operating System Kernels

License
BSD License

"Nuttx is a ***real time embedded*** operating system (RTOS). Its goals are: (1) small footprint usable in ***deeply embedded*** environments, (2) fully ***scalable*** from tiny (8-bit) to moderate (32-bit), (3) ***standards compliance***, (4) ***real time***, and (5) ***totally open***"

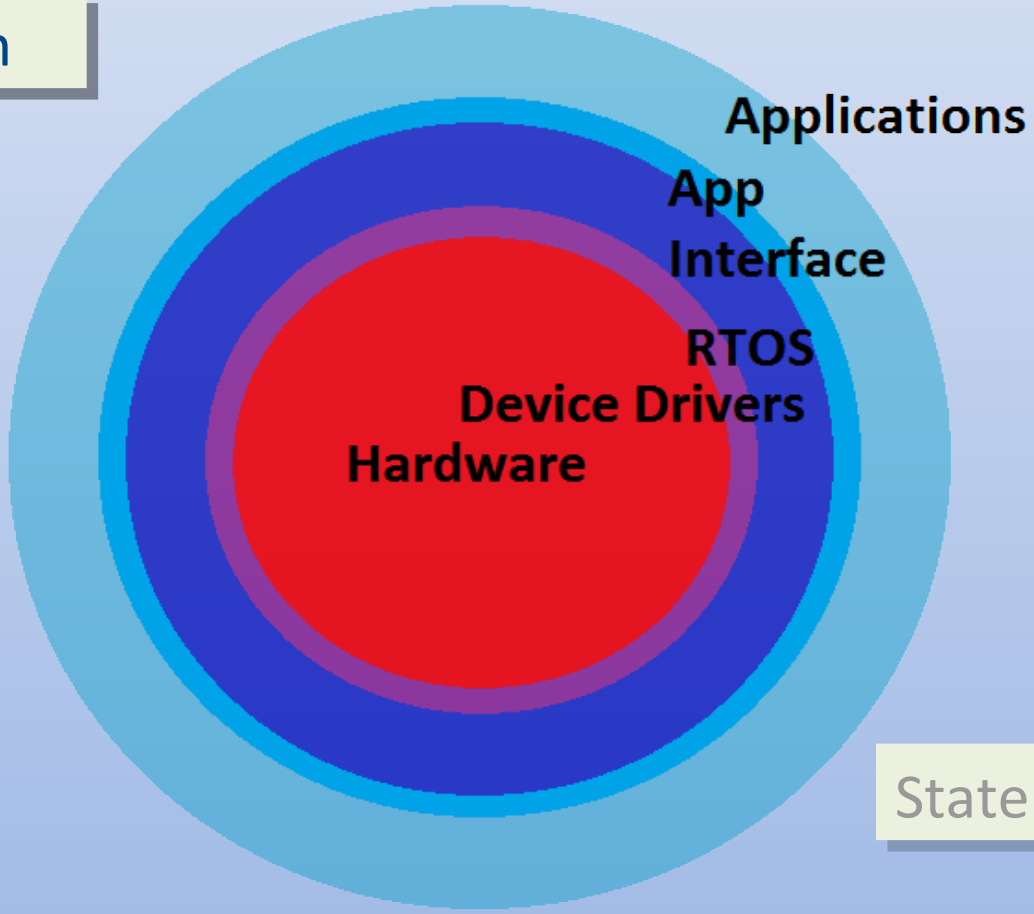
- ***Real time***
  "Deterministic"
- ***Deeply embedded***
  Targeted for application specific MCUs
- ***Scalable***
- ***Standards compliance***
  Think "Tiny Linux"
- ***Totally Open***
  BSD license

RTOS == *"Real Time Operating System"*

MCU == "Micro-controller Unit"

Operating System:
Hardware Abstraction Layer / Portability

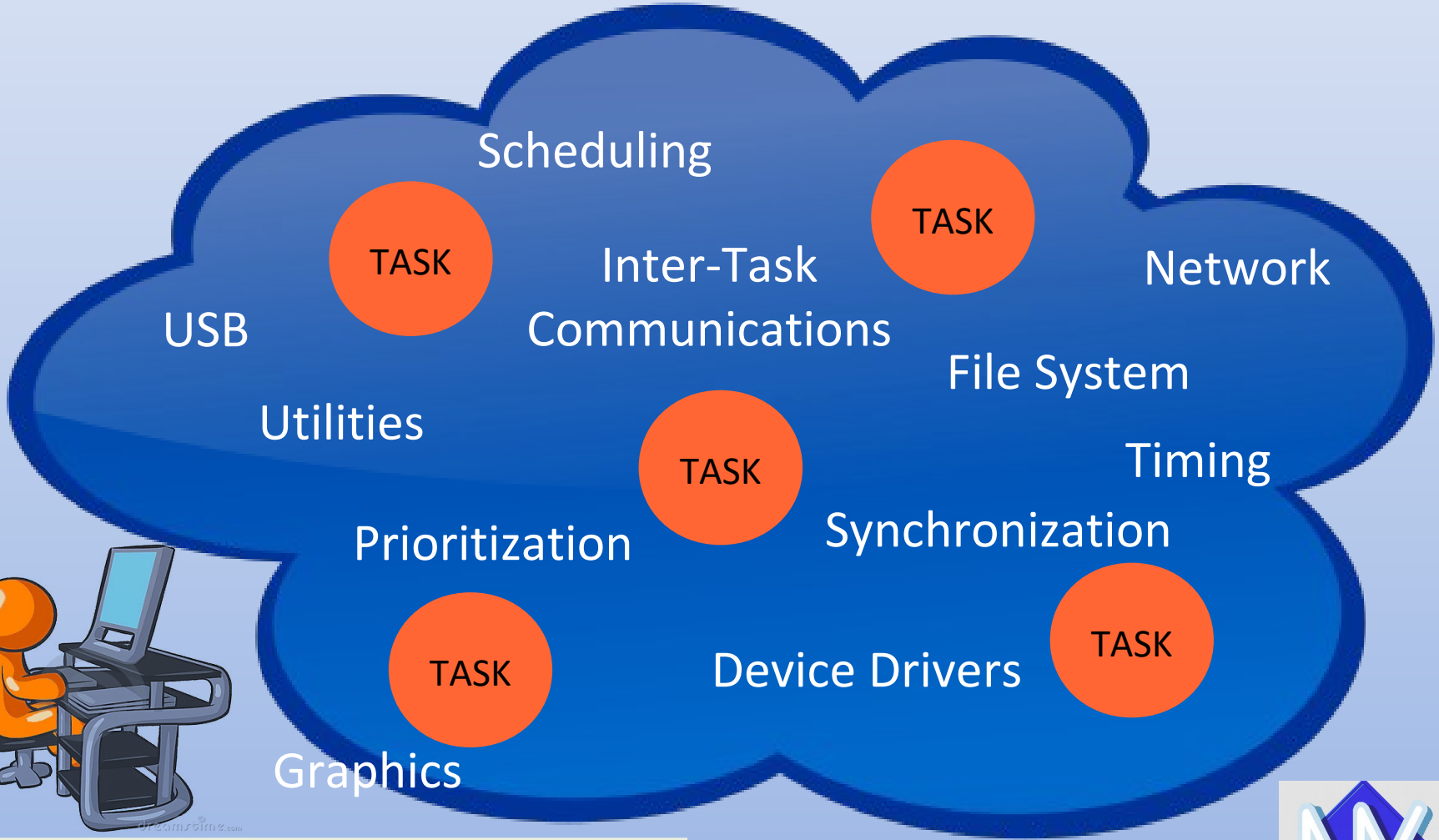
Onion Diagram



The Architect's Point of View

State Diagrams

Operating System:
provides an operating environment for tasks

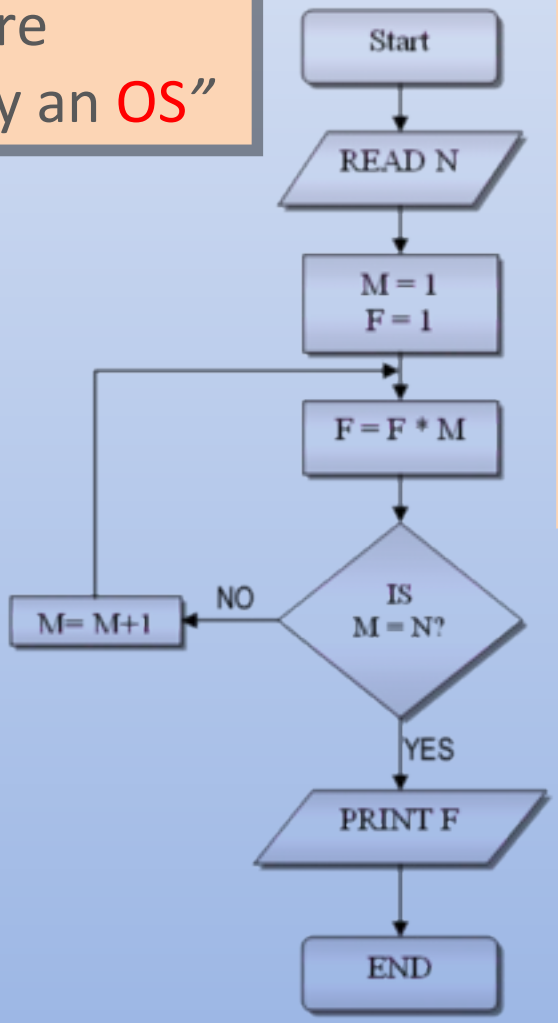


The Programmer's Point of View

Tasks

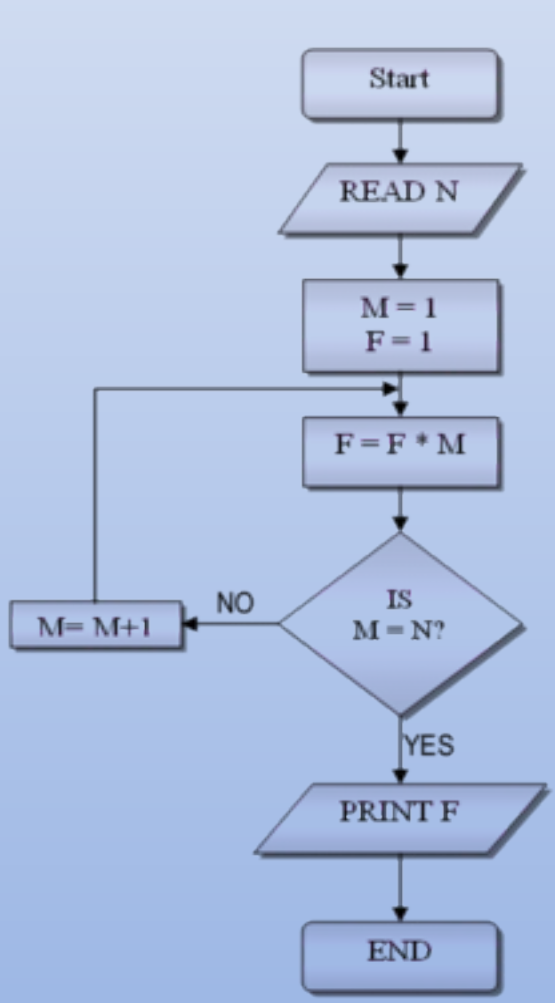
Task = “Unit of software execution managed by an OS”

- ***Processes***
- ***Tasks***
- ***Threads***



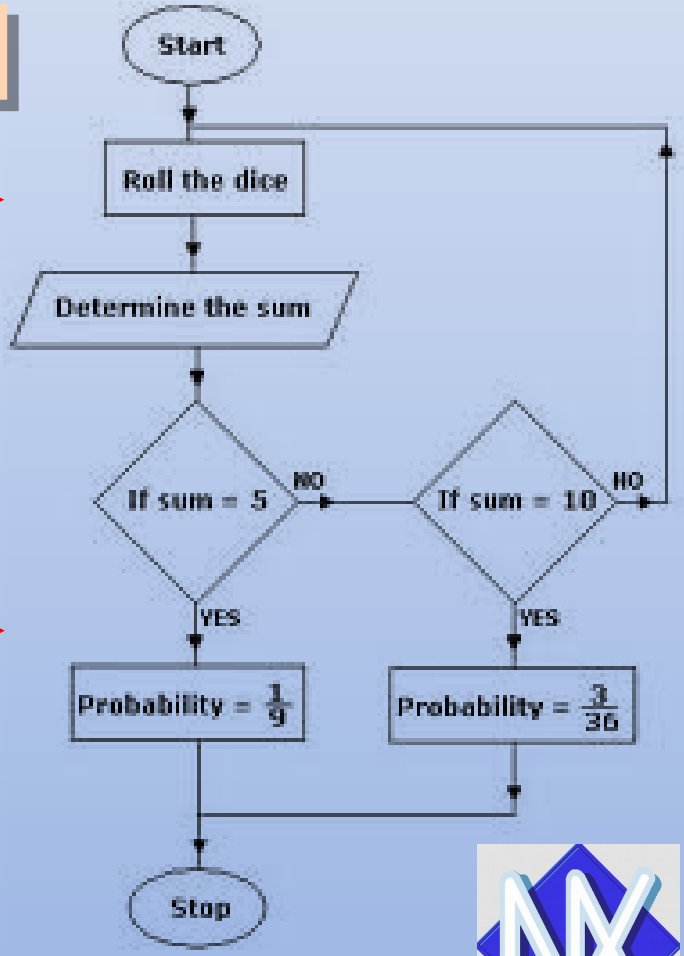
Thread = Sequence of machine instruction executions controlled by *normal* jump and call type instructions. Each thread has its own stack.”

Multi-Tasking and Concurrency

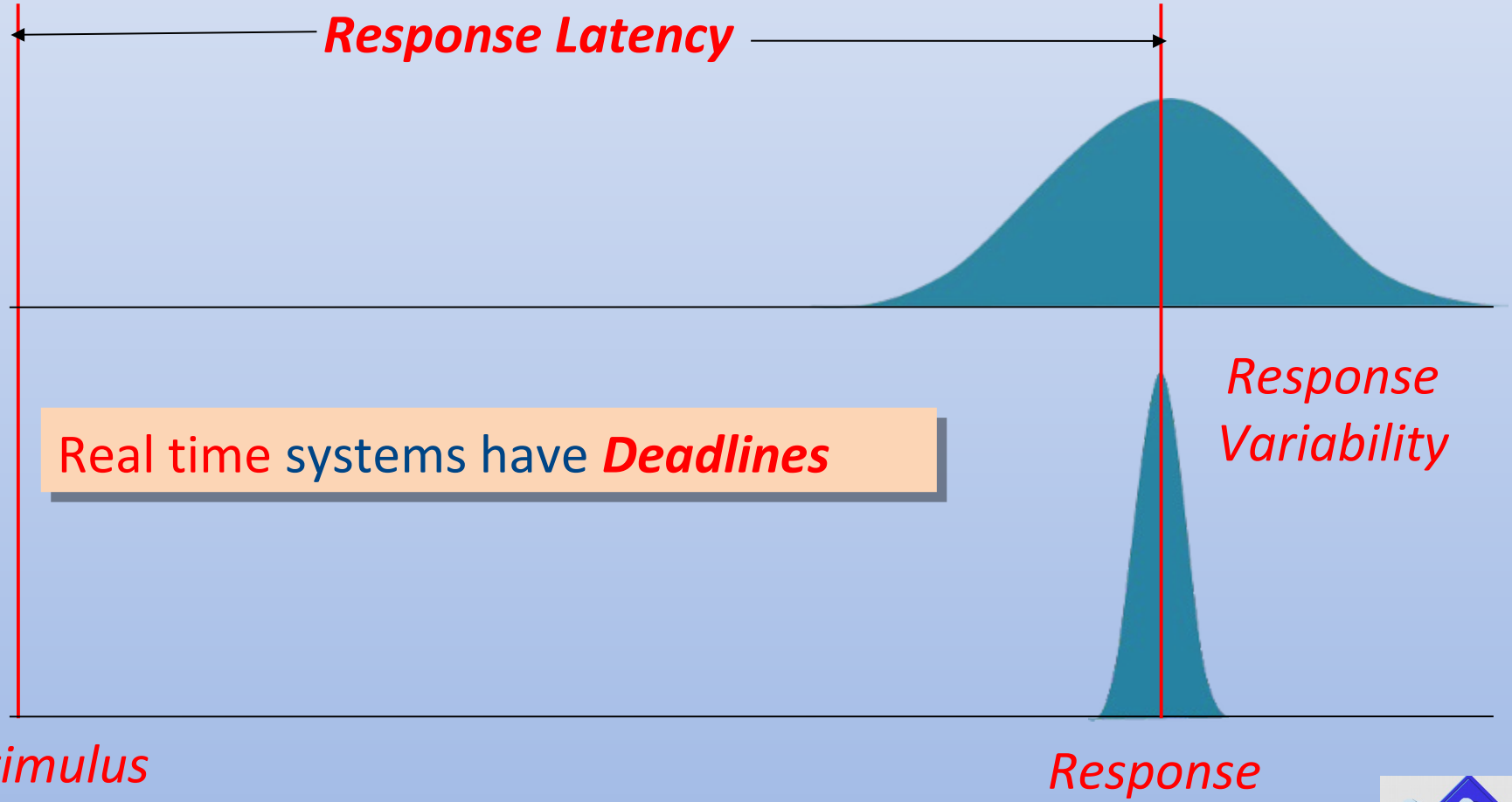


Context Switching

Concurrency



Real Time == Determinant



Real time systems have ***Deadlines***

Real time does not mean “fast”

MCUs vs. Microprocessors

MicroController Unit

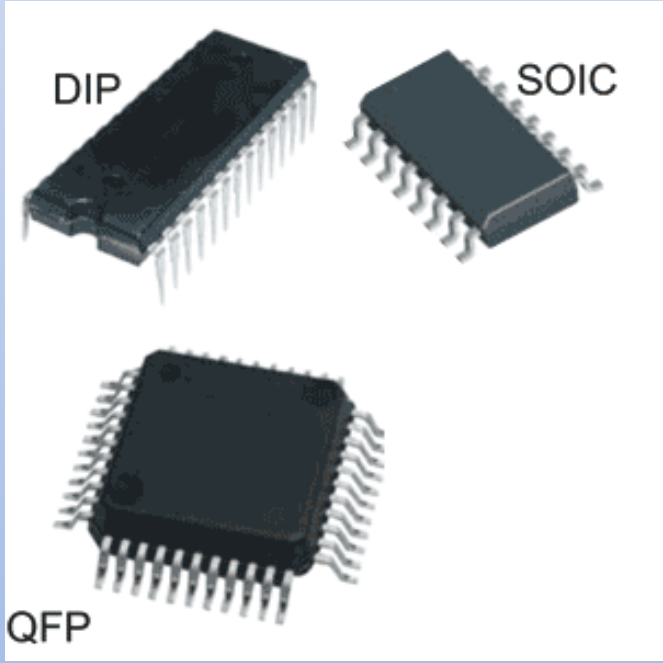
- Applications: Deeply embedded (think cars, refrigerators, MP3 players, etc.)
- Cost: Cheap
- Minimal board support required.
- Lower pin count
- Small internal memories (32KB - 2MB FLASH , 2Kb - 512KB SRAM)
- Slow: 48-100s MHz

Microprocessor

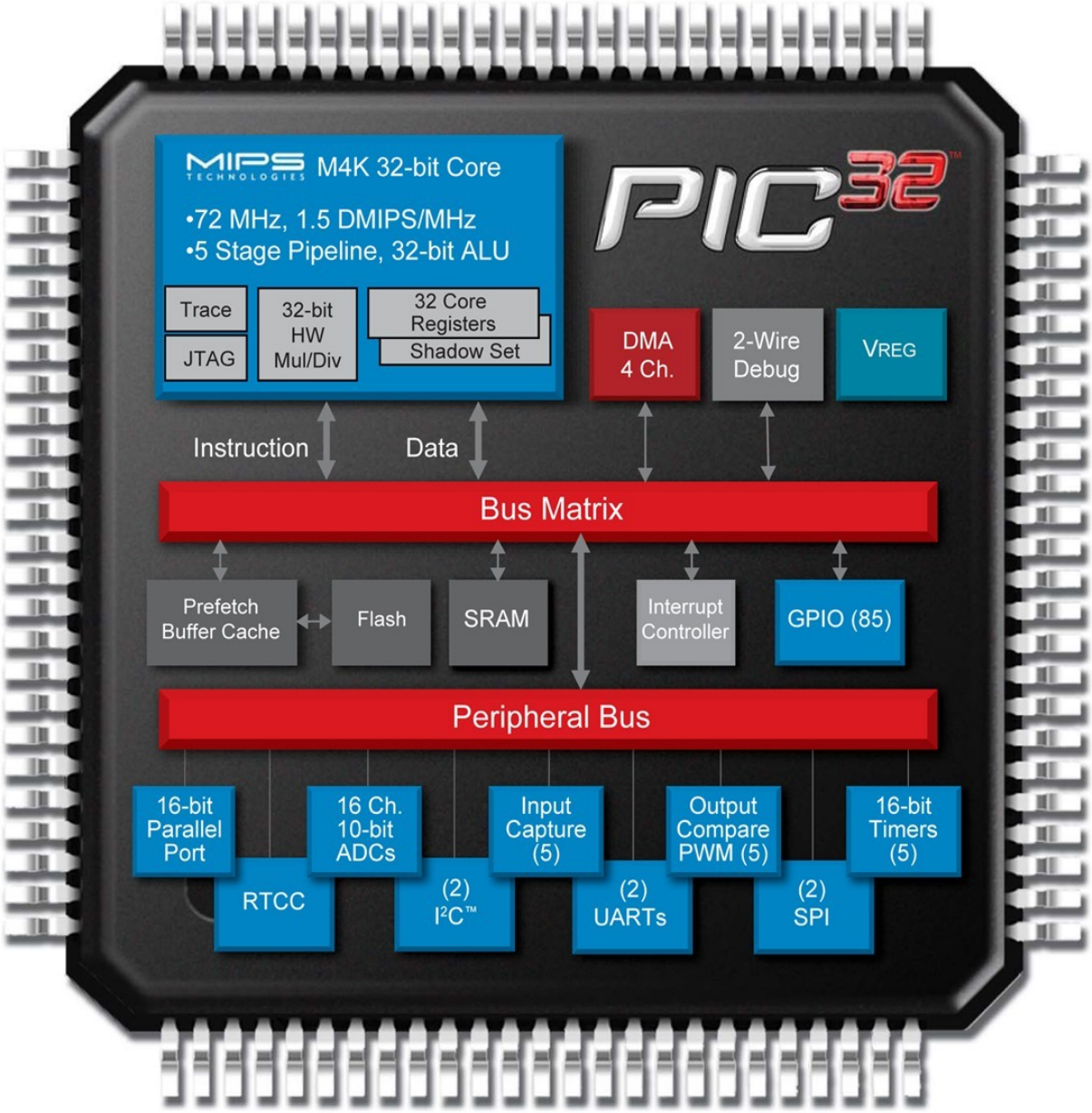
- Applications: Platforms (think smart phones, HDTVs, DVRs, set-top-boxes, etc).
- Cost: Pricey
- Extensive board support required
- Higher pin count
- External memories (Mega- or Gigabytes of SDRAM and FLASH)
- Fast: 100s of MHz - GHz

MCUs vs. Microprocessors

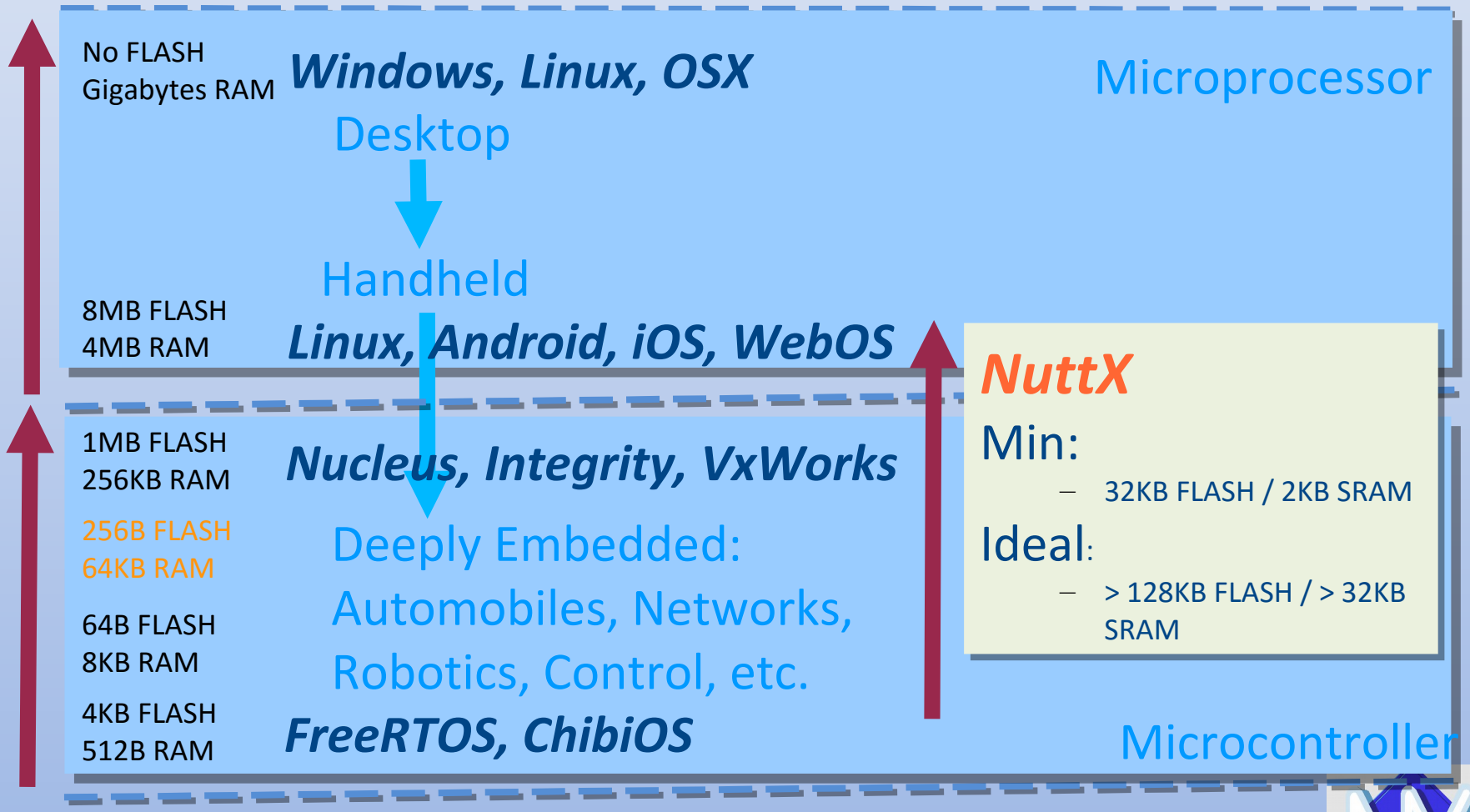
MicroController Unit



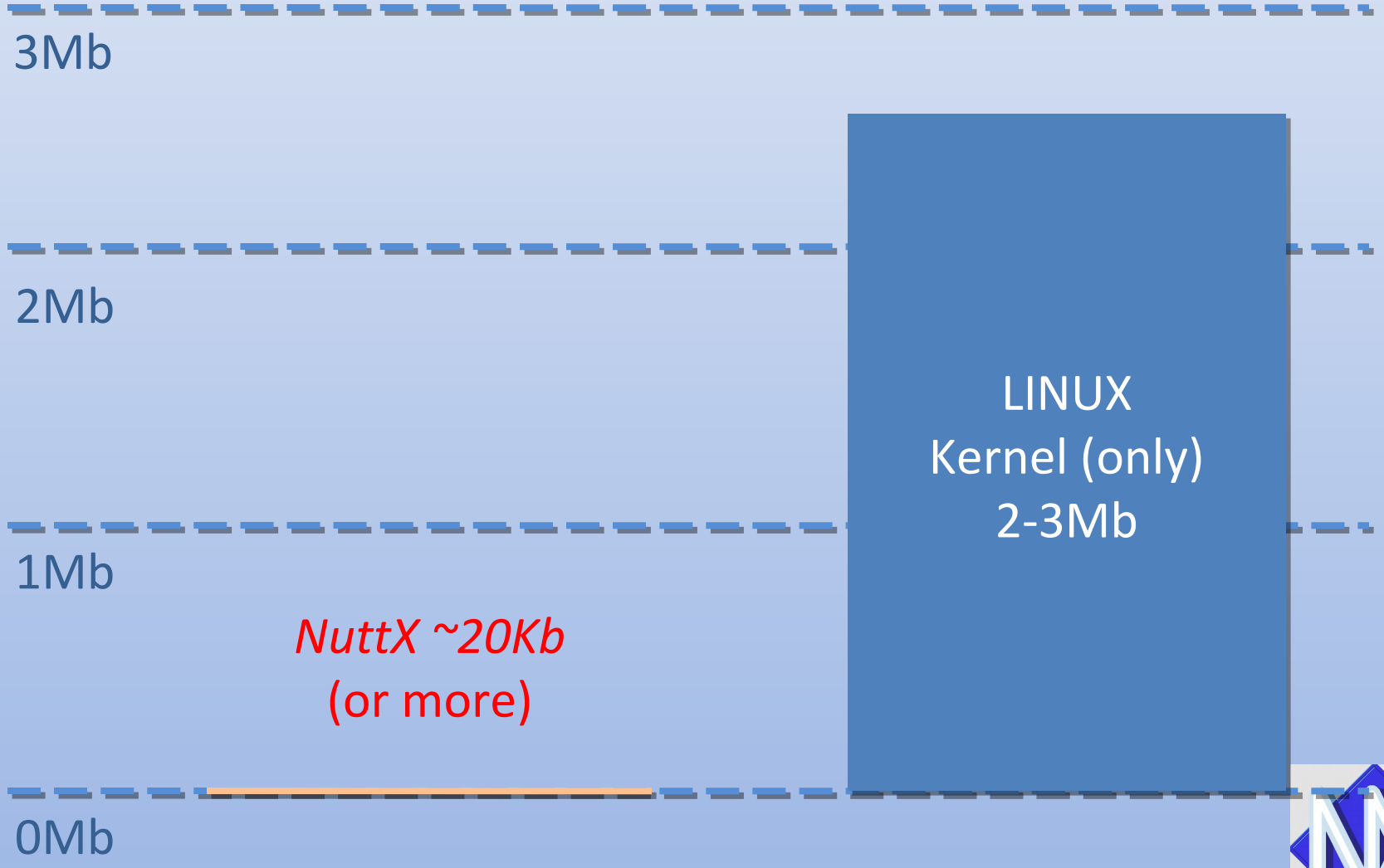
Microprocessor


MIPS TECHNOLOGIES
M4K 32-bit Core
•72 MHz, 1.5 DMIPS/MHz
•5 Stage Pipeline, 32-bit ALU
Trace
JTAG
32-bit HW Mul/Div
32 Core Registers
Shadow Set
PIC32
DMA 4 Ch.
2-Wire Debug
VREG
Instruction
Data
Bus Matrix
Prefetch Buffer Cache
Flash
SRAM
Interrupt Controller
GPIO (85)
Peripheral Bus
16-bit Parallel Port
RTCC
16 Ch. 10-bit ADCs
(2) I²C™
Input Capture (5)
(2) UARTs
Output Compare PWM (5)
(2) SPI
16-bit Timers (5)

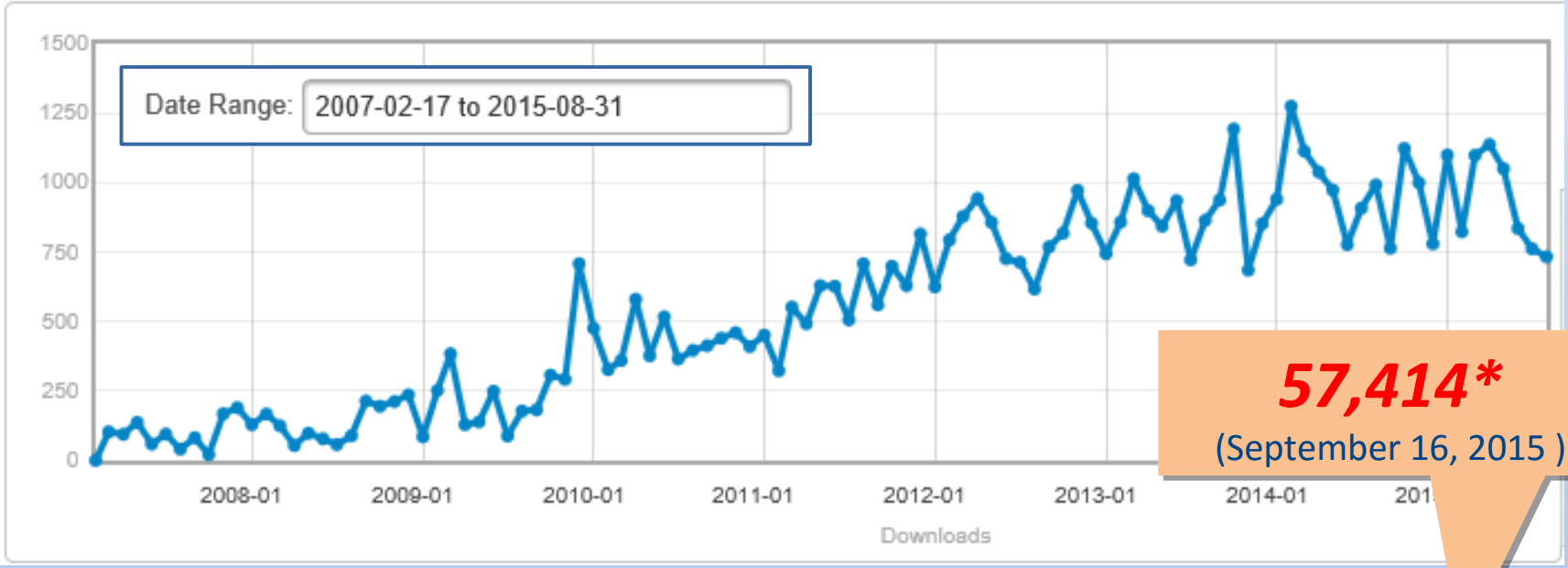
An OS for Every Purpose



Small Footprint



How Many People Use NuttX?



57,414*

(September 16, 2015)

sourceforge ?

Bitbucket

*Excludes main Bitbucket.org downloads, other download sites; excludes direct repository accesses, shadow repositories, etc.

DOWNLOADS

57,414

In the selected date range

TOP COUNTRY *

China

26% of downloaders

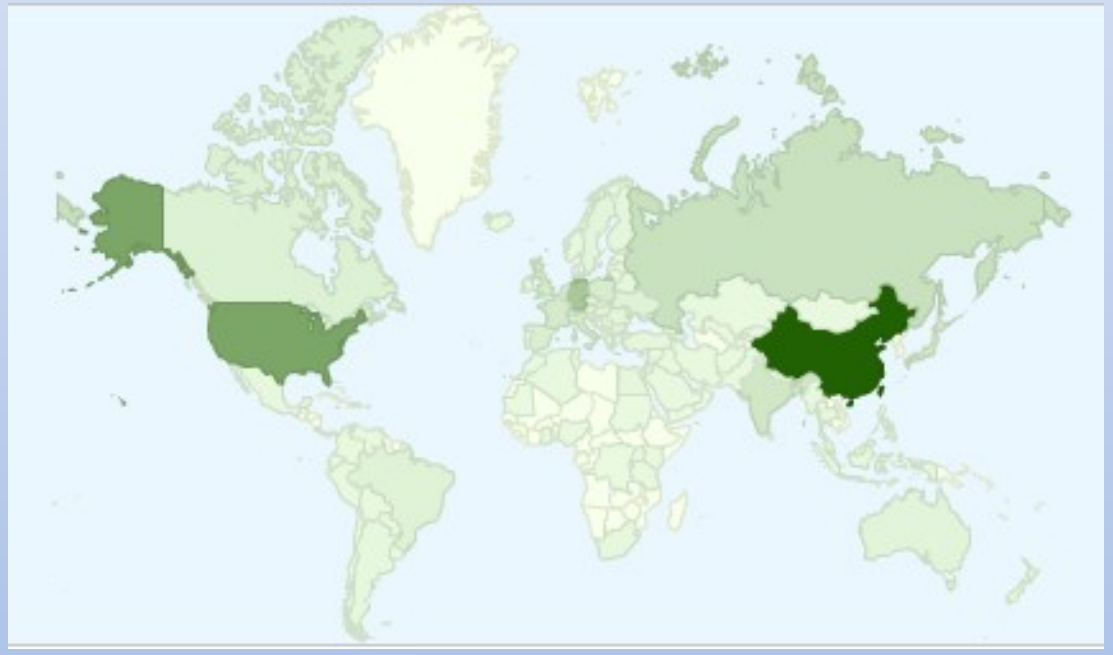
TOP OS *

Windows

63% of downloaders

Who Uses NuttX?

Date Range: 2007-02-17 to 2015-09-16



DOWNLOADS

57,414

In the selected date range

TOP COUNTRY *

China

26% of downloaders

TOP OS *

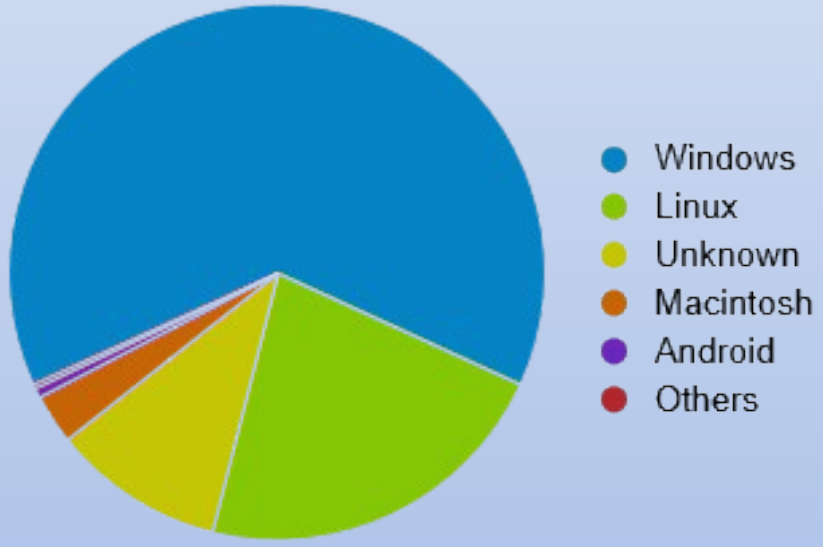
Windows

63% of downloaders

	Country	Android	BSD	Linux	Macintosh	Solaris	Unknown	Windows	Total
1.	China	1%	0%	6%	1%	0%	3%	58%	13,820
2.	United States	0%	0%	18%	5%	0%	14%	33%	7,546
3.	Germany	1%	0%	20%	1%	0%	17%	31%	4,420
4.	Russia	0%	0%	17%	3%	0%	5%	50%	2,176
5.	India	0%	0%	22%	0%	0%	20%	33%	1,666

What Host OS Do They Use?

Date Range: 2007-02-17 to 2015-09-16



DOWNLOADS
57,414
In the selected date range

TOP COUNTRY *
China
26% of downloaders

TOP OS *
Windows
63% of downloaders

Operating System	Downloads
Windows	32,730
Linux	11,268
Unknown	5,326
Macintosh	1,550
Android	301
BSD	122
Solaris	5

	Country	Android	BSD	Linux	Macintosh	Unknown	Windows	Total
1.	China	0%	0%	1%	0%	0%	16%	5,231
2.	United States	0%	0%	4%	2%	0%	7%	2,562
3.	Germany	0%	0%	7%	0%	1%	9%	1,611
14.	Costa Rica	0%	0%	2%	0%	0%	7%	261

NuttX Core Features

- Standards compliant, modular core task management
- Memory management
- File system
- Binary Loader
- C library
- Audio
- Cryptographic
- Drivers
- Networking
- NX Graphics support
- Key Applications: NuttShell (NSH)

Standards Compliant, Modular Core Task Management

- Modular kernel design
- Fully pre-emptible
- Naturally scalable
- Highly configurable
- Easily extensible to new processor architectures, SoC architecture, or board architectures
- FIFO, round-robin, and Sporadic scheduling.
- Realtime, deterministic, with support for priority inheritance
- POSIX/ANSI-like task controls, named message queues, counting semaphores, clocks/timers, signals, pthreads, environment variables, file system.
- VxWorks-like task management and timers.
- Tickless operation

NuttX is a "Tiny Linux"

Standards Compliant, Modular Core Task Management (cont'd)

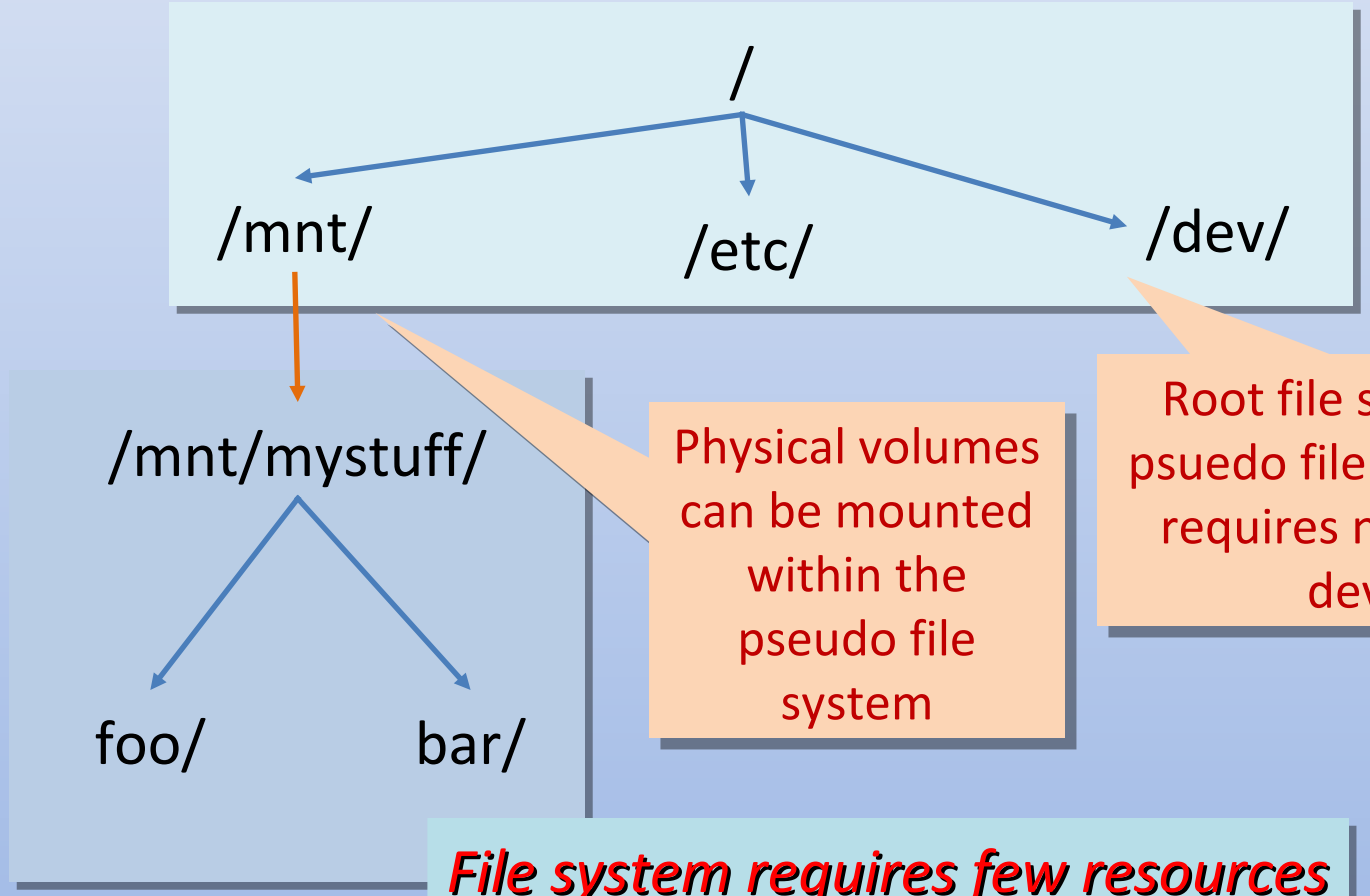
NuttX is a "Tiny Linux"

- System logging
- Extensions to manage pre-emption
- On-demand paging.
- System logging
- May be built either as an open, flat embedded RTOS or as a separately built, secure kernel with a system call interface.
- Flat, embedded build or Linux-like kernel build with "processes"
- Built-in, per-thread CPU load measurements.
- Well documented in the NuttX User Guide.

File system

- Tiny in-memory, root pseudo-file-system
- Virtual file system supports drivers and mountpoints.
- Character and block drivers
- Mount-able volumes. Bind mount point, file system, and block device driver
- Generic system logging (SYSLOG) support.
- FAT12/16/32 file systems with optional FAT long file name support.
- NFS Client. Client side support for a Network File System (NFS, version 3, UDP).
- NXFFS. The tiny NuttX wear-leveling FLASH file system.
- SMART. Another wear-leveling FLASH file system.
- TMPFS. A tiny, dynamic RAM file file system.
- ROMFS file system support
- BINFS and PROCFS pseudo-filesystems.
-

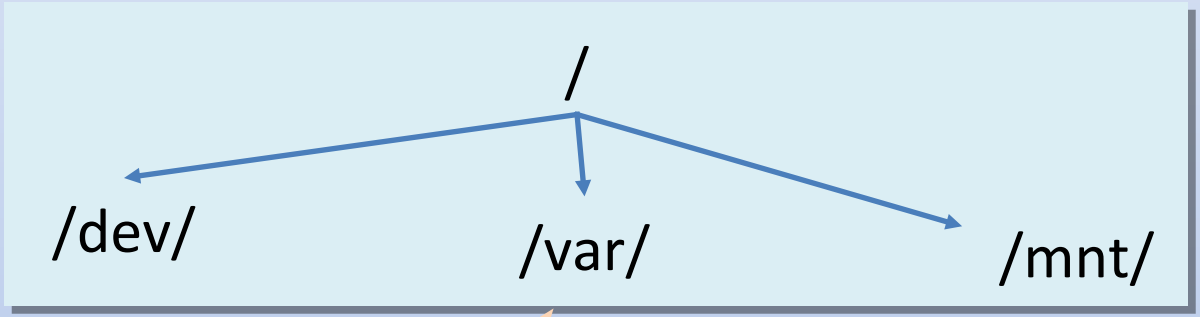
Pseudo Root File System



File system requires few resources

File system does not require media

Special Files



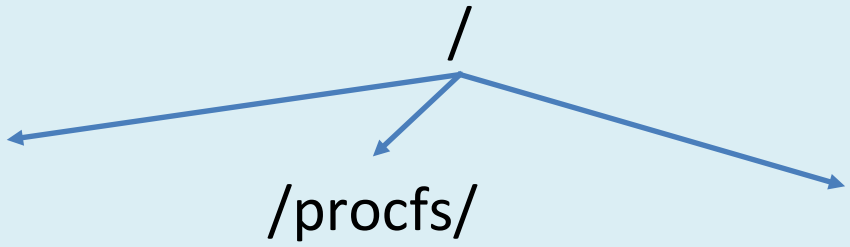
Character drivers
Block drivers
Special Drivers

All OS named resources are maintained in the VFS:

- Named Message Queues
- Named Semaphores

Mountpoints are also special files

procfs



procfs file system can be mounted in the pseudo-file system.

Provides information about internal state of OS, tasks, and resources.

Binary Loader

- A Binary Loader with support for multiple formats.
- Can be easily extended.
- Separately linked ELF modules. Loaded from file system, relocated into RAM.
- Separately linked NXFLAT modules. Execute XiP in ROMFS file system.
- Interpreted P-code
- Built-in applications
- PATH variable support.

C/C++ Libraries

- *libc*
  - Built-in, fully integrated C library
  - Asynchronous I/O
  - Floating point math library
  - NetDB
- libxx. Lightweight, basic C++ support
- *uClibc++*
  - Standard C++ Library (LGPL)
  - iostreams, strings, STL, RTTI, exceptions, etc.
- *libnx*. Graphics Library

Audio / Cryptographic Subsystems

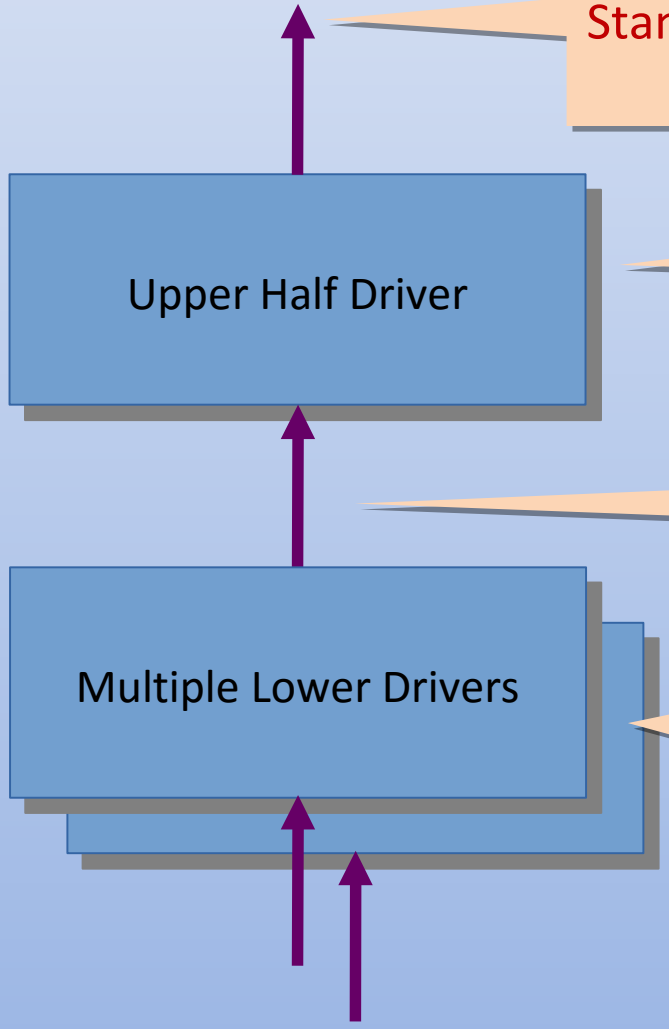
- Audio
  - Stream audio buffer management
  - PCM CODEC
  - Graphic and command line audio players (applications)
- Cryptographic Support

  AES

File system: Device Drivers

- Character and block drivers.
- Special files in VFS.
- Block drivers support file systems.
- Character drivers accessible via standard, POSIX interfaces like a file (open, close, read, write, etc.)
- Loop device – Convert file or character device to a block device
- BCH device – Convert a block device to a character device

Modular Device Driver Design



Standard Character or block driver interface

Common upper half driver.
Hardware Independent!

Common interfaces with multiple instances of lower half driver

Multiple lower half drivers
Bound to upper half via common interface
Provide hardward specific interface

Device Drivers

- Analog: ADC, DAC, Amplifiers
- Audio: CODECs, Audio DAC, I2S
- CAN
- /dev/null, /dev/zero, /dev/random
- FLASH drivers / FLASH File Systems
- Input Devices: Mouse, Touchscreen, Keypad, Buttons, Joysticks
- I/O Expanders
- LCD
- MMC/SD. SPI or SDIO based FLASH card interfaces
- Network Interface devices
- Pipes / FIFOs
- Power: Power management subsystem, batteries

Device Drivers (Cont'd)

- PWM, Quadrature encoder
- RAM Disk
- Sensors: Accelerometers, temperature sensors, many more
- Serial / TTY
- System logging
- Timer drivers: Generic, watchdog, RTC
- USB host / USB device
- Video
- Wireless

Device Drivers: FLASH Support

MTD-inspired interfaces for Memory Technology Devices.

- NAND Support.
- Serial FLASH devices (SPI, QuadSPI)
- EEPROM devices (SPI, I2C)
- On-chip FLASH support

FLASH File Systems

- NXFFS. The NuttX wear-leveling FLASH file system.
- SMART. A new wear-leveling FLASH file system.
- FTL. Simple Flash Translation Layer support any file systems on FLASH.
- Block size conversion layers

USB Host Support

- USB host architecture for USB host controller drivers and device-dependent USB class drivers.
- USB host controller drivers.
- USB class driver registry.
- USB class drivers available for USB hub, USB mass storage, CDC/ACM serial, HID keyboard, HID mouse, RTL8187 Wireless.
- Built-in USB trace functionality.

USB Device Support

- *Gadget*-like architecture for USB device controller drivers and device-dependent USB class drivers.
- USB device controller drivers
- USB class device drivers available for USB serial and for USB mass storage.
- Composite USB devices
- Built-in USB trace functionality.

Networking

- IPv4 or IPv6 or both
- TCP/IP, UDP, ARP, ICMP, ICMPv6, IGMP stacks
- Ethernet, SLIP, PPD link layers
- Routing support
  - Multiple network devices, multiple link layers
  - Routing table
- TUN and local loopback devices
- ICMPv6 autonomous auto-configuration
- Small footprint
- BSD compatible socket layer:
  - Stream, datagram, Raw packet, Unix domain local

Networking (Cont'd)

- DNS Client / NetDB
- PHY Link Status Management
- Networking Utilities for applications
  - DHCPC, DHCPD
  - FTPC, FTPD, TFTPC
  - NTP
  - SMTP
  - Telnet
  - Web client, Web servers
  - XMP RPC
- Network modules (such as the TI CC3000 WLAN module).

Supported MCUs

- **Intel**
  - Linux user-mode simulation, 486SX, QEMU
  - 80c52 (Obsoleted)
- **ARM**
  - **ARM920T**: Freescale i.MX1
  - **ARM926EJS**: TI TMS320DMS320, NXP LPC313x
  - **ARM Cortex A5**: Atmel SAMA5D2, SAMA5D3, SAMA5D4
  - **ARM Cortex A8**: Allwinner A10
  - **ARM Cortex M0**: nuvoTon NUC120, Freescale KL25Z, KL26Z, Atmel SAMD20/21, SAML21
  - **ARM Cortex M3**: ST Micro STM32 F1/F2, TI/Stellaris LM3S, NXP LPC17xx, Atmel SAM3U/3X, SiliconLabs EFM32
  - **ARM Cortex M4**: with/without floating point unit: ST Micro STM32 F3/F4, TI/Stellaris LM4F/TM4C, NXP LPC43xx, Freescale Kinetis K20/K40/60, Atmel SAM4C/4E/4S/4L
  - **ARM Cortex-M7**: Atmel SAMV7, ST Micro STM32 F7

Supported MCUs (Cont'd)

- **Atmel AVR**
  - Atmel 8-bit AVR: AT90USB, Atmega
  - AVR32: AT32UC3Bxx
- **Freescale**
  - HCS12 – MC9S12NExx
- **MIPS**
  - MIPS32 24Kc: PIC32MX
  - MIPS32 M14k: MicroChip PIC32MZ
- **Renesas/Hitachi**
  - Renesas/Hitachi SuperH
  - Renesas M16C/26
- **Zilog**
  - Zilog Z16F ZNeo
  - Zilog eZ80 Acclaim!
  - Zilog Z8Encore!
  - Zilog Z80

Graphics Support

Graphics Drivers

- Framebuffer drivers.
- Graphic LCD drivers for both parallel and SPI LCDs and OLEDs.
- Segment LCD drivers

NX Graphics Subsystem

- A graphics library, windowing system and font support that works with either framebuffer or LCD drivers.

NuttX Widgets

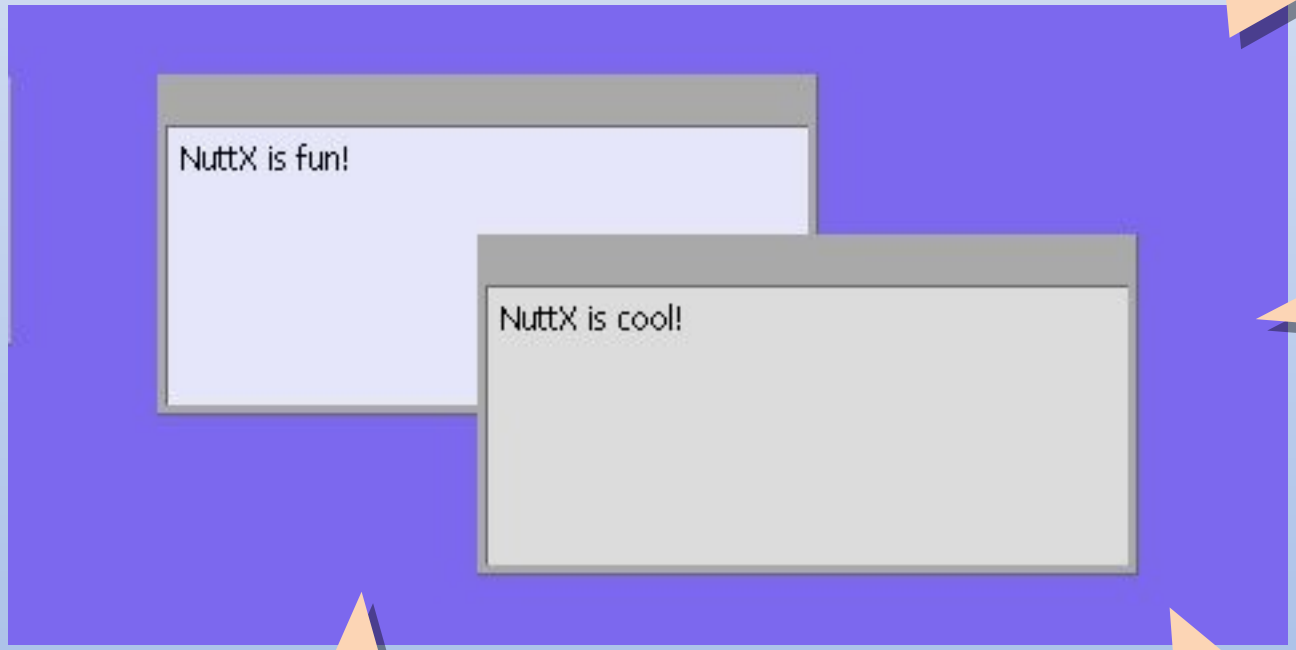
- A graphical user interface
- Written in conservative C++
- Integrates with NX Graphics.

NuttX Window Manager

- Tiny window manager
- Based on the NX Graphics Subsystem and NuttX Widgets.

NX Graphics support

Graphics Server, Multi-threaded

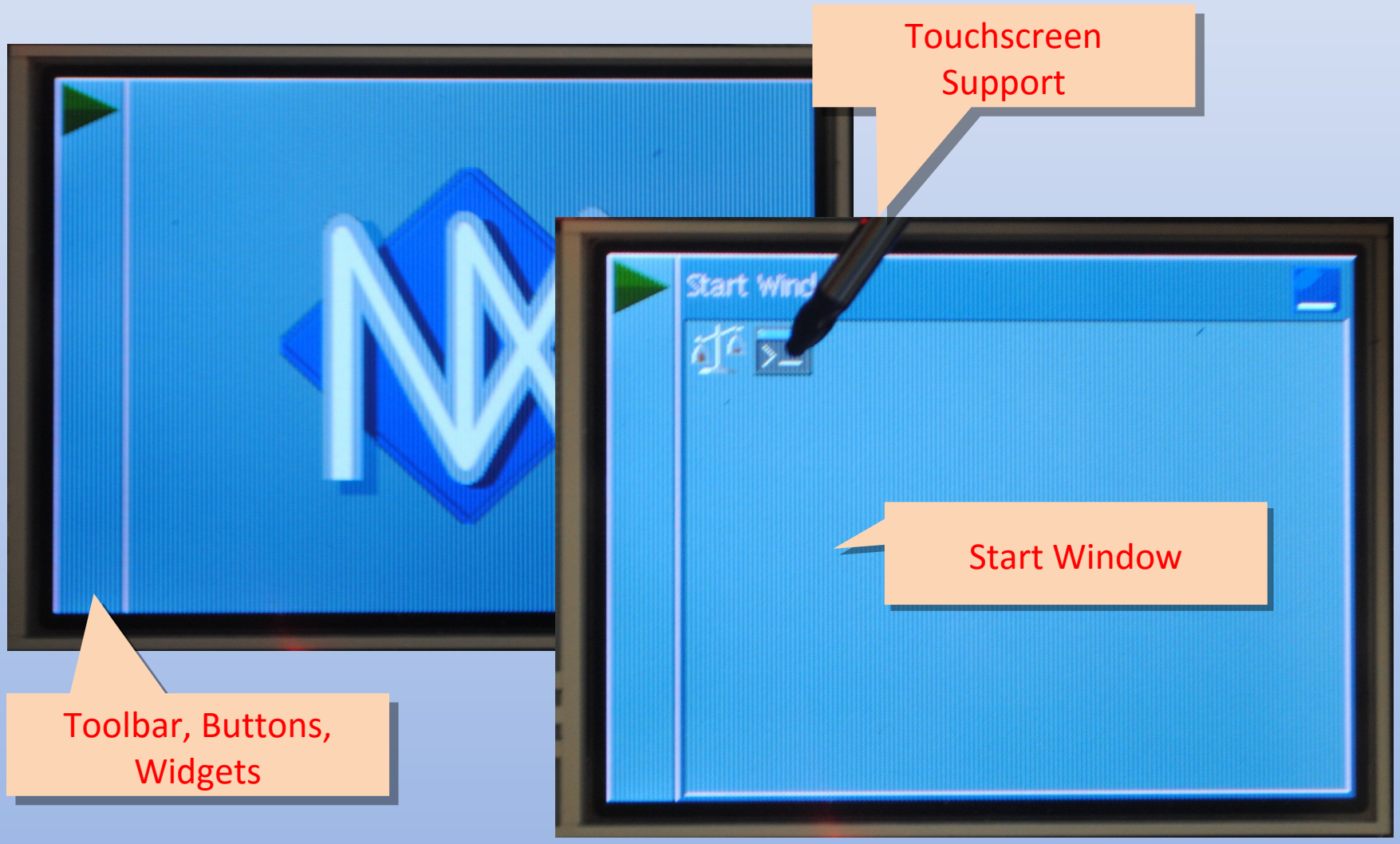


Z-Axis, Raise, Lower, Focus

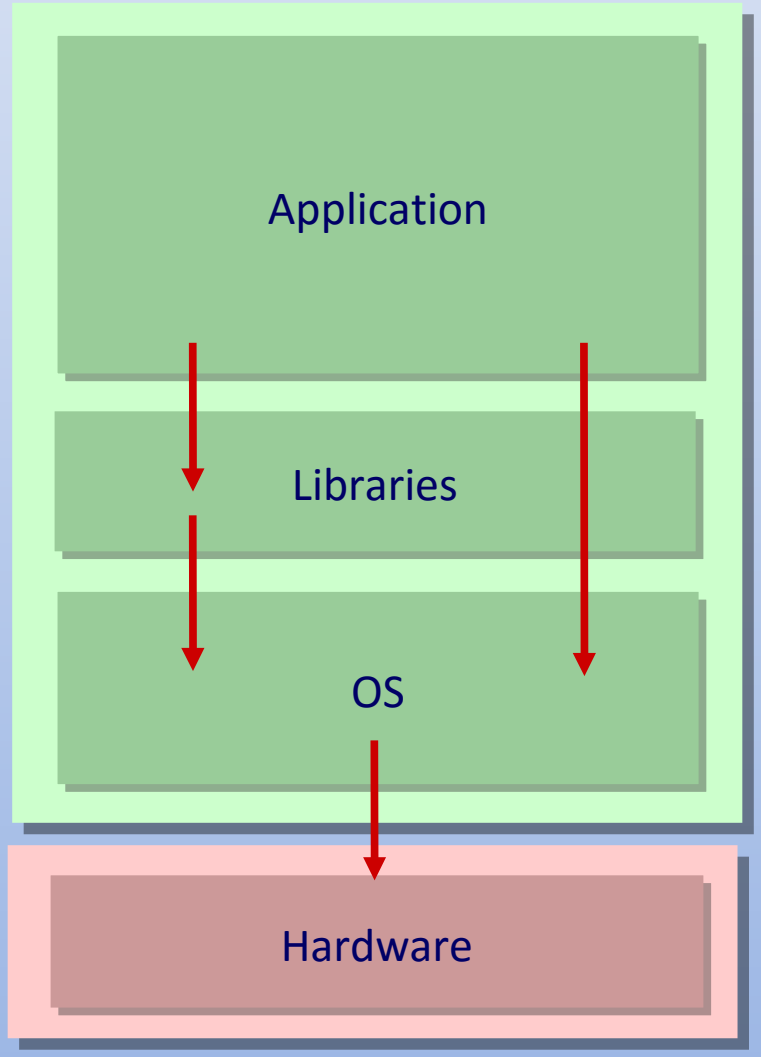
Frames, toolbars

Mouse/Keyboard input directed to window with focus

NxWidgets/NxWM

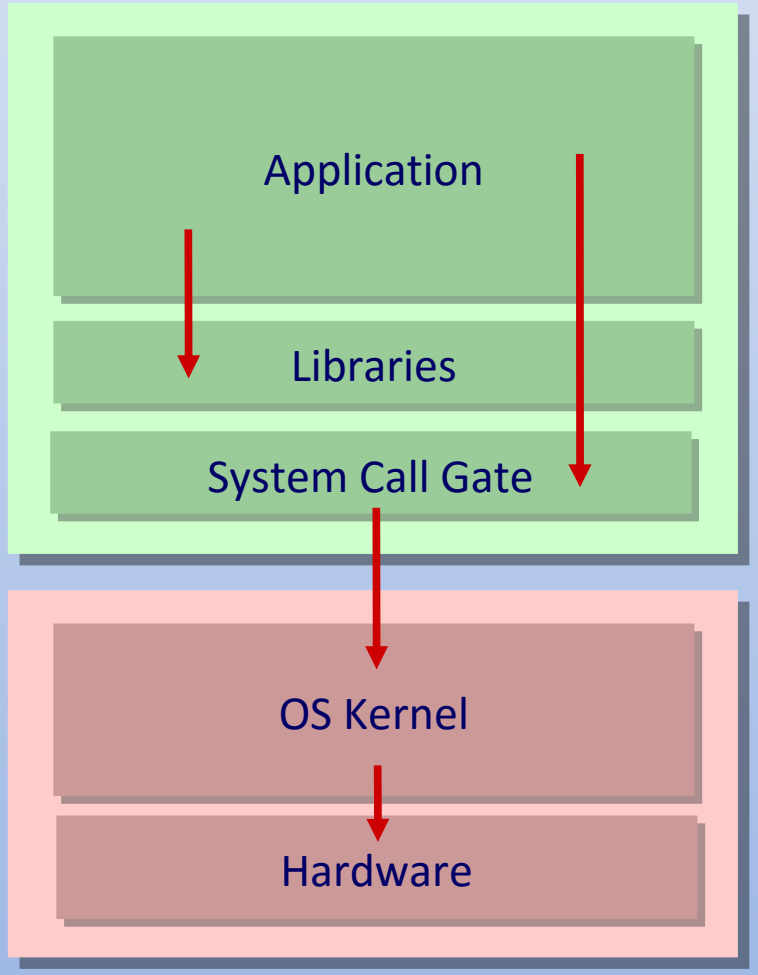


Build Modes: *Flat*



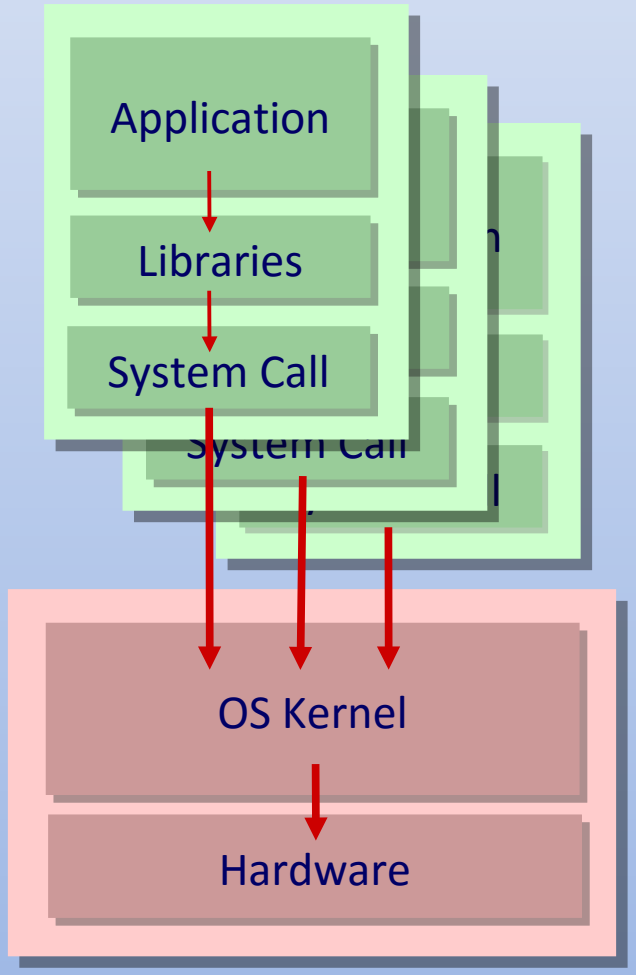
- *Flat* address space: All memory accessible by the application.
- Nothing is protected.
- Application can directly access OS resources
- No special hardware required
- `CONFIG_BUILD_FLAT=y`

Build Modes: *Protected*



- OS resources protected from application.
- Application accesses OS resources indirectly through a *call gate*
- Require Memory Protection Unit (MPU)
- `CONFIG_BUILD_PROTECTED=y`

Build Modes: *Kernel*



- *Processes*: Each application protected in a separate address space.
- Requires Memory Management Unit (MMU)
- `CONFIG_BUILD_KERNEL=y`

Key Applications: apps/ Package

Examples and tests

Graphics utilities and games

Interpreters

- FICL, BAS, Micropython
- Lua ports available

ModBus

- FreeModBus version 1.5.0.

Networking Utilities

- DHCP/DHCPC, FTPC/FTPD, SMTP, TELNET, TFTPC, HTTP, wget
- THTTPD web server with true embedded CGI, UDP Network Discovery, XML RPC Server, cJSON, and more.

Key Applications (Cont'd)

The NuttShell (NSH)

System Utilities

- Zmodem (sz and rz).
- Intel HEX conversions.
- USB connection management
- USB trace monitor
- PHY Link Status Management
- Command line audio player
- RAM testing
- Application Libraries

NuttShell

- Inspired by bash
- Conditionals (`if-then-else-fi`)
- Looping (`while-done`, `until-done`, `break`)
- NSH Scripts, Startup scripts, login scripts
- Background execution (`&`)
- Pipes (`>`, `>>`)
- Multiple telnet and console sessions
- Environment variables

bash-like: `addroute`, `cat`, `cd`, `cp`, `date`, `dd`, `delroute`, `df`, `echo`, `free`, `exit`, `get`, `ifconfig`, `kill`, `losetup`, `ls`, `ps`, `md5`, `mkdir`, `mkfatfs`, `mkfifo`, `mount`, `mv`, `nfsmount`, `nice`, `ping`, `ping6`, `poweroff`, `pwd`, `reboot`, `rm`, `rmdir`, `set`, `sh`, `shutdown`, `sleep`, `test`, `[`, `umount`, `unset`, `usleep`, `wget`, `xd`

NSH-only: `base64dec`, `exec`, `get`, `ifdown`, `ifup`, `help`, `hexdump`, `mb|h|w`, `mkrd`, `put`, `urldecode`, `urlencode`

Online Resources

NuttX DOT Org

NuttX website

http://www.nuttx.org

Forum

https://groups.yahoo.com/neo/groups/nuttx/info

Documentation

http://www.nuttx.org/doku.php?id=documentation

Wiki

http://www.nuttx.org/doku.php?id=wiki

Downloads

NuttX DOT Org

SourceForge project:

http://sourceforge.net/projects/nuttx

SourceForge Downloads:

http://sourceforge.net/projects/nuttx/files

Bitbucket.org Repositories:

https://bitbucket.org/patacongo/nuttx

https://bitbucket.org/nuttx/

Navigation

Home
Downloads
Documentation
Forum
Wiki
Links

Print/export

Download as PDF
Printable version

Toolbox

What links here
Recent changes
Media Manager
Site index
Permanent link
Cite this page

NuttX Real-Time Operating System

NuttX is a real-time operating system (RTOS) with an emphasis on standards compliance and small footprint. Scalable from 8-bit to 32-bit microcontroller environments, the primary governing standards in NuttX are Posix and ANSI standards. Additional standard APIs from Unix and other common RTOS's (such as VxWorks) are adopted for functionality not available under these standards, or for functionality that is not appropriate for deeply-embedded environments (such as fork()).

NuttX was first released in 2007 by Gregory Nutt under the permissive BSD license.

Table of Contents

Key features

edit

- Standards Compliant.
- Core Task Management.
- Modular design.
- Fully preemptible.
- Naturally scalable.
- Highly configurable.
- Easily extensible to new processor architectures, SoC architecture, or board architectures. See Porting Guide.
- FIFO, round-robin, and "sporadic" scheduling.
- Realtime, deterministic, with support for priority inheritance.
- Tickless operation.
- POSIX/ANSI-like task controls, named message queues, counting semaphores, clocks/timers, signals, pthreads, environment variables, filesystem.
- VxWorks-like task management and watchdog timers.
- BSD socket interface.
- Extensions to manage pre-emption.
- Optional tasks with address environments (*Processes*).