Forth and C on the Cortex-M3

Arm Cortex-M3

- 32-Bit Architecture
- Low-latency Prioritized Interrupt controller - NVIC
- Single-Cycle Multiply
- Sophisticated Debug
- C and Assembly friendly
- Common across all major Micro-controller vendors
- Available in Simplified (M0) and Enhanced (M4)

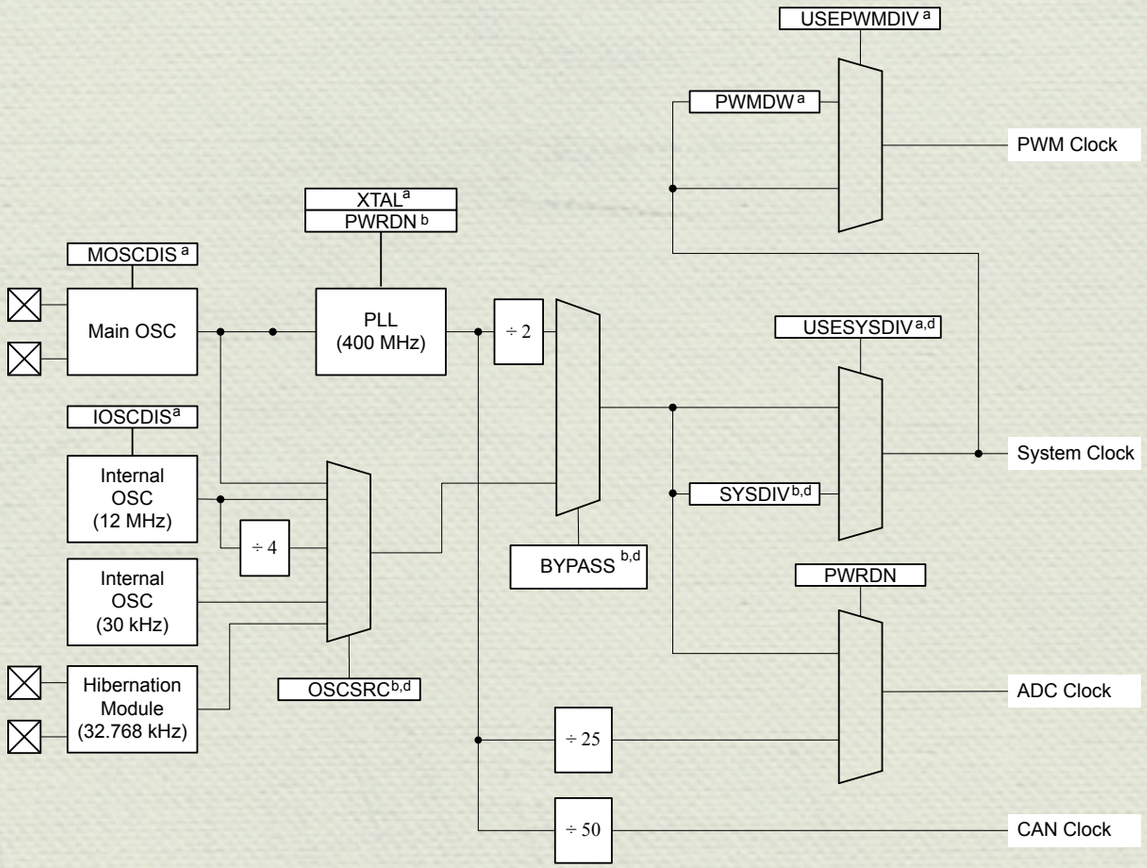
State of Forth on the M3

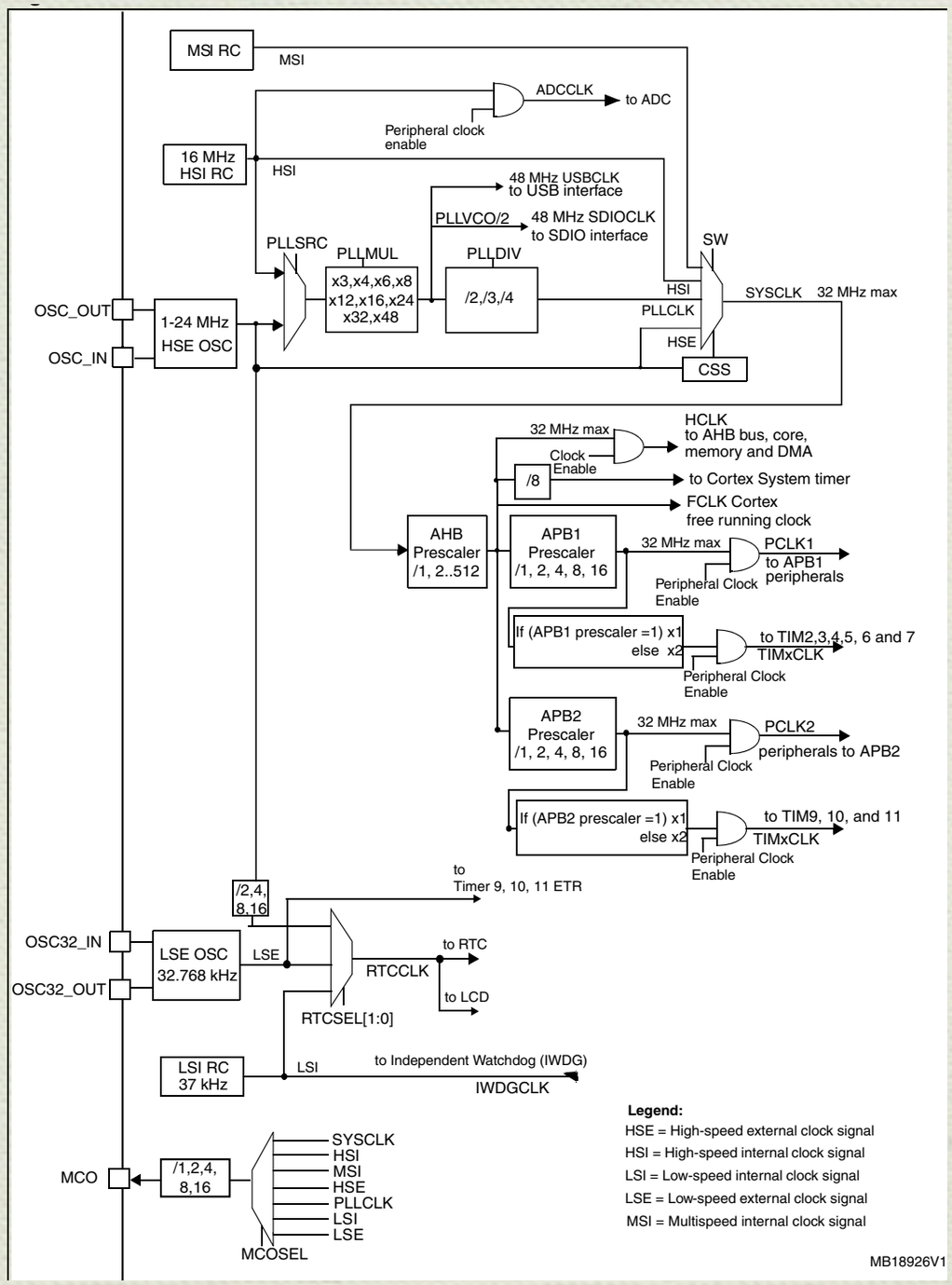
- Plenty of Forths for ARM
- Cortex-M3 is Thumb-2 - Few off the shelf implementations
  - Riscy Pygness Forth - Umbilical
  - MPE Forth - Commercial
- Machine Initialization varies widely across vendors

Options for Forth

- Port Riscy Pygness to a new flavor of M3
  - Strip out machine init code, and call from C
    - `main() { ..... ; asm("b initForth"); }`
- Port HForth or similar direct-threaded Forths
  - Rely heavily upon assembler macros
- Commercial Forth

Clock Trees!


USEPWMDIV a
PWMDW a
PWM Clock
XTAL a
PWRDN b
MOSCDIS a
Main OSC
PLL
(400 MHz)
÷ 2
USESYSDIV a,d
IOSCDIS a
Internal
OSC
(12 MHz)
÷ 4
SYSDIV b,d
System Clock
BYPASS b,d
Internal
OSC
(30 kHz)
PWRDN
Hibernation
Module
(32.768 kHz)
OSCSRC b,d
ADC Clock
÷ 25
÷ 50
CAN Clock


MSI RC
MSI
ADCCLK
to ADC
Peripheral clock
enable
16 MHz
HSI RC
HSI
48 MHz USBCLK
to USB interface
PLLVCO/2
48 MHz SDIOCLK
to SDIO interface
PLLSRC
PLLMUL
PLLDIV
SW
x3,x4,x6,x8
x12,x16,x24
x32,x48
/2,/3,/4
HSI
PLLCLK
HSE
SYSCLK
32 MHz max
OSC_OUT
OSC_IN
1-24 MHz
HSE OSC
CSS
HCLK
to AHB bus, core,
memory and DMA
32 MHz max
Clock
Enable
/8
to Cortex System timer
FCLK Cortex
free running clock
AHB
Prescaler
/1, 2..512
APB1
Prescaler
/1, 2, 4, 8, 16
32 MHz max
PCLK1
to APB1
peripherals
Peripheral Clock
Enable
If (APB1 prescaler =1) x1
else x2
to TIM2,3,4,5, 6 and 7
TIMxCLK
Peripheral Clock
Enable
APB2
Prescaler
/1, 2, 4, 8, 16
32 MHz max
PCLK2
peripherals to APB2
Peripheral Clock
Enable
If (APB2 prescaler =1) x1
else x2
to TIM9, 10, and 11
TIMxCLK
Peripheral Clock
Enable
/2,4,
8,16
to
Timer 9, 10, 11 ETR
OSC32_IN
OSC32_OUT
LSE OSC
32.768 kHz
LSE
RTCCLK
to RTC
to LCD
RTCSEL[1:0]
LSI RC
37 kHz
LSI
to Independent Watchdog (IWDG)
IWDGCLK
MCO
/1,2,4,
8,16
SYSCLK
HSI
MSI
HSE
PLLCLK
LSI
LSE
MCOSEL
Legend:
HSE = High-speed external clock signal
HSI = High-speed internal clock signal
LSI = Low-speed internal clock signal
LSE = Low-speed external clock signal
MSI = Multispeed internal clock signal
MB18926V1

Bootstrapping the M3

- Complex Initialization/Reconfiguration
- Large body of demonstration/reusable code
- Re-Implementation in Forth is a bad option
  - Opportunities for Error
  - Time is best spent on Application Development

Existing Vendor Code

- TI StellarisWare
  - `SysCtlClockSet()` - 177 Lines of Code
  - `ADCSequenceStepConfigure()` - 90 Lines of Code
- ST Micro Driverlib
  - `SetSysClock()` - 82 Lines of Code
  - `ADC_Init()` - 52 Lines of Code

Code thats rich in bit-whacking and specific rules

Partitioning

Application Layer
31k Flash/12k Ram

Application Development
Control Interface
Debugging/Bringup

System Services - 13k/4k

USB CDC Services
Timing Services
Frame Buffer

DFU Boot loader - 8.5k/0k

Inactive after initial boot

System Call Interface

- Minimal System Call Support:
  - API Version
  - getchar, putchar
  - get shared variables list
  - set power state
  - get current value of millisecond counter

Forth SVC Calls

```
\ **********************************************
\ SVC 0: Return the version of the API in use.
\ **********************************************
CODE API-Version  ( -- n )
   svc #0                        ( Call Supervisor)
   str tos, [ psp, # -4 ] !      ( Push TOS)
   mov tos, r0                   ( return value)
   next,
END-CODE
```

Note: Cortex-M3 Pushes R0-R4, R12, LR, PC, xPSR automatically

Catching it!

Handler Looks up the SVC Number
and jumps via table to the right function:

```
// Return an API Version
void __SAPI_Version(long *frame) {
        frame[0] = 0x0201;
        return;
        }
```

Handler-Thread Sync

- Interrupt handlers need to send ‘Events’ to each other and to the Thread-Mode tasks.
- Unidirectional events makes this safe:
  - ISR: `notify_1Hz = 1;`
  - Thread resets the event register and waits for change:
    - `0 notify_1Hz !`
    - `notify_1Hz @ if <something> then`

TCB status byte

Bit	When set	When Reset
0	Task is running	Task is halted
1	Message pending but not read	No messages
2	Event triggered	No events
3	Event handler has been run	No events (reset by user)
4..	User defined	User defined

- Scheduler uses non-zero value to trigger task execution or event hander
- ISR can set the event bit to trigger task execution
- Scheduler calls event handler then the task

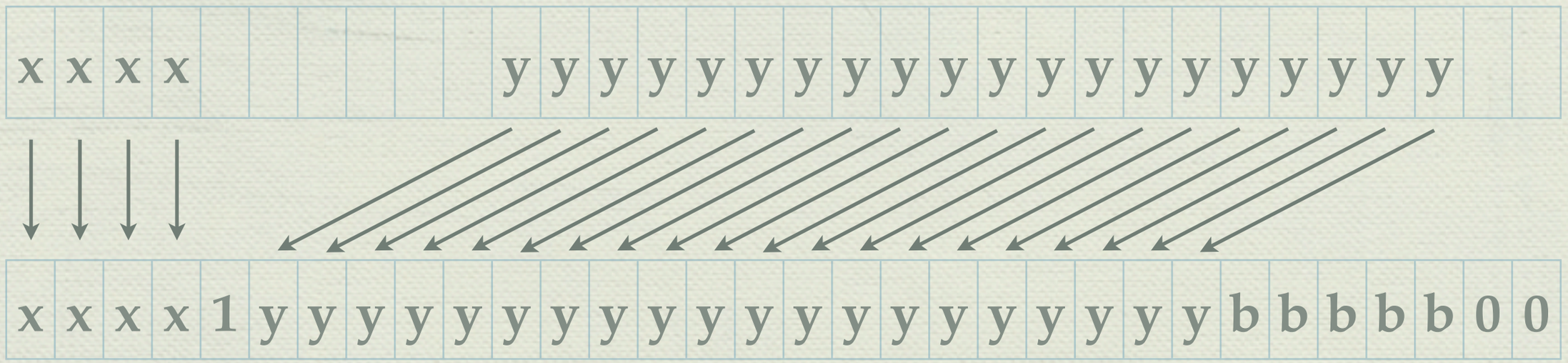
Hooking it into a Thread

```
: update-hue-out ( -- )
  stop
  begin
        rtc @ calc_hue rtc2hue_out !
        pause
  again
;


task clock2huetask

: launch-tasks
  ['] clr-event-run clock2huetask to-event
  ['] update-hue-out clock2huetask initiate
  ;
```

Bit-Banding on the M3

Atomic!

Target Address -> Bit Band Alias Transformation



Bit
Number

Lock-Avoidance

- Bit-Banding eliminates interrupt lockout
  - Fewer hazards
  - Similar instruction count
  - add an extra word to the TCB
- if (notify_refresh_bbp != 0) { *notify_refresh_bbp = 1; }

```
: clr-event-run    \ --
\  * Reset the current task's  EVENT_RUN flag.
   0    up@ tcb.bbstatus @ evt-bit# +  !  ;
```

Dynamic Linking

- Linker places C variables where it sees fit
- Manually updating pre-defined constants is error-prone
- Table-Driven approach allows automatic updates

```
runtimelink_t dynamiclinks[] = {
{ (int*) sizeof(runtimelink_t), sizeof("RECORDLE"), "RECORDLEN", 0, 0},
{ (int*) hsv_in,        sizeof("hsv_i"),     "hsv_in",    sizeof(uint32_t),  3 } };
```

The address of this array is available via system call.
Names are encoded as counted strings

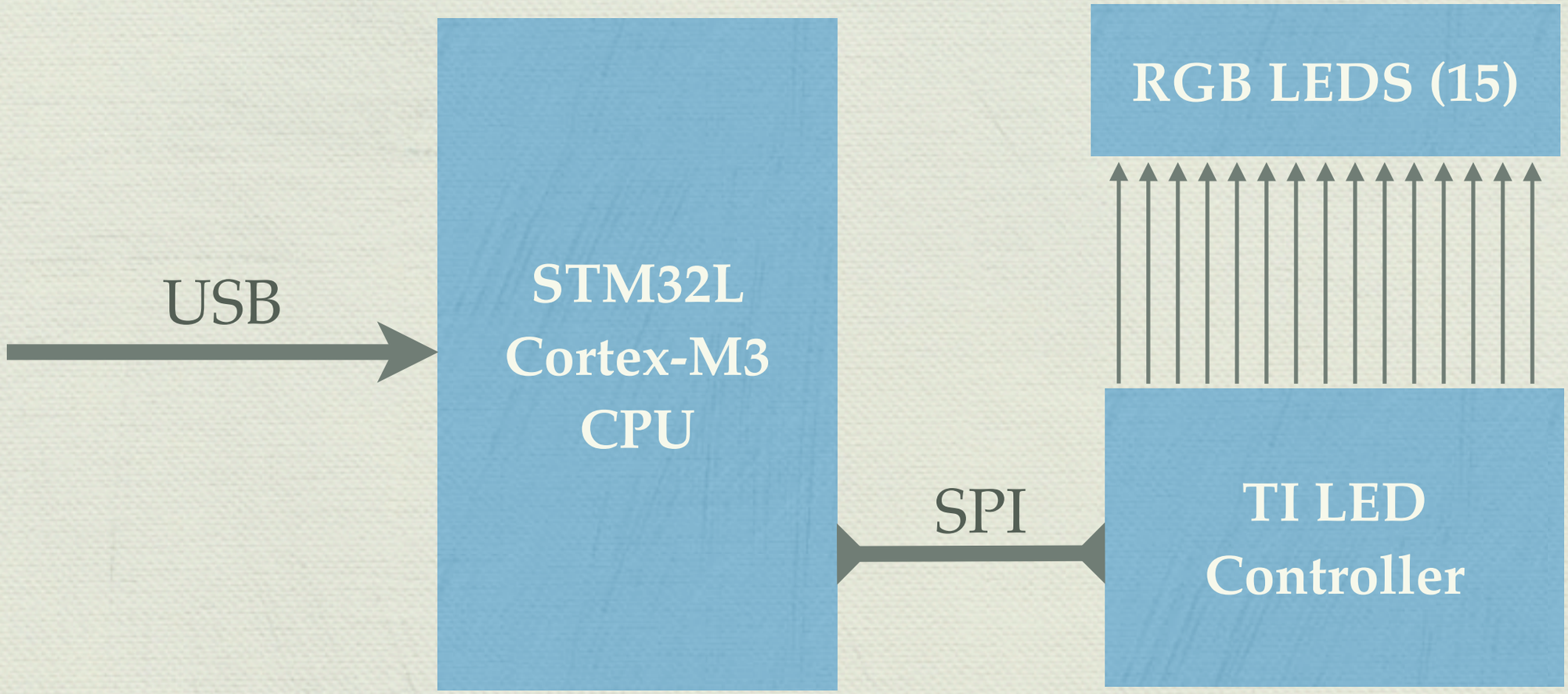
Questions?

http://www.kudra.com/forth

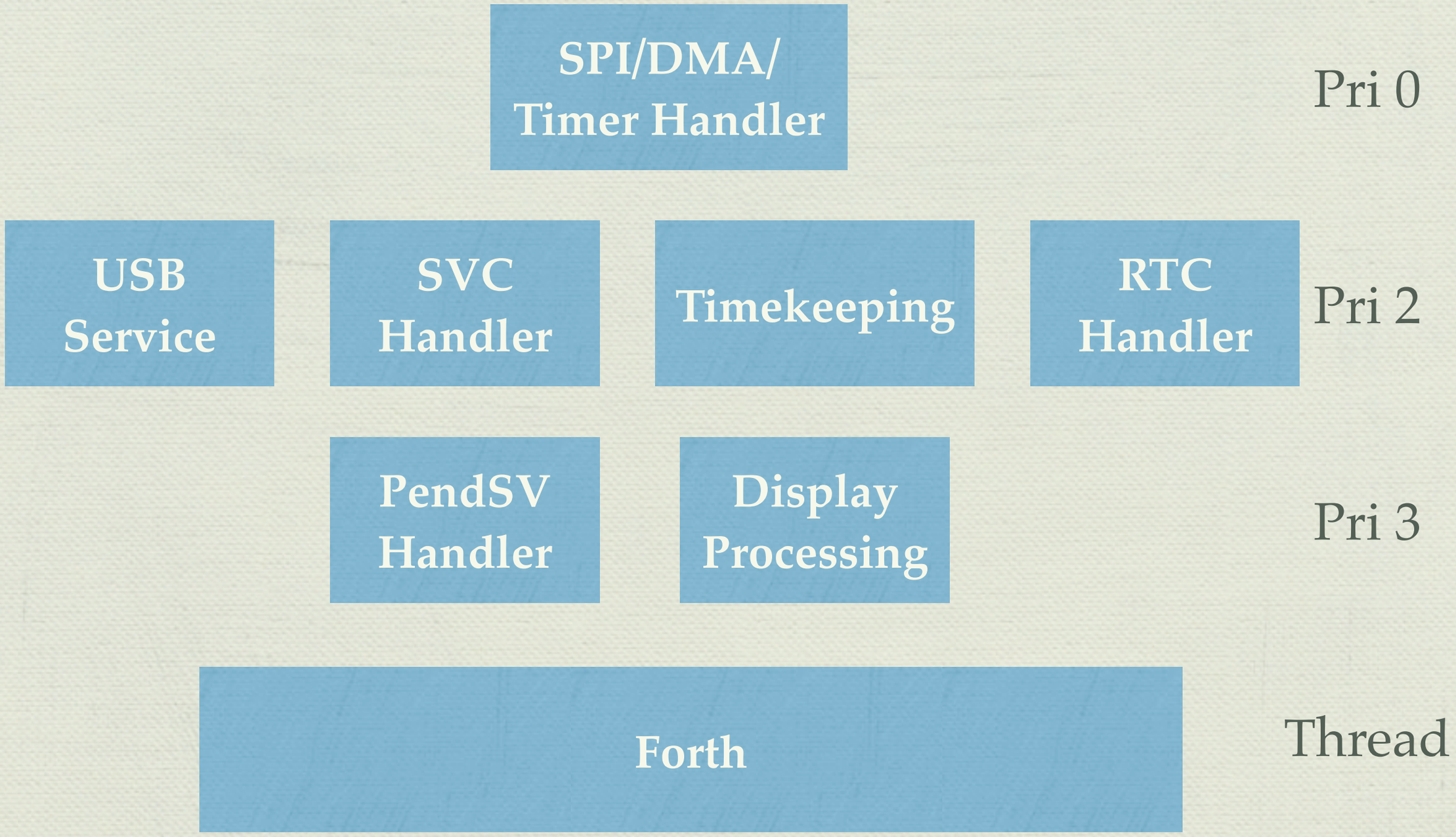
Ambient Devices

- Invented at the MIT Media Lab
- Concept - Low Bandwidth Information
  - Weather Forecasts
  - Stock Prices
  - Mail Notification

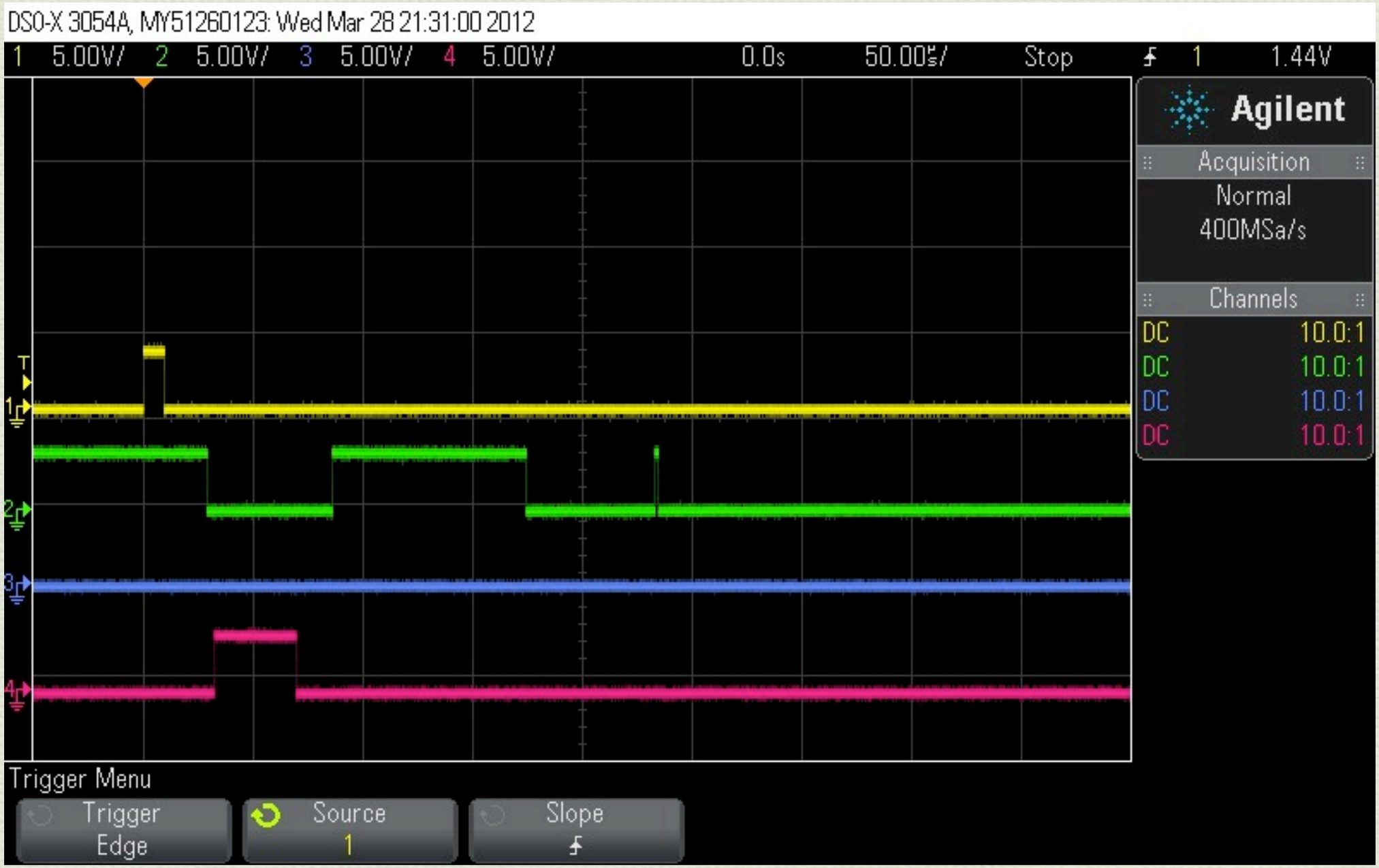
Hardware


RGB LEDS (15)
USB
STM32L
Cortex-M3
CPU
SPI
TI LED
Controller
192 bits / 128 Hz

Firmware Architecture



NVIC Priority Behavior



Power Consumption

- Exclusive of LED Power
  - 10 mA in USB Connected Mode
    - 7.7mA waiting for a connection
    - 1.5mA in suspend (131 kHz CPU)
    - Suspend/Wake Changes CPU Voltage and Frequency
- More opportunities remain for power reduction

USB Integration

- SendChar puts an application byte into a transmission buffer
- Minimize the number of USB packets:
  - Send if there is a full USB buffer
  - Send if the buffer is old (> 10ms)
- If the transmission buffer is too full, wait for the USB interrupt handlers to drain it before returning control to the application (PendSV mechanism)

User Interface

- USB CDC Device - Ascii Command Set
- 60 hue, $ffff 0 0 rgbset, etc.
- fast pulse, slow pulse
- extensible