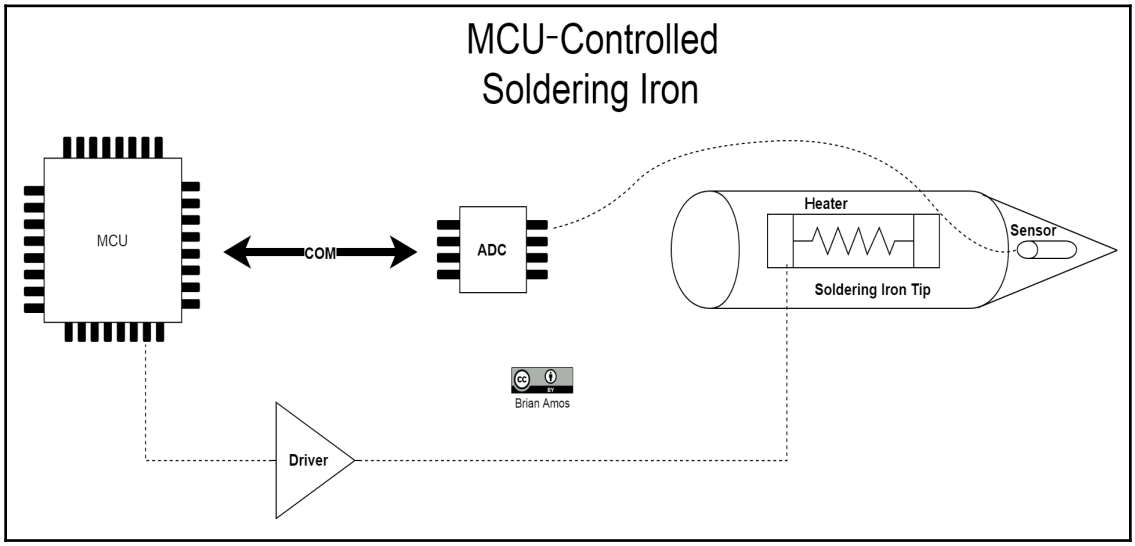
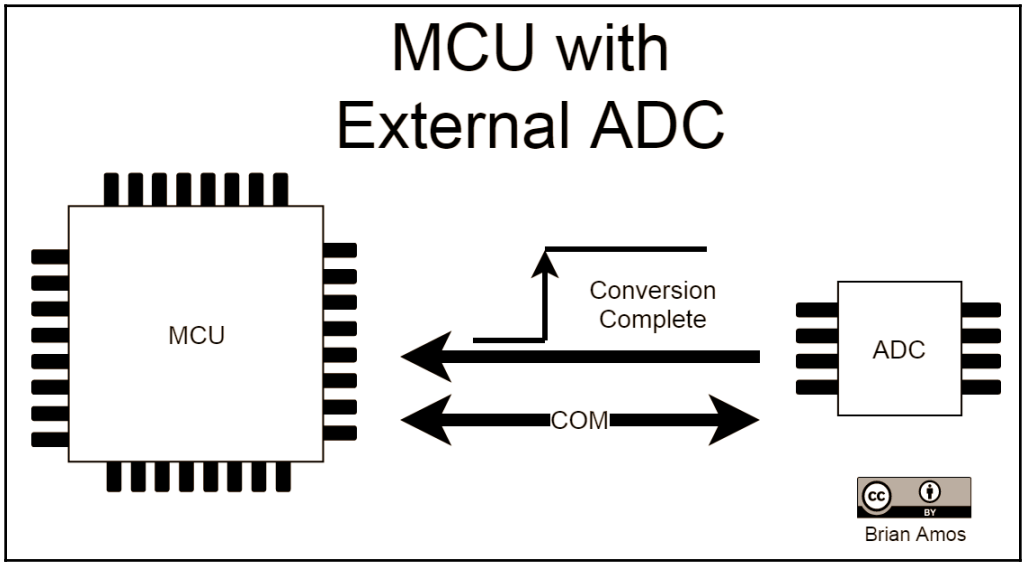
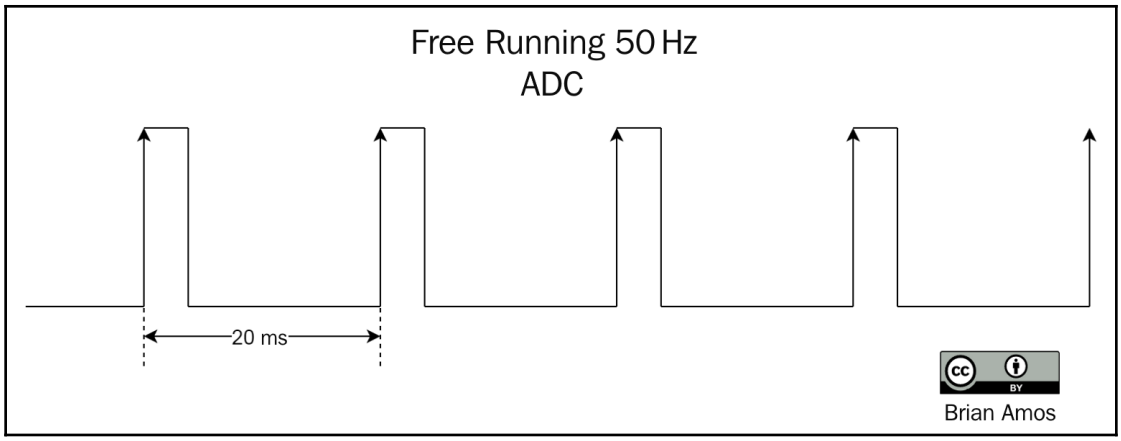
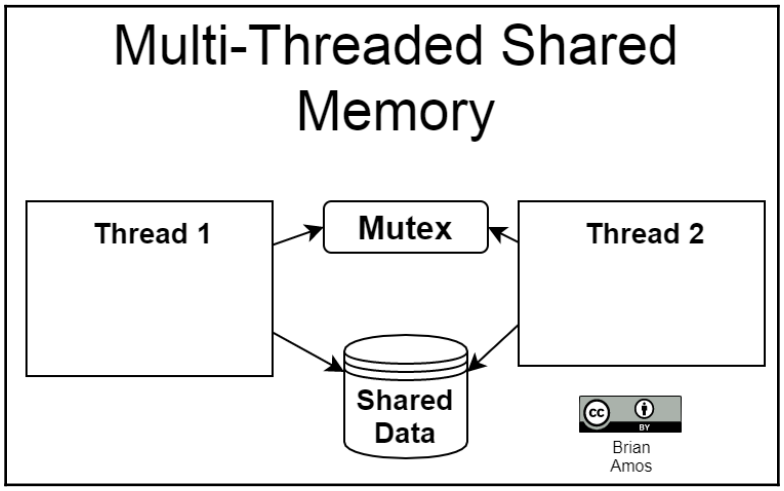
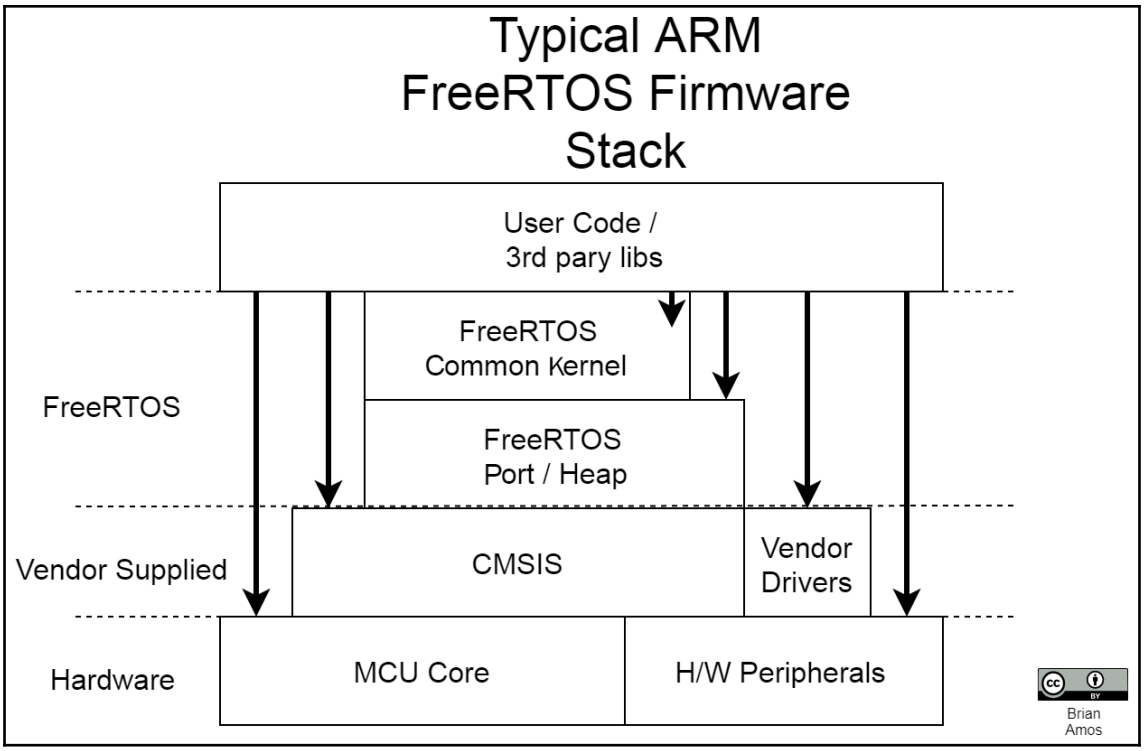
Chapter 1: Introducing Real-Time Systems



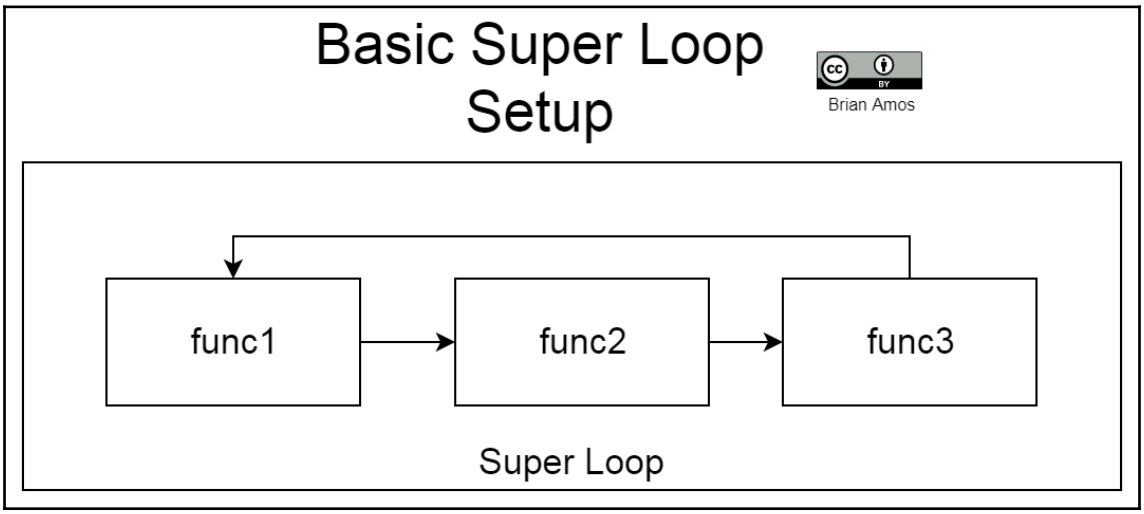


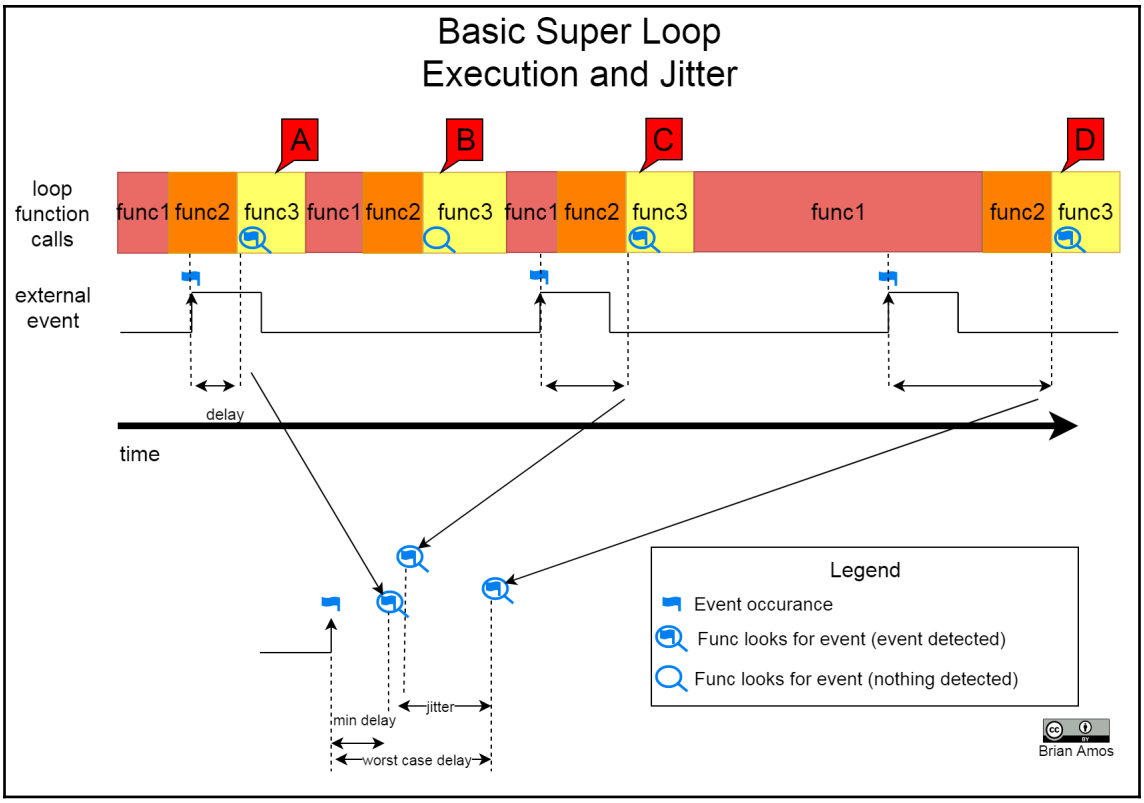

Free Running 50 Hz
ADC
20 ms
Brian Amos

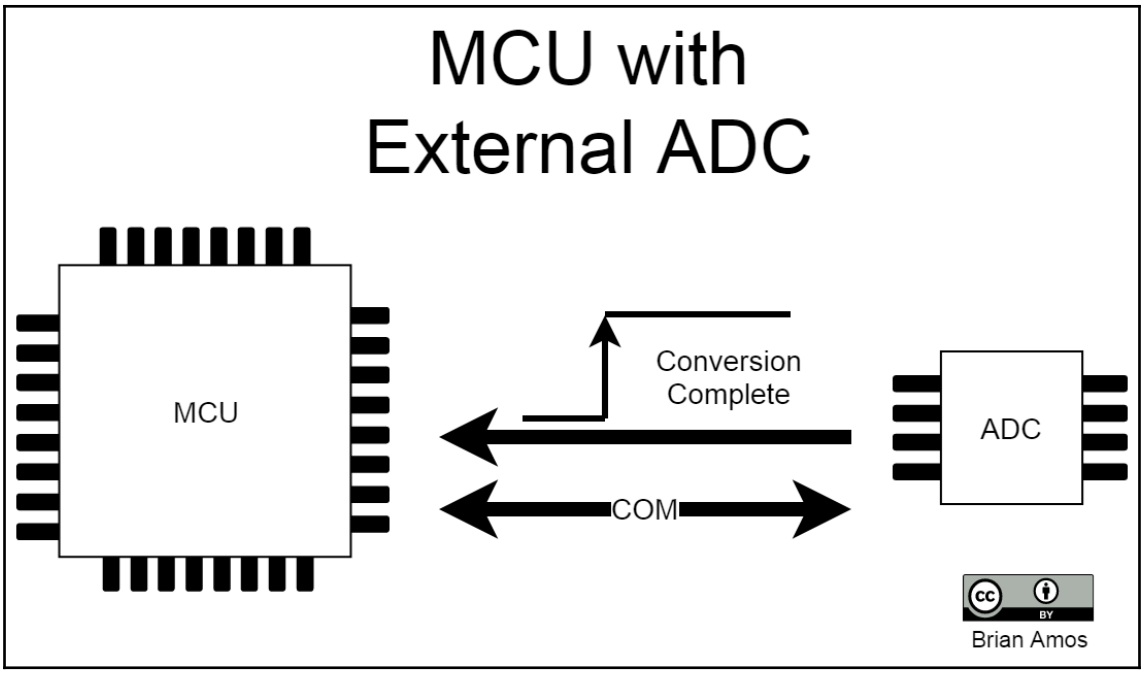

Multi-Threaded Shared Memory
Thread 1
Mutex
Thread 2
Shared Data
Brian Amos

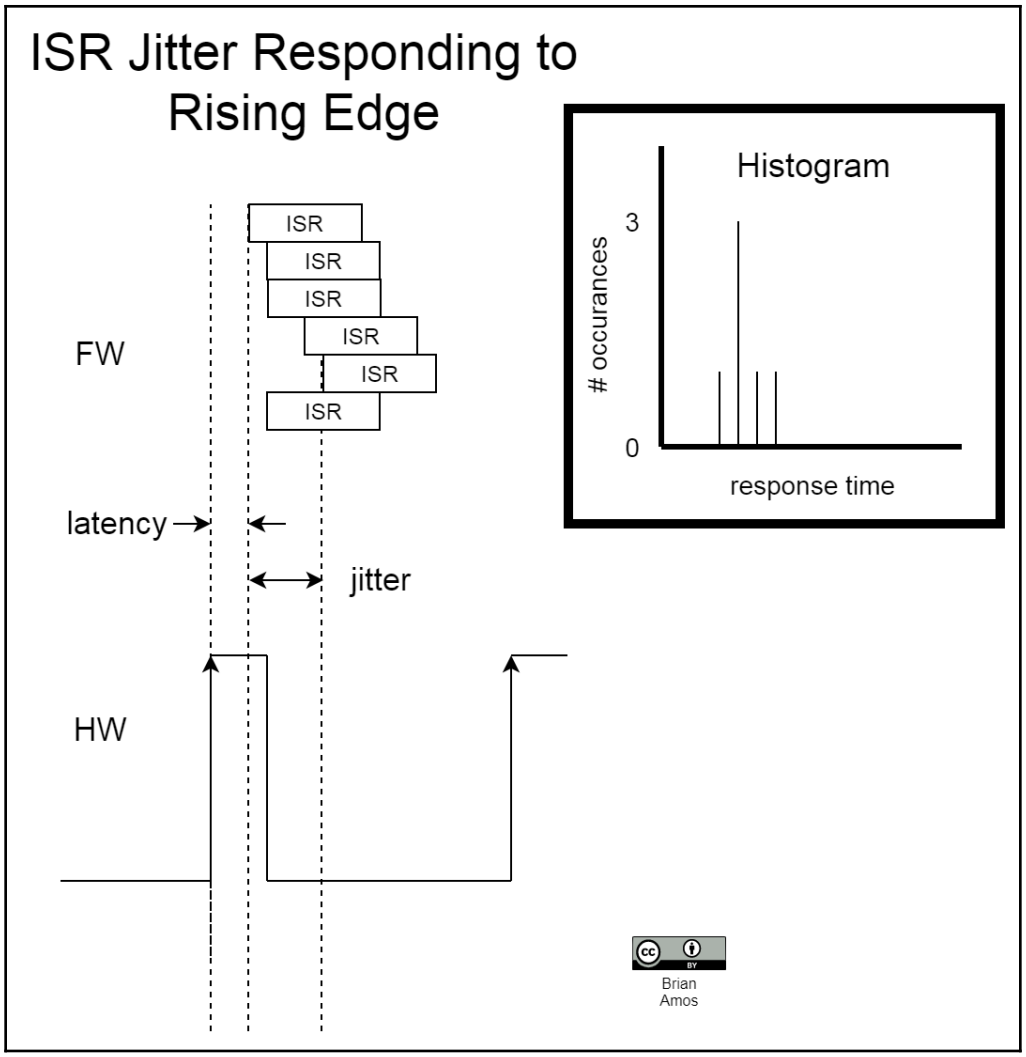

Typical ARM
FreeRTOS Firmware
Stack
User Code /
3rd pary libs
FreeRTOS
Common Kernel
FreeRTOS
FreeRTOS
Port / Heap
Vendor Supplied
CMSIS
Vendor
Drivers
Hardware
MCU Core
H/W Peripherals
Brian
Amos

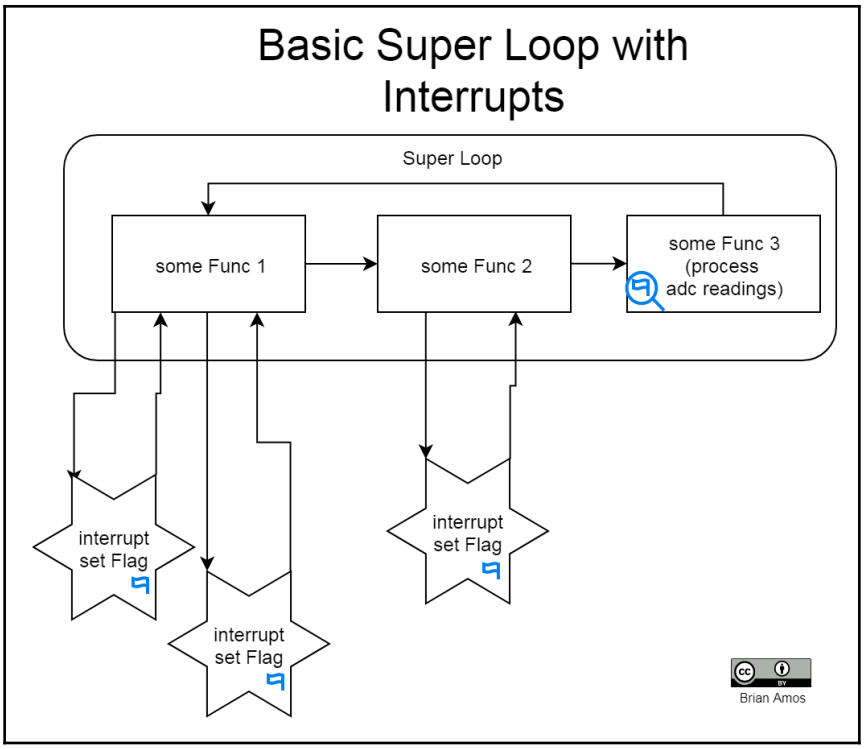
Chapter 2: Understanding RTOS Tasks

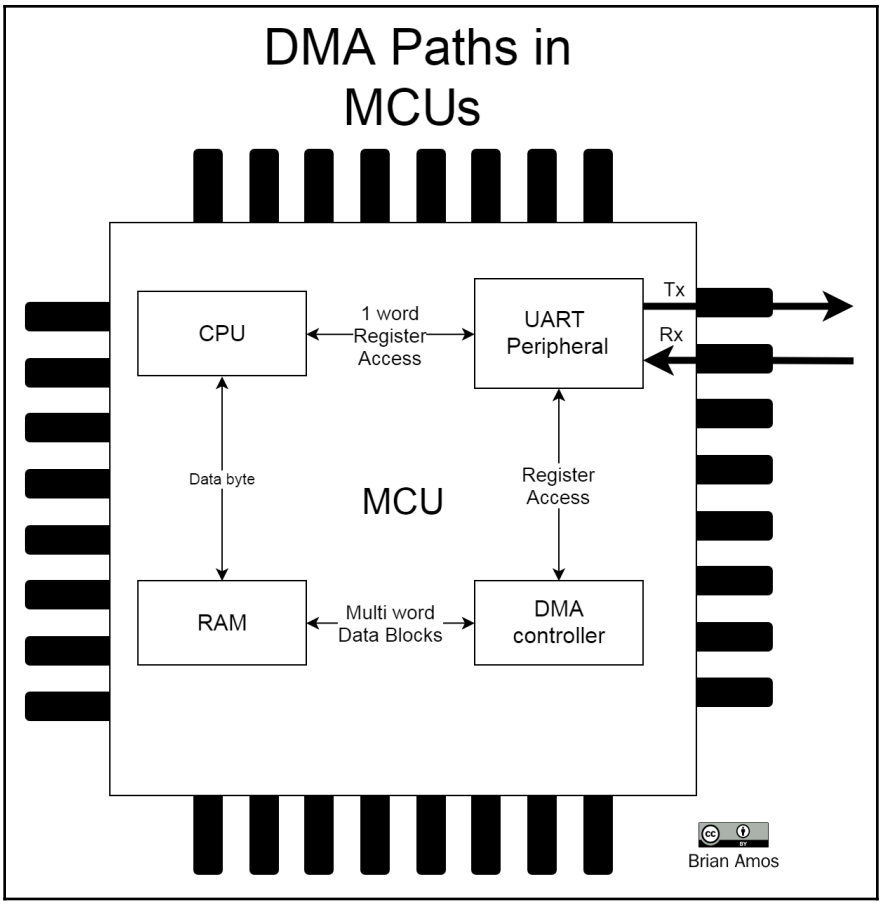



Basic Super Loop
Execution and Jitter
A
B
C
D
loop
function
calls
func1 func2 func3 func1 func2 func3 func1 func2 func3 func1 func2 func3
external
event
delay
time
Legend
Event occurance
Func looks for event (event detected)
Func looks for event (nothing detected)
jitter
min delay
worst case delay
Brian Amos


MCU with
External ADC
MCU
Conversion
Complete
COM
ADC
Brian Amos


ISR Jitter Responding to Rising Edge
Histogram
3
occurances
0
response time
ISR
ISR
ISR
ISR
ISR
ISR
FW
latency
jitter
HW
Brian Amos


Basic Super Loop with Interrupts
Super Loop
some Func 1
some Func 2
some Func 3 (process adc readings)
interrupt set Flag
interrupt set Flag
interrupt set Flag
Brian Amos

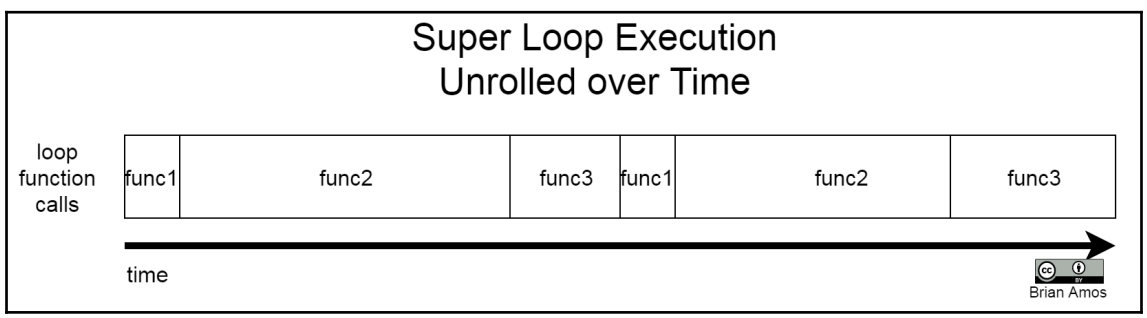

DMA Paths in MCUs
CPU
1 word Register Access
UART Peripheral
Tx
Rx
Data byte
MCU
Register Access
RAM
Multi word Data Blocks
DMA controller
Brian Amos

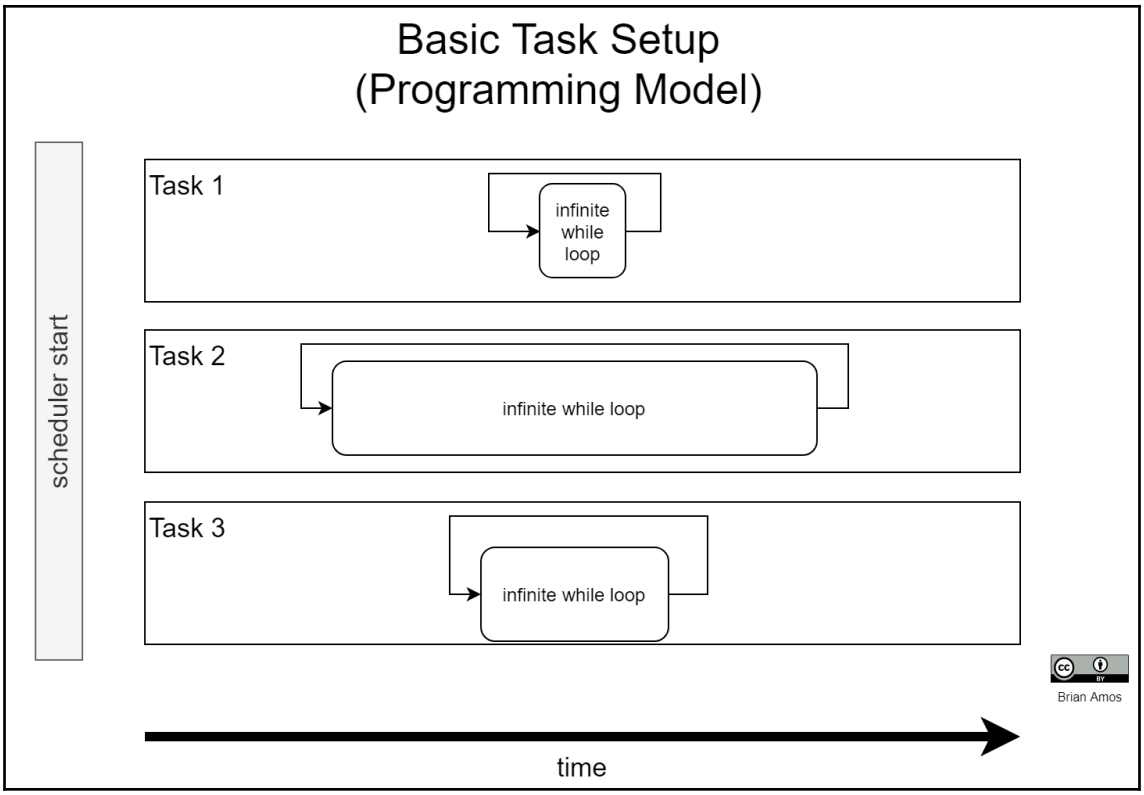
Super Loop

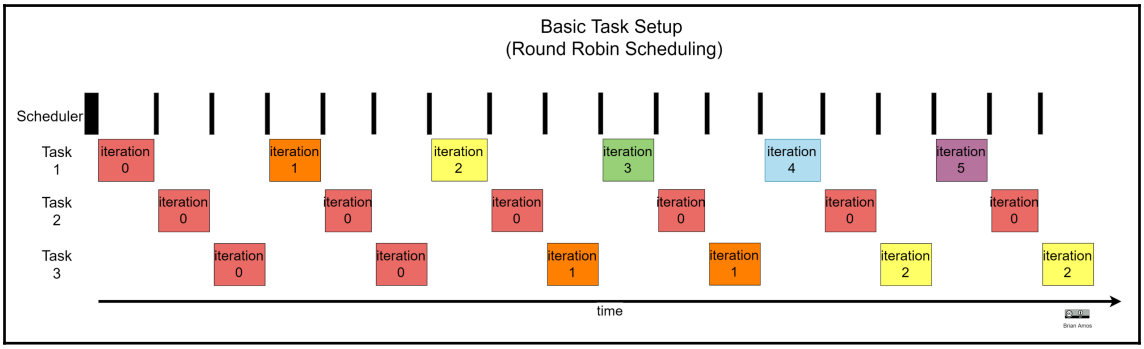
```
func1()
{
    //functionality 1
}

func2()
{
    //functionality 2
}

func3()
{
    //functionality 3
}

main( )
{
    while(1)
    {
        func1();
        func2();
        func3();
    }
}
```

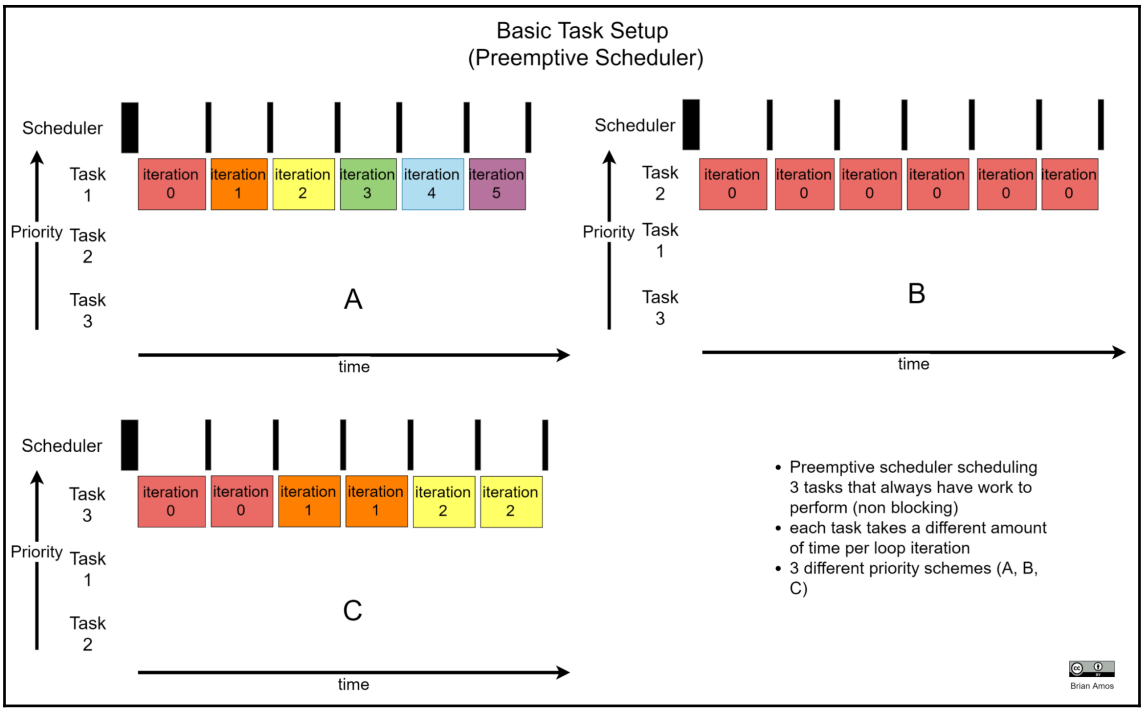
RTOS Tasks

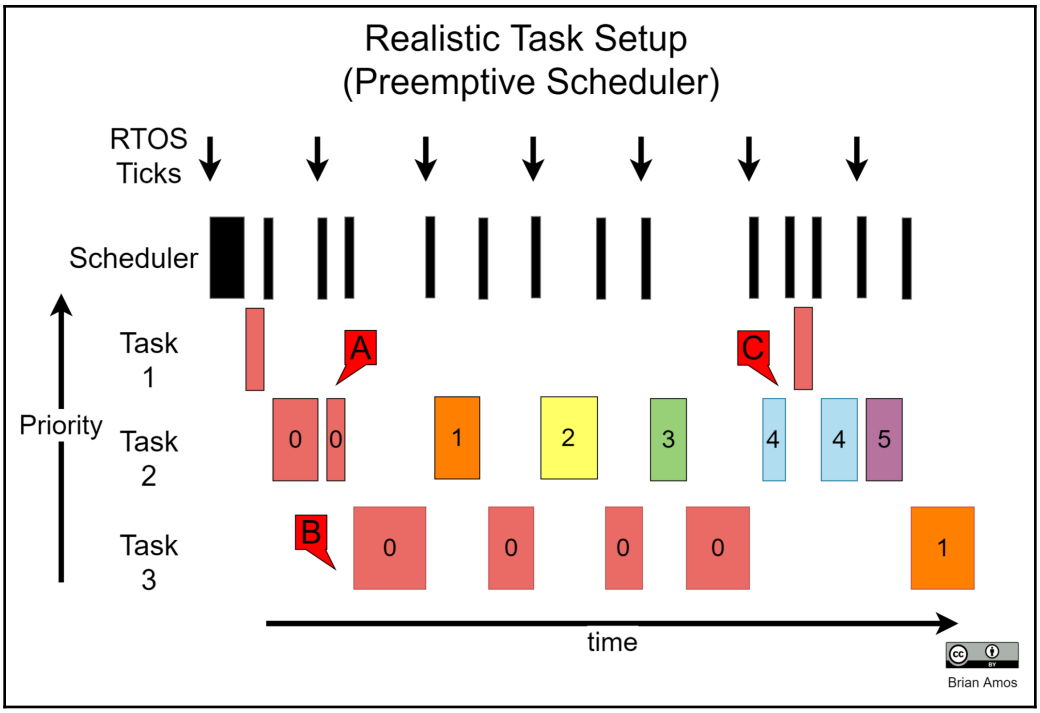
```
Task1( )
{
    while(1)
    {
        //functionality of task 1
    }
}

Task2( )
{
    while(1)
    {
        //functionality of task 2
    }
}

Task3( )
{
    while(1)
    {
        //functionality of task 3
    }
}

main( )
{
    createTask1(&Task1);
    createTask2(&Task2);
    createTask3(&Task3);
    StartScheduler();   //never returns
}
```



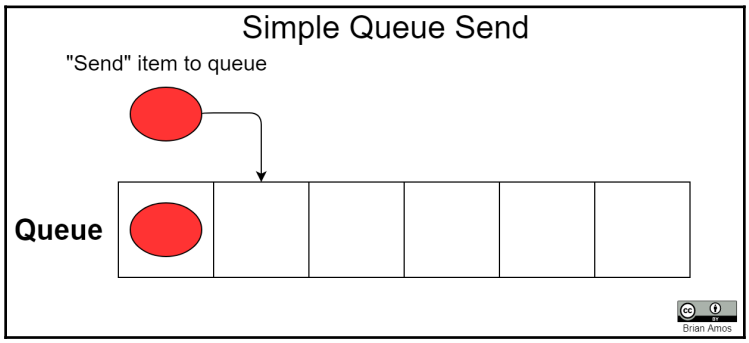

Basic Task Setup
(Programming Model)
scheduler start
Task 1
infinite
while
loop
Task 2
infinite while loop
Task 3
infinite while loop
Brian Amos
time

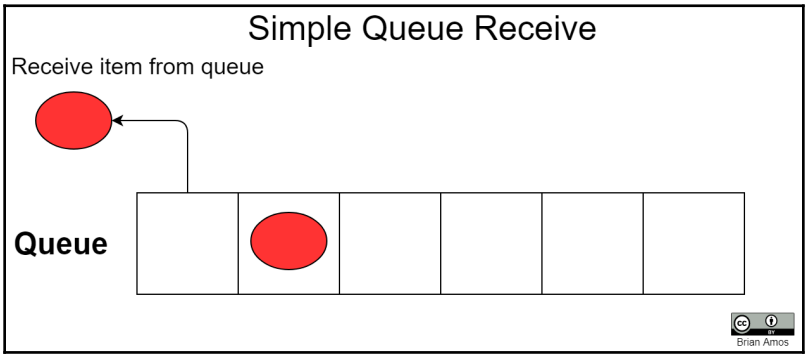

Basic Task Setup
(Round Robin Scheduling)
Scheduler
Task 1
Task 2
Task 3
iteration 0
iteration 1
iteration 2
iteration 3
iteration 4
iteration 5
time

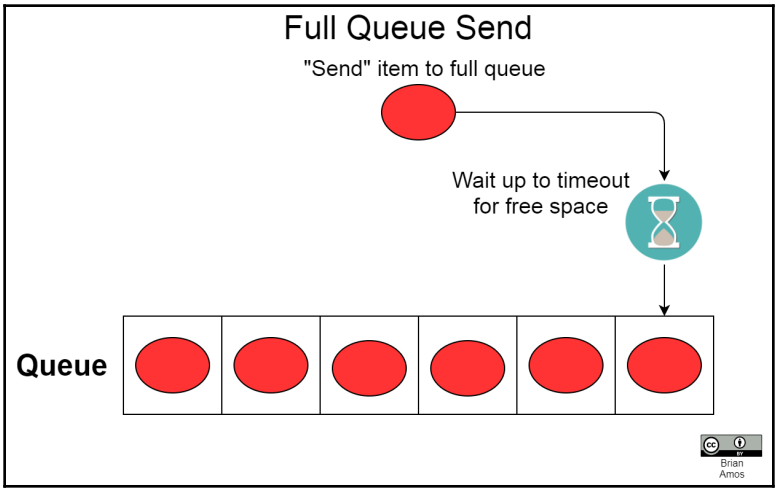

Basic Task Setup
(Preemptive Scheduler)
Scheduler
Priority
Task 1
Task 2
Task 3
iteration 0
iteration 1
iteration 2
iteration 3
iteration 4
iteration 5
A
time
Scheduler
Priority
Task 2
Task 1
Task 3
iteration 0
iteration 0
iteration 0
iteration 0
iteration 0
iteration 0
B
time
Scheduler
Priority
Task 3
Task 1
Task 2
iteration 0
iteration 0
iteration 1
iteration 1
iteration 2
iteration 2
C
time
• Preemptive scheduler scheduling 3 tasks that always have work to perform (non blocking)
• each task takes a different amount of time per loop iteration
• 3 different priority schemes (A, B, C)
Brian Amos

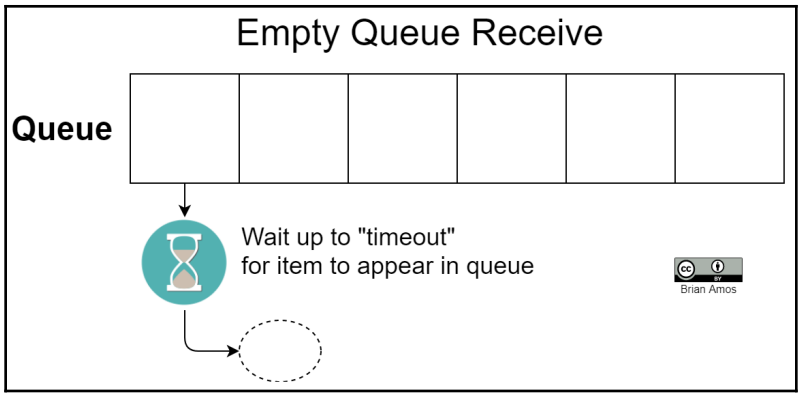

Realistic Task Setup
(Preemptive Scheduler)
RTOS
Ticks
Scheduler
Priority
Task
1
Task
2
Task
3
A
B
C
0
0
1
2
3
4
4
5
0
0
0
0
1
time
Brian Amos

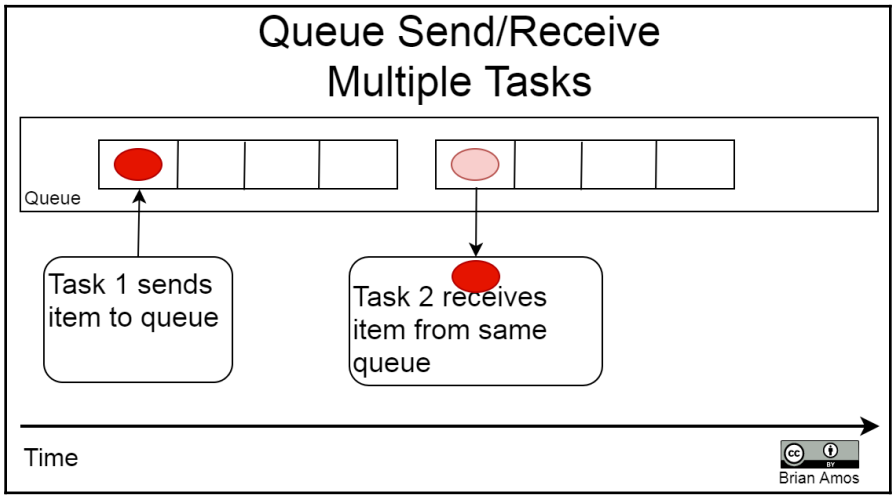
Chapter 3: Task Signaling and Communication Mechanisms

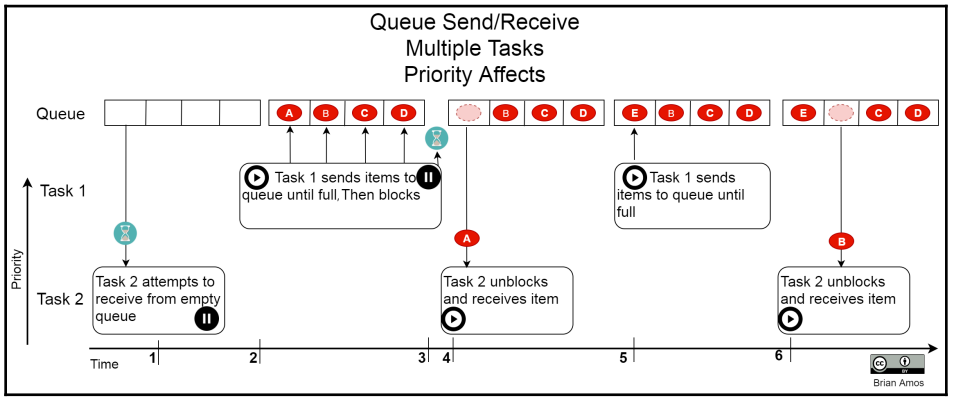


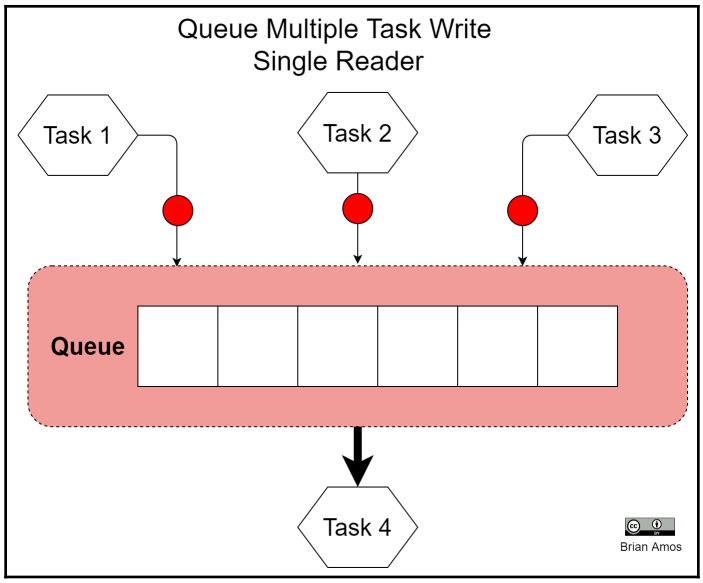


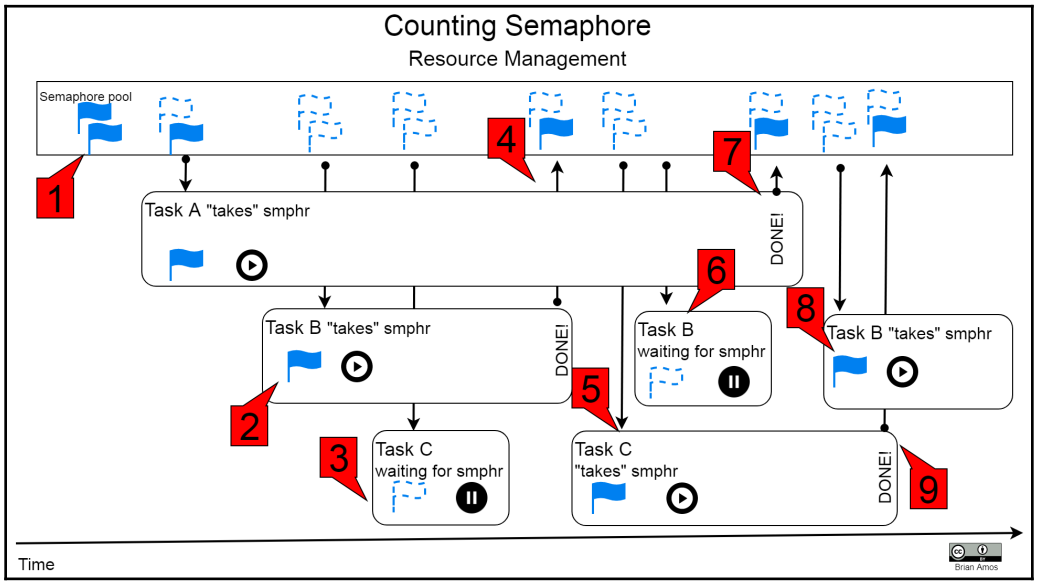


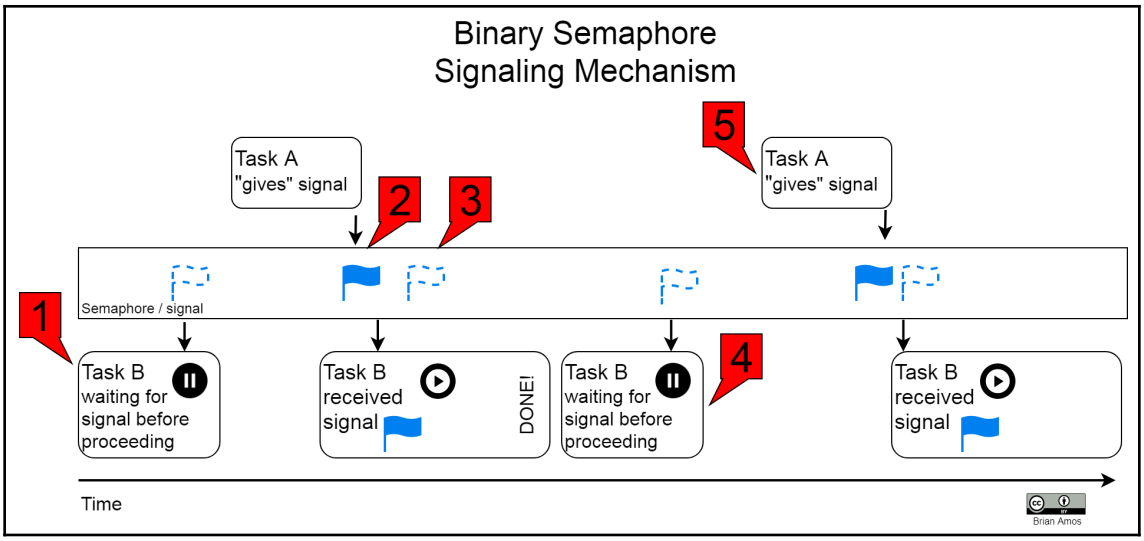

Empty Queue Receive
Queue
Wait up to "timeout"
for item to appear in queue
Brian Amos

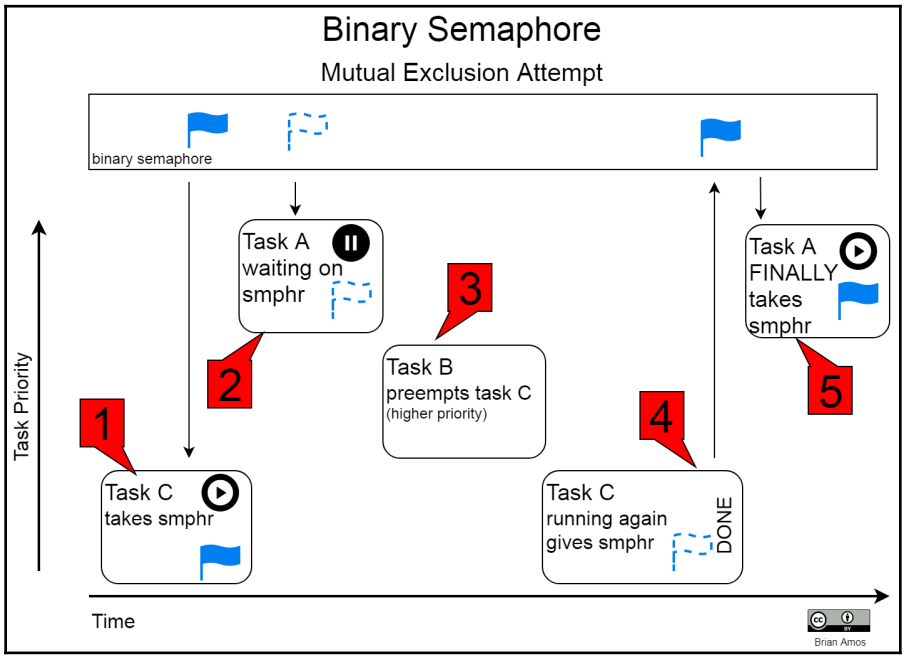

Queue Send/Receive
Multiple Tasks
Queue
Task 1 sends
item to queue
Task 2 receives
item from same
queue
Time
Brian Amos

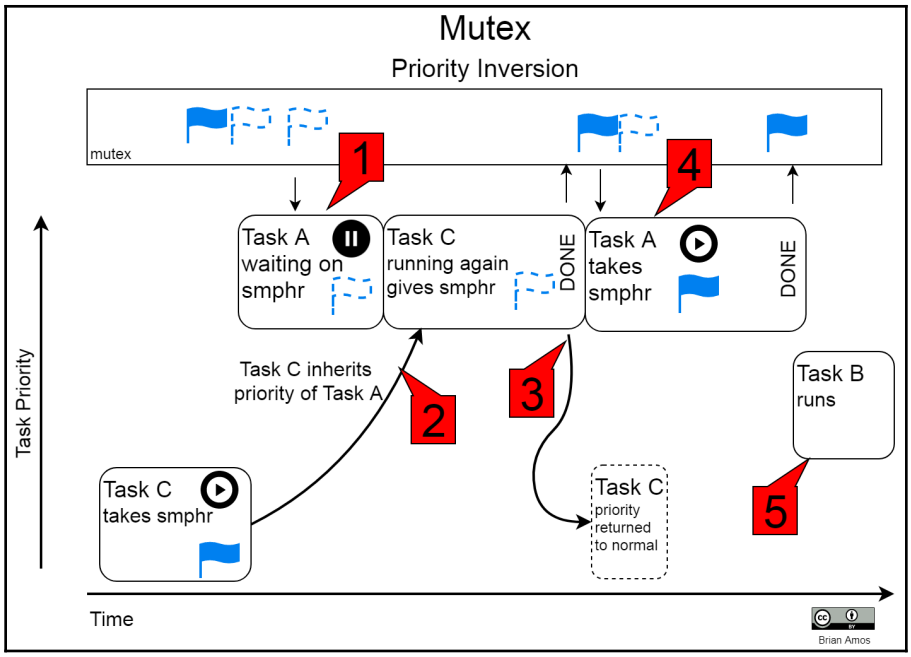

Queue Send/Receive
Multiple Tasks
Priority Affects
Queue
Task 1
Task 2
Priority
Task 1 sends items to queue until full, Then blocks
Task 1 sends items to queue until full
Task 2 attempts to receive from empty queue
Task 2 unblocks and receives item
Task 2 unblocks and receives item
Time
1
2
3
4
5
6
Brian Amos


Queue Multiple Task Write
Single Reader
Task 1
Task 2
Task 3
Queue
Task 4
Brian Amos

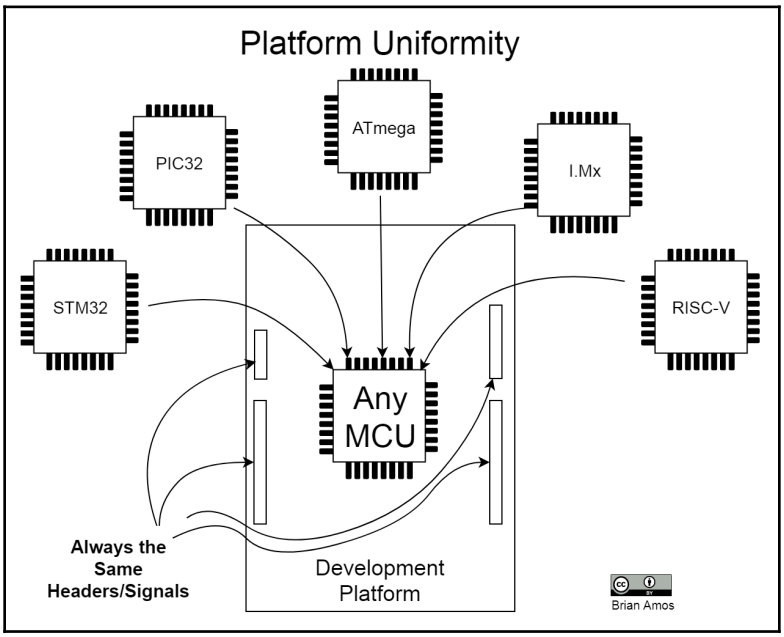

Counting Semaphore
Resource Management
Semaphore pool
1
Task A "takes" smphr
4
7
DONE!
6
Task B "takes" smphr
DONE!
2
Task B
waiting for smphr
8
Task B "takes" smphr
5
3
Task C
waiting for smphr
Task C
"takes" smphr
DONE!
9
Time
Brian Amos

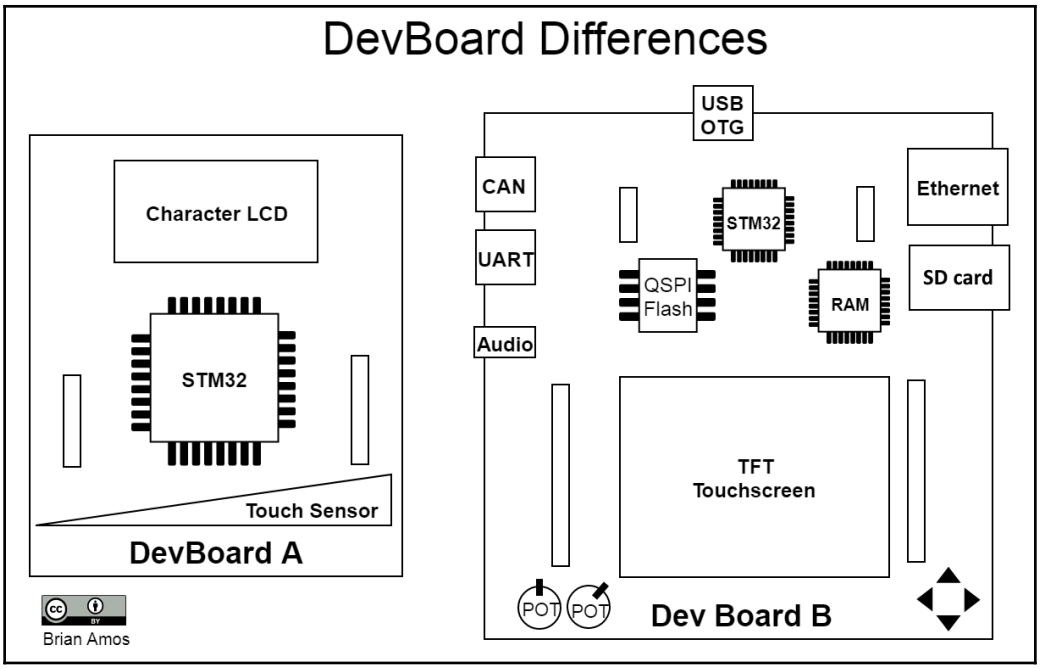

Binary Semaphore
Signaling Mechanism
Task A
"gives" signal
2
3
5
Task A
"gives" signal
Semaphore / signal
1
Task B
waiting for
signal before
proceeding
Task B
received
signal
DONE!
Task B
waiting for
signal before
proceeding
4
Task B
received
signal
Time
Brian Amos

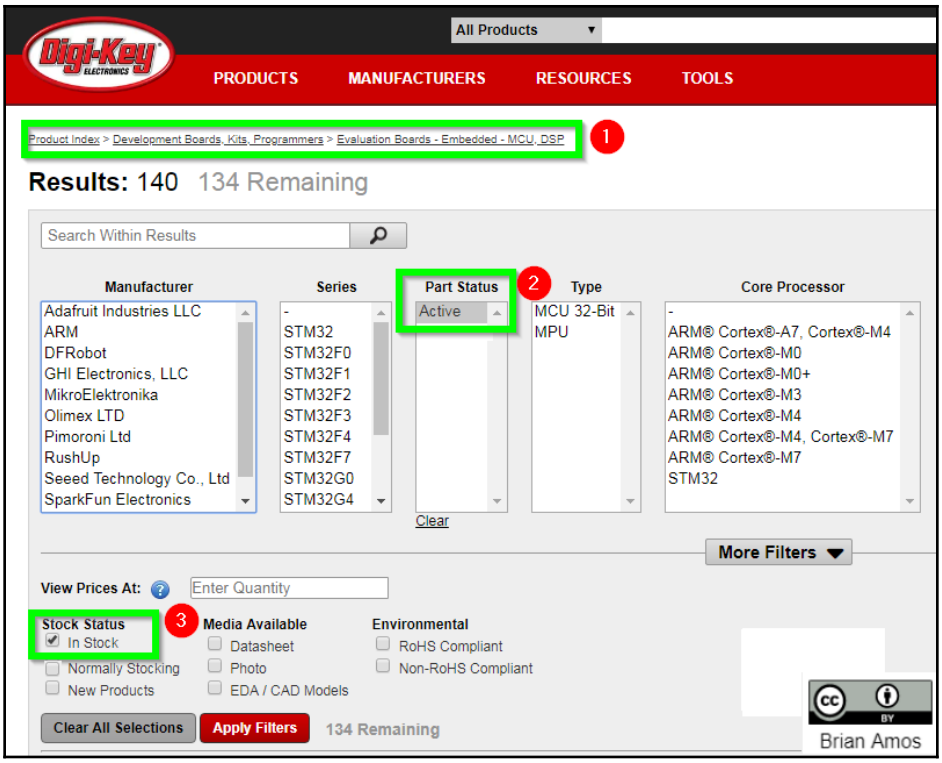

Binary Semaphore
Mutual Exclusion Attempt
binary semaphore
Task A
waiting on
smphr
2
3
Task B
preempts task C
(higher priority)
Task A
FINALLY
takes
smphr
5
1
Task C
takes smphr
4
Task C
running again
gives smphr
DONE
Task Priority
Time
Brian Amos

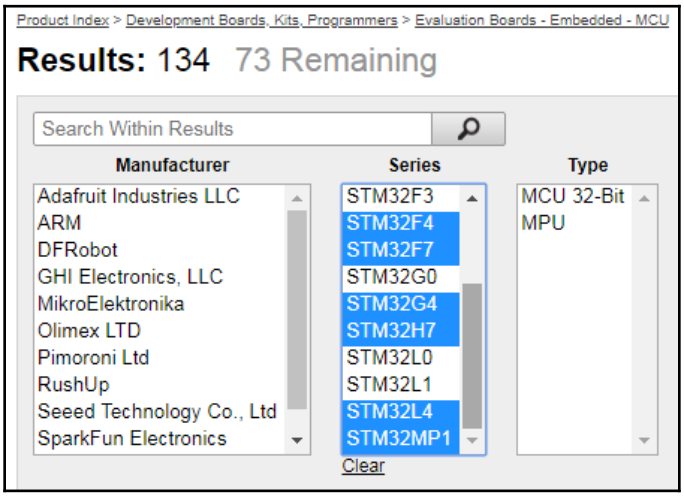

Mutex
Priority Inversion
mutex
1
4
Task A
waiting on
smphr
Task C
running again
gives smphr
DONE
Task A
takes
smphr
DONE
Task C inherits
priority of Task A
2
3
Task B
runs
Task C
takes smphr
Task C
priority
returned
to normal
5
Task Priority
Time
Brian Amos

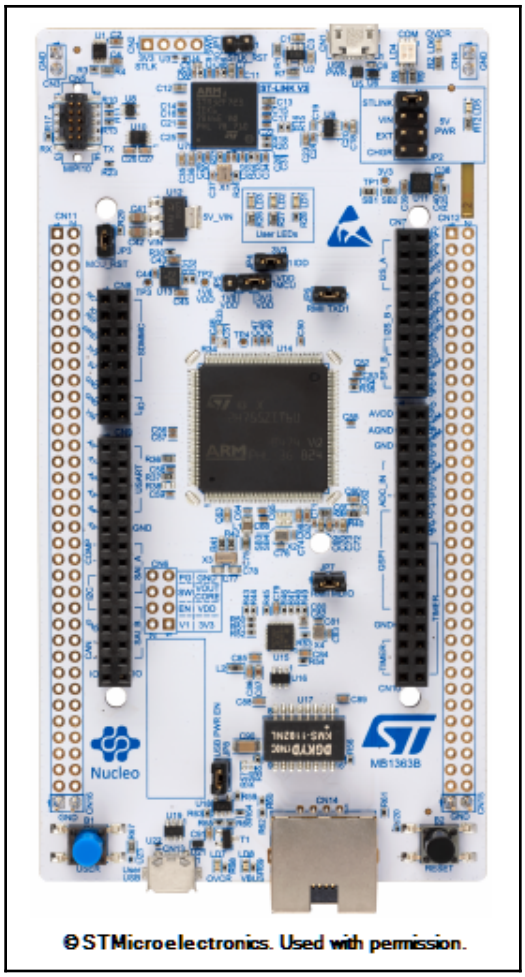
Chapter 4: Selecting the Right MCU




DevBoard Differences
Character LCD
STM32
Touch Sensor
DevBoard A
Brian Amos
USB
OTG
CAN
UART
Audio
STM32
QSPI
Flash
RAM
Ethernet
SD card
TFT
Touchscreen
POT
POT
Dev Board B

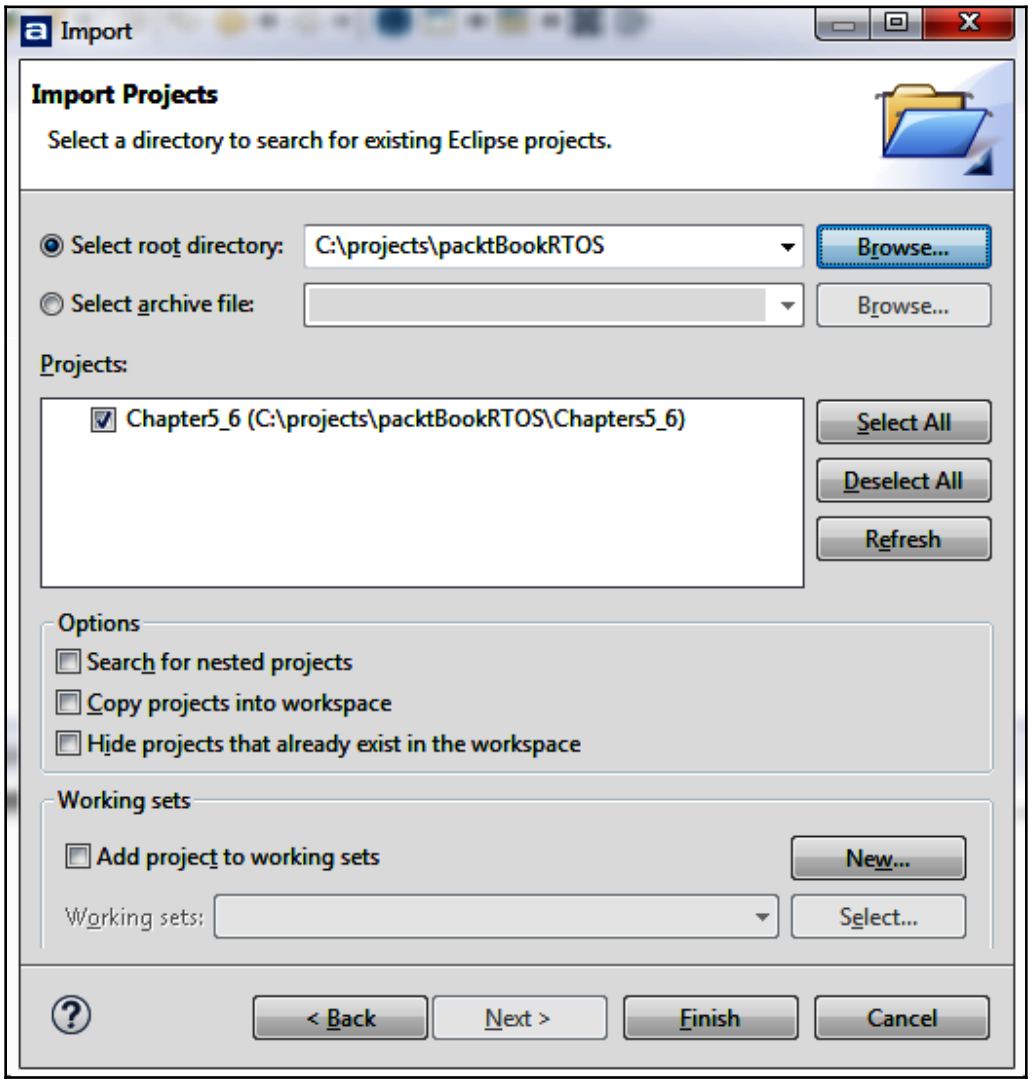

All Products
PRODUCTS
MANUFACTURERS
RESOURCES
TOOLS
Product Index > Development Boards, Kits, Programmers > Evaluation Boards - Embedded - MCU, DSP
1
Results: 140 134 Remaining
Search Within Results
Manufacturer
Adafruit Industries LLC
ARM
DFRobot
GHI Electronics, LLC
MikroElektronika
Olimex LTD
Pimoroni Ltd
RushUp
Seeed Technology Co., Ltd
SparkFun Electronics
Series
-
STM32
STM32F0
STM32F1
STM32F2
STM32F3
STM32F4
STM32F7
STM32G0
STM32G4
Part Status
Active
2
Type
MCU 32-Bit
MPU
Core Processor
-
ARM® Cortex®-A7, Cortex®-M4
ARM® Cortex®-M0
ARM® Cortex®-M0+
ARM® Cortex®-M3
ARM® Cortex®-M4
ARM® Cortex®-M4, Cortex®-M7
ARM® Cortex®-M7
STM32
Clear
More Filters
View Prices At:
Enter Quantity
Stock Status
In Stock
Normally Stocking
New Products
3
Media Available
Datasheet
Photo
EDA / CAD Models
Environmental
RoHS Compliant
Non-RoHS Compliant
Clear All Selections
Apply Filters
134 Remaining
Brian Amos

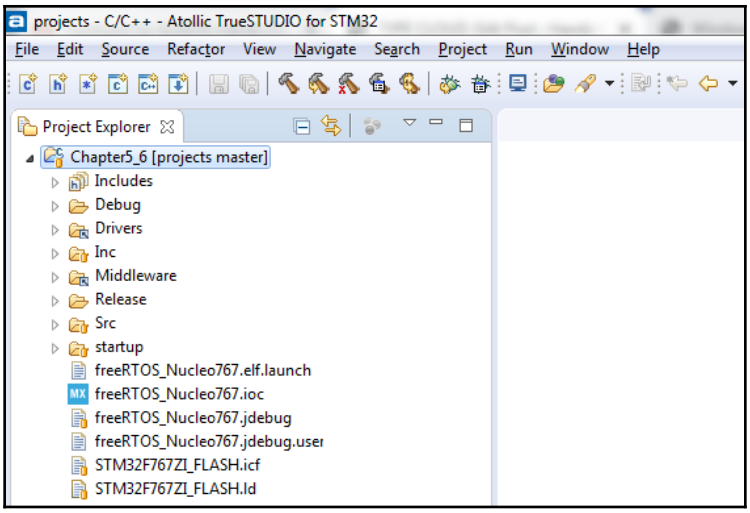

Product Index > Development Boards, Kits, Programmers > Evaluation Boards - Embedded - MCU
Results: 134 73 Remaining
Search Within Results
Manufacturer
Adafruit Industries LLC
ARM
DFRobot
GHI Electronics, LLC
MikroElektronika
Olimex LTD
Pimoroni Ltd
RushUp
Seeed Technology Co., Ltd
SparkFun Electronics
Series
STM32F3
STM32F4
STM32F7
STM32G0
STM32G4
STM32H7
STM32L0
STM32L1
STM32L4
STM32MP1
Type
MCU 32-Bit
MPU
Clear

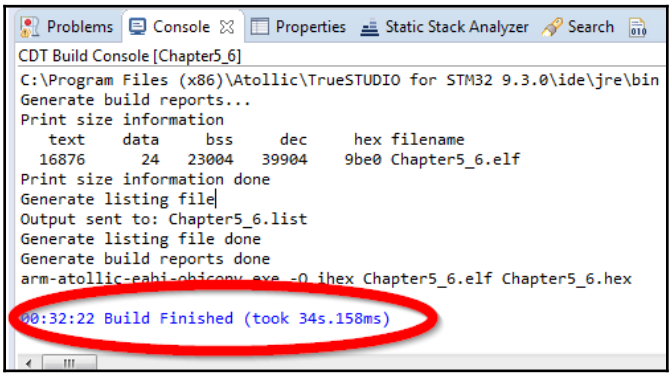


© STMicroelectronics. Used with permission.

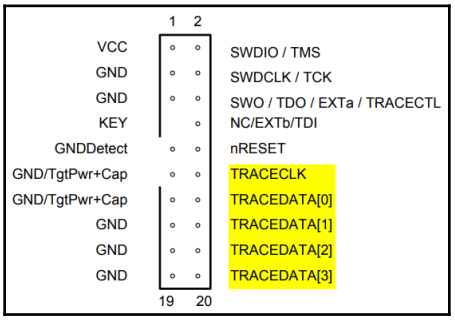
Chapter 5: Selecting an IDE

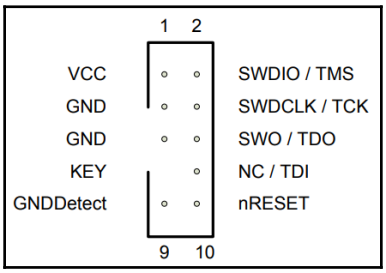


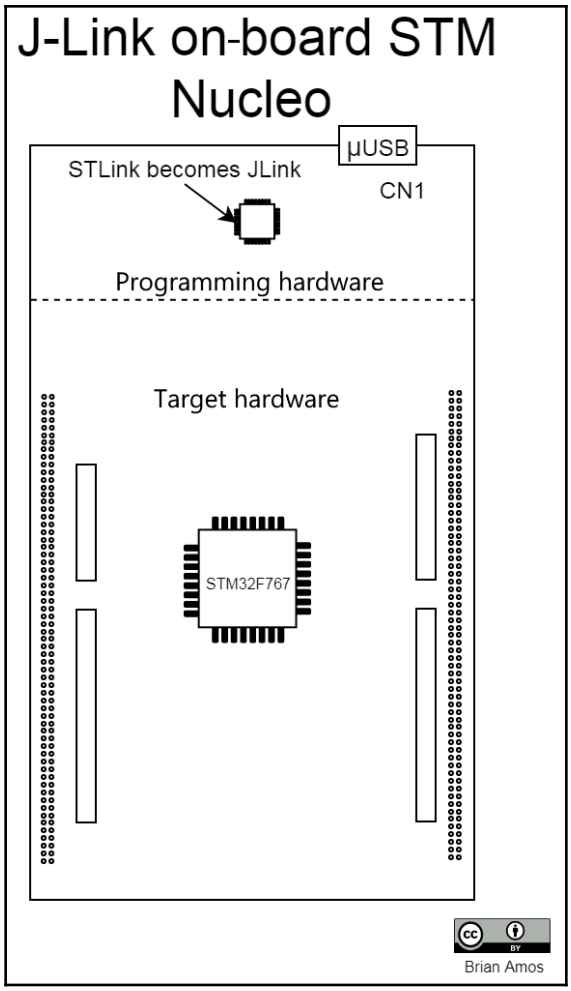

projects - C/C++ - Atollic TrueSTUDIO for STM32
File Edit Source Refactor View Navigate Search Project Run Window Help
Project Explorer
Chapter5_6 [projects master]
Includes
Debug
Drivers
Inc
Middleware
Release
Src
startup
freeRTOS_Nucleo767.elf.launch
freeRTOS_Nucleo767.ioc
freeRTOS_Nucleo767.jdebug
freeRTOS_Nucleo767.jdebug.user
STM32F767ZI_FLASH.icf
STM32F767ZI_FLASH.ld

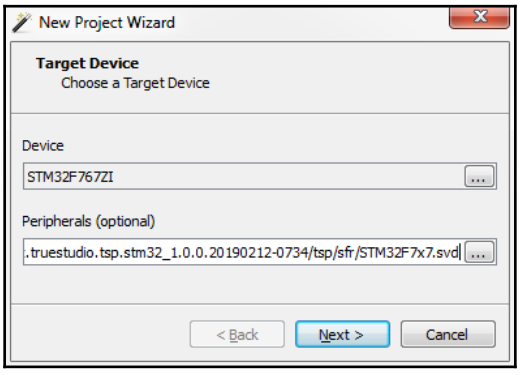

Problems
Console
Properties
Static Stack Analyzer
Search
CDT Build Console [Chapter5_6]
C:\Program Files (x86)\Atollic\TrueSTUDIO for STM32 9.3.0\ide\jre\bin
Generate build reports...
Print size information
text data bss dec hex filename
16876 24 23004 39904 9be0 Chapter5_6.elf
Print size information done
Generate listing file
Output sent to: Chapter5_6.list
Generate listing file done
Generate build reports done
arm-atollic-eabi-objcopy.exe -O ihex Chapter5_6.elf Chapter5_6.hex
00:32:22 Build Finished (took 34s.158ms)

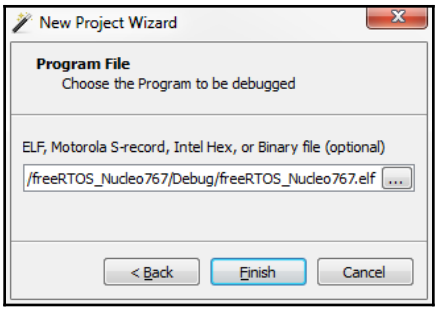
Chapter 6: Debugging Tools for Real-Time Systems

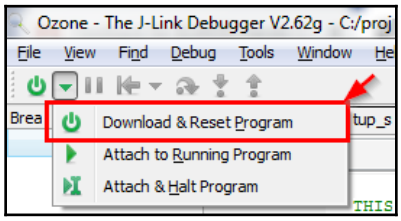





J-Link on-board STM Nucleo
µUSB
CN1
STLink becomes JLink
Programming hardware
Target hardware
STM32F767
Brian Amos

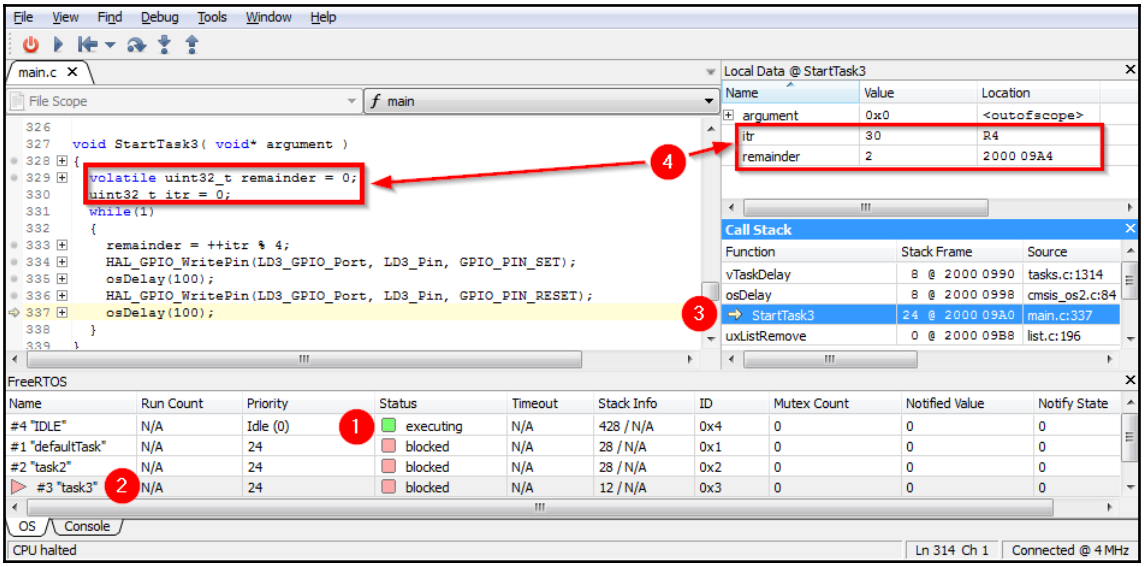

New Project Wizard
Target Device
Choose a Target Device
Device
STM32F767ZI
Peripherals (optional)
.truestudio.tsp.stm32_1.0.0.20190212-0734/tsp/sfr/STM32F7x7.svd
< Back
Next >
Cancel

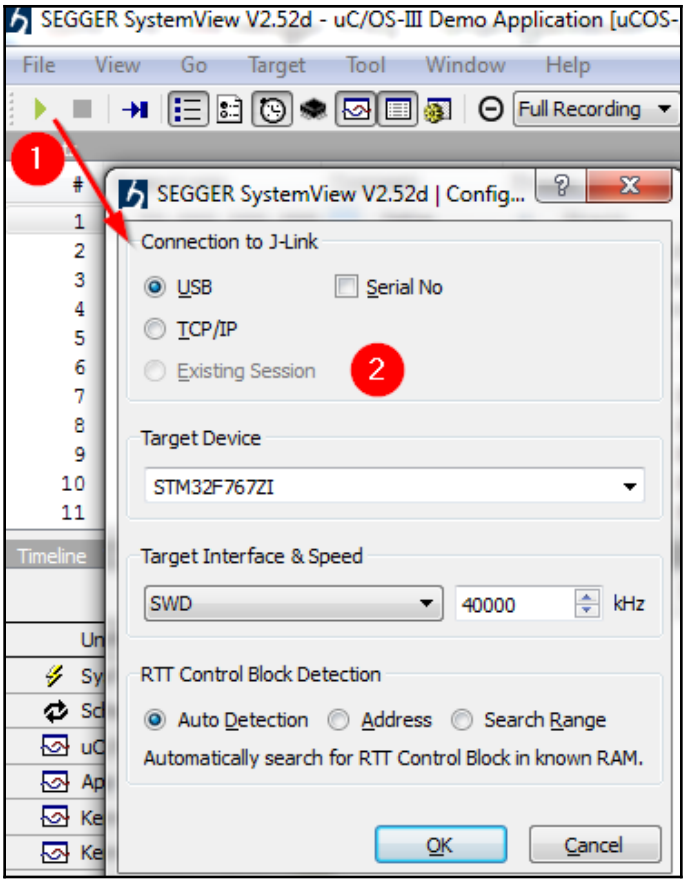


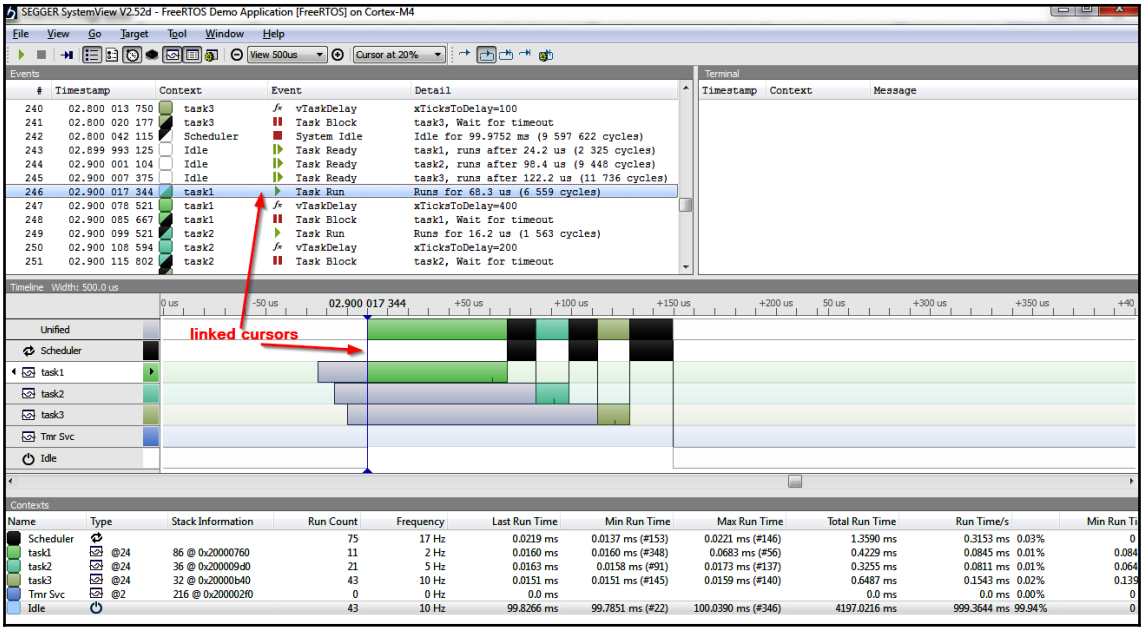


FreeRTOS

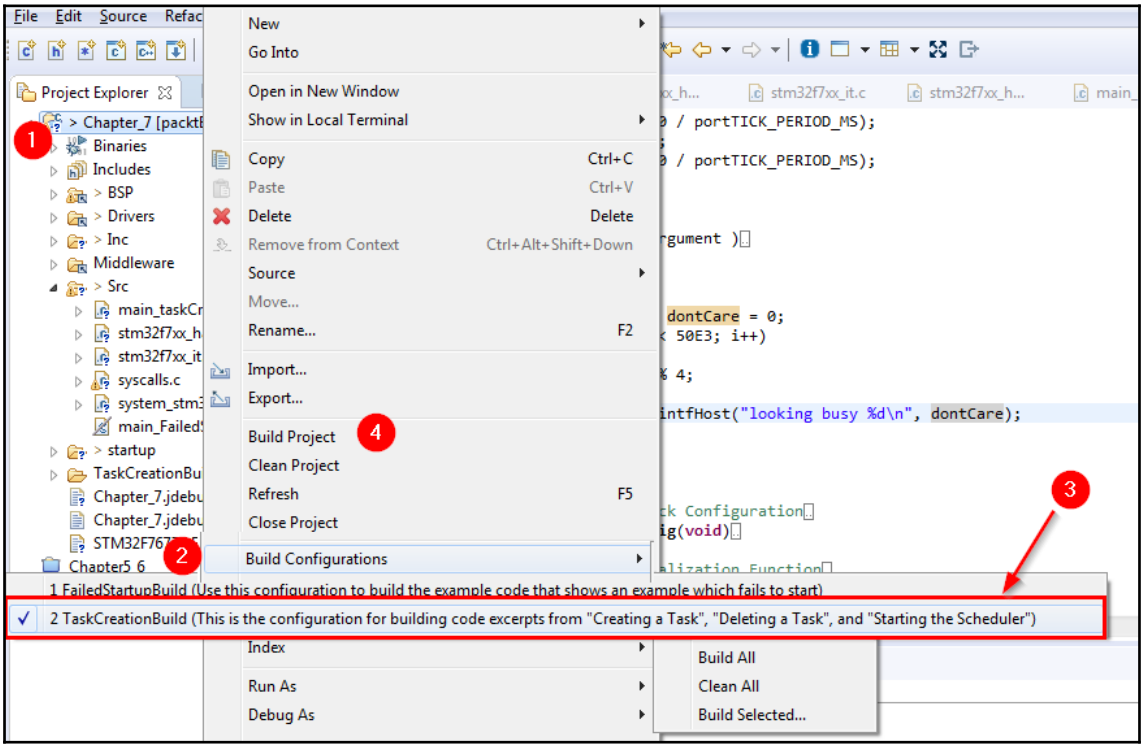
Name	Run Count	Priority	Status	Timeout	Stack Info	ID	Mutex Count	Notified Value	Notify State
#4 "IDLE"	N/A	Idle (0)	executing	N/A	428 / N/A	0x4	0	0	0
#1 "defaultTask"	N/A	24	blocked	N/A	28 / N/A	0x1	0	0	0
#2 "task2"	N/A	24	blocked	N/A	28 / N/A	0x2	0	0	0
#3 "task3"	N/A	24	blocked	N/A	28 / N/A	0x3	0	0	0
#5 "Tmr Svc"	N/A	Normal (2)	suspended	N/A	876 / N/A	0x5	0	0	0

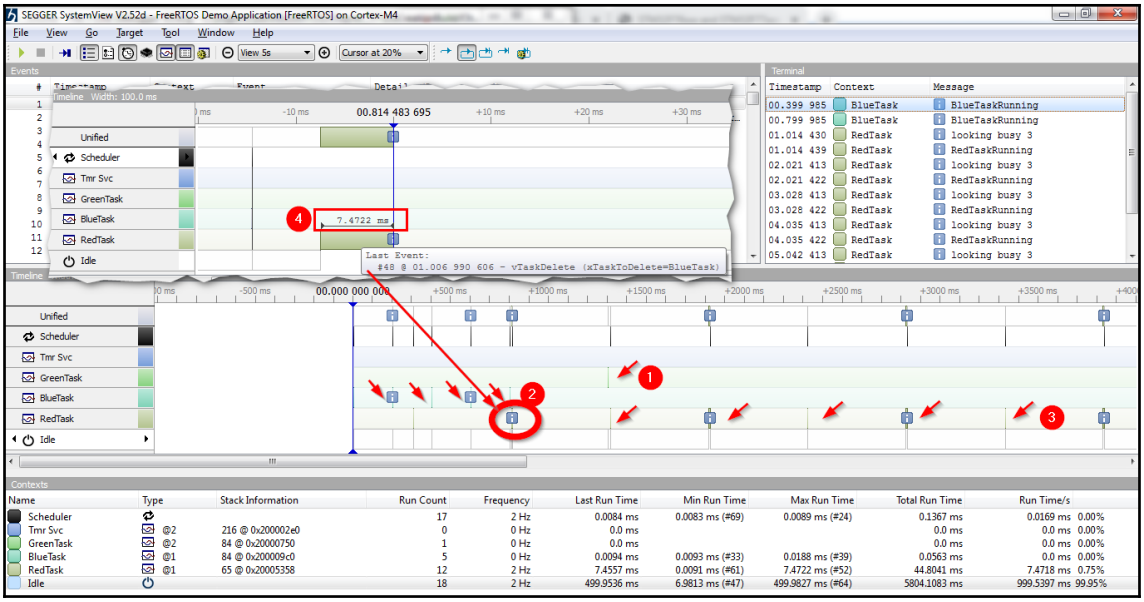


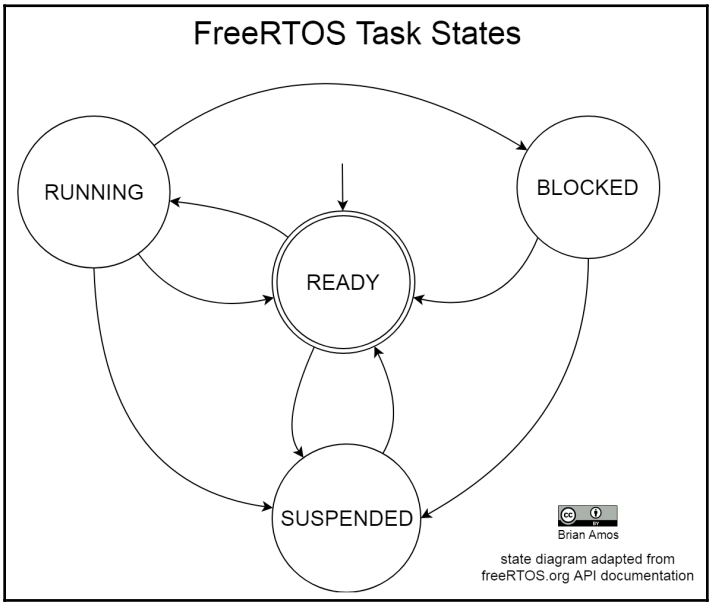

SEGGER SystemView V2.52d - uC/OS-III Demo Application [uCOS-
File
View
Go
Target
Tool
Window
Help
Full Recording
1
SEGGER SystemView V2.52d | Config...
Connection to J-Link
USB
Serial No
TCP/IP
Existing Session
2
Target Device
STM32F767ZI
Target Interface & Speed
SWD
40000
kHz
RTT Control Block Detection
Auto Detection
Address
Search Range
Automatically search for RTT Control Block in known RAM.
OK
Cancel
Timeline

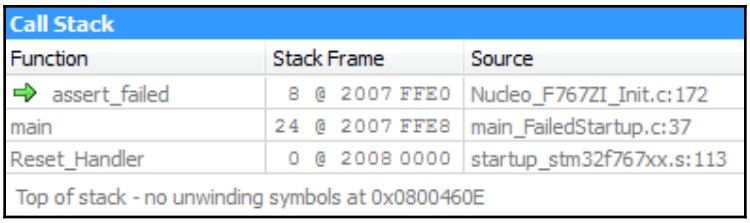

SEGGER SystemView V2.52d - FreeRTOS Demo Application [FreeRTOS] on Cortex-M4
File View Go Target Tool Window Help
Events
Terminal
Timestamp Context Message
Timestamp Context Event Detail
240 02.800 013 750 task3 vTaskDelay xTicksToDelay=100
241 02.800 020 177 task3 Task Block task3, Wait for timeout
242 02.800 042 115 Scheduler System Idle Idle for 99.9752 ms (9 597 622 cycles)
243 02.899 993 125 Idle Task Ready task1, runs after 24.2 us (2 325 cycles)
244 02.900 001 104 Idle Task Ready task2, runs after 98.4 us (9 448 cycles)
245 02.900 007 375 Idle Task Ready task3, runs after 122.2 us (11 736 cycles)
246 02.900 017 344 task1 Task Run Runs for 68.3 us (6 559 cycles)
247 02.900 078 521 task1 vTaskDelay xTicksToDelay=400
248 02.900 085 667 task1 Task Block task1, Wait for timeout
249 02.900 099 521 task2 Task Run Runs for 16.2 us (1 563 cycles)
250 02.900 108 594 task2 vTaskDelay xTicksToDelay=200
251 02.900 115 802 task2 Task Block task2, Wait for timeout
Timeline Width: 500.0 us
02.900 017 344
linked cursors
Unified
Scheduler
task1
task2
task3
Tmr Svc
Idle
Contexts
Name Type Stack Information Run Count Frequency Last Run Time Min Run Time Max Run Time Total Run Time Run Time/s
Scheduler 75 17 Hz 0.0219 ms 0.0137 ms (#153) 0.0221 ms (#146) 1.3590 ms 0.3153 ms 0.03%
task1 @24 86 @ 0x20000760 11 2 Hz 0.0160 ms 0.0160 ms (#348) 0.0683 ms (#56) 0.4229 ms 0.0845 ms 0.01%
task2 @24 36 @ 0x200009d0 21 5 Hz 0.0163 ms 0.0158 ms (#91) 0.0173 ms (#137) 0.3255 ms 0.0811 ms 0.01%
task3 @24 32 @ 0x20000b40 43 10 Hz 0.0151 ms 0.0151 ms (#145) 0.0159 ms (#140) 0.6487 ms 0.1543 ms 0.02%
Tmr Svc @2 216 @ 0x200002f0 0 0 Hz 0.0 ms 0.0 ms 0.00%
Idle 43 10 Hz 99.8266 ms 99.7851 ms (#22) 100.0390 ms (#346) 4197.0216 ms 999.3644 ms 99.94%

Chapter 7: The FreeRTOS Scheduler

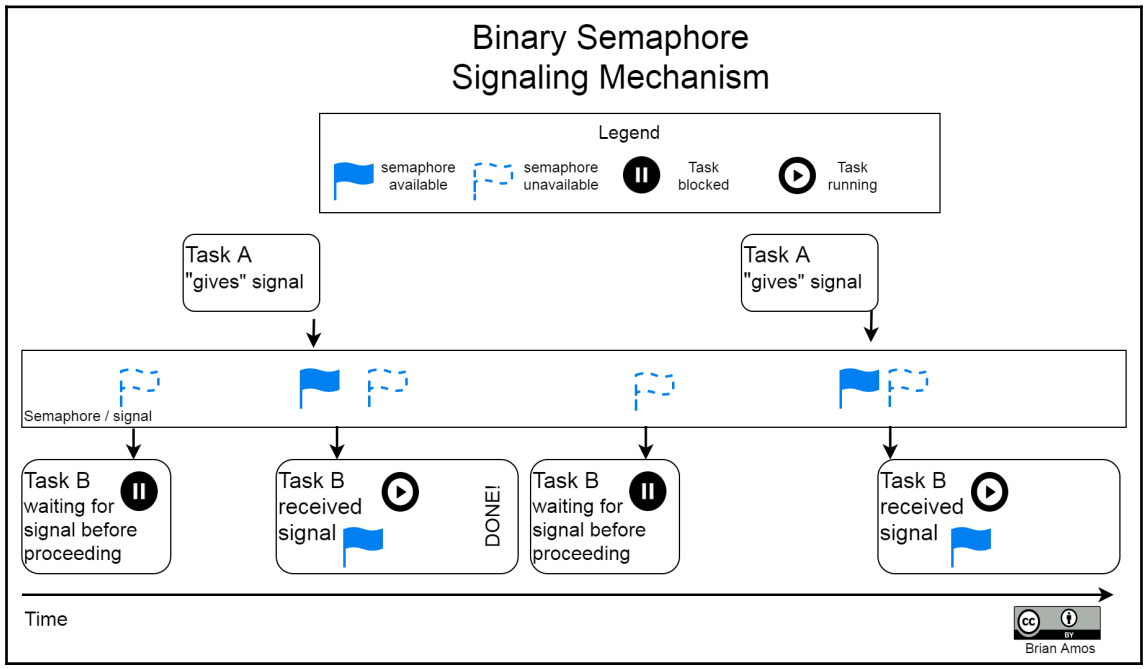


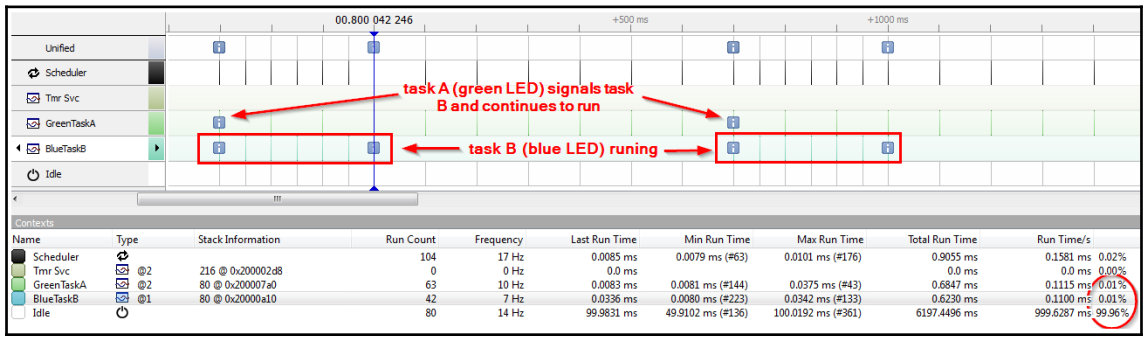

SEGGER SystemView V2.52d - FreeRTOS Demo Application [FreeRTOS] on Cortex-M4
File View Go Target Tool Window Help
Events
Terminal
Timestamp Context Message
00.399 985 BlueTask BlueTaskRunning
00.799 985 BlueTask BlueTaskRunning
01.014 430 RedTask looking busy 3
01.014 439 RedTask RedTaskRunning
02.021 413 RedTask looking busy 3
02.021 422 RedTask RedTaskRunning
03.028 413 RedTask looking busy 3
03.028 422 RedTask RedTaskRunning
04.035 413 RedTask looking busy 3
04.035 422 RedTask RedTaskRunning
05.042 413 RedTask looking busy 3
00.814 483 695
Unified
Scheduler
Tmr Svc
GreenTask
BlueTask
RedTask
Idle
7.4722 ms
Last Event:
#48 @ 01.006 990 606 - vTaskDelete (xTaskToDelete=BlueTask)
Timeline
00.000 000 000
Contexts
Name Type Stack Information Run Count Frequency Last Run Time Min Run Time Max Run Time Total Run Time Run Time/s
Scheduler 17 2 Hz 0.0084 ms 0.0083 ms (#69) 0.0089 ms (#24) 0.1367 ms 0.0169 ms 0.00%
Tmr Svc @2 216 @ 0x200002e0 0 0 Hz 0.0 ms 0.0 ms 0.00%
GreenTask @2 84 @ 0x20000750 1 0 Hz 0.0 ms 0.0 ms 0.00%
BlueTask @1 84 @ 0x200009c0 5 0 Hz 0.0094 ms 0.0093 ms (#33) 0.0188 ms (#39) 0.0563 ms 0.0 ms 0.00%
RedTask @1 65 @ 0x20005358 12 2 Hz 7.4557 ms 0.0091 ms (#61) 7.4722 ms (#52) 44.8041 ms 7.4718 ms 0.75%
Idle 18 2 Hz 499.9536 ms 6.9813 ms (#47) 499.9827 ms (#64) 5804.1083 ms 999.5397 ms 99.95%

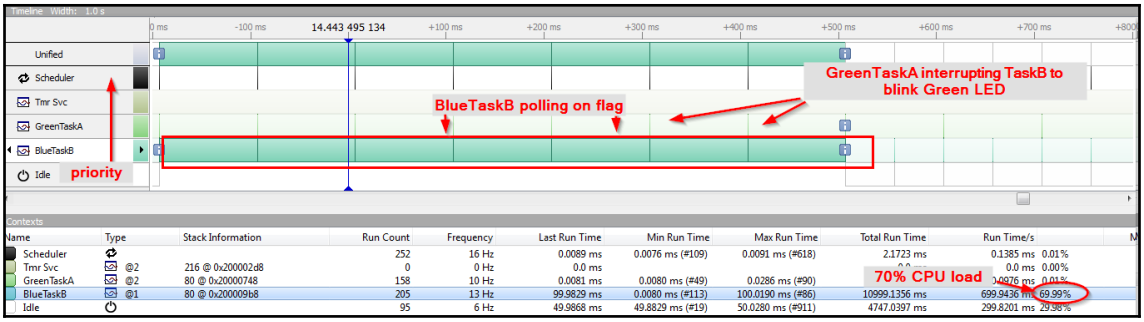

FreeRTOS Task States
RUNNING
BLOCKED
READY
SUSPENDED
Brian Amos
state diagram adapted from
freeRTOS.org API documentation

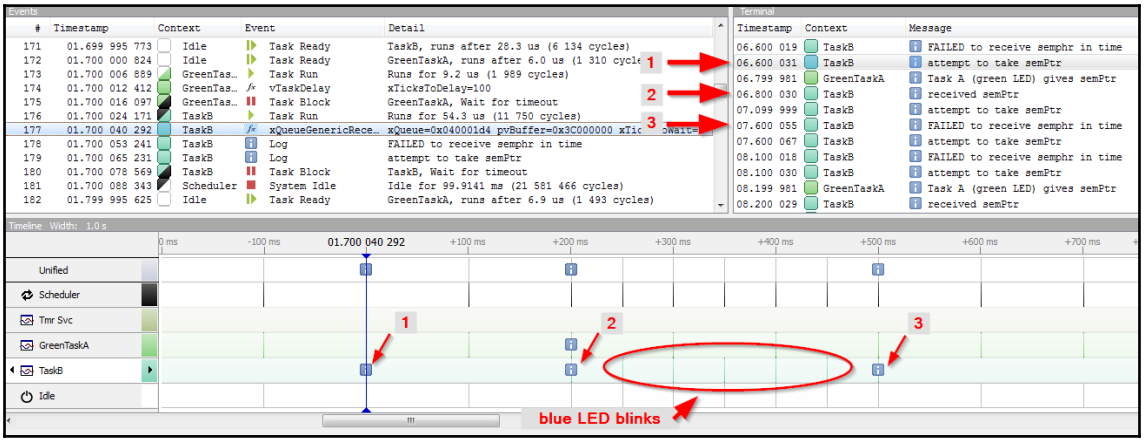

Call Stack
Function
Stack Frame
Source
assert_failed
8 @ 2007 FFE0
Nucleo_F767ZI_Init.c:172
main
24 @ 2007 FFE8
main_FailedStartup.c:37
Reset_Handler
0 @ 2008 0000
startup_stm32f767xx.s:113
Top of stack - no unwinding symbols at 0x0800460E

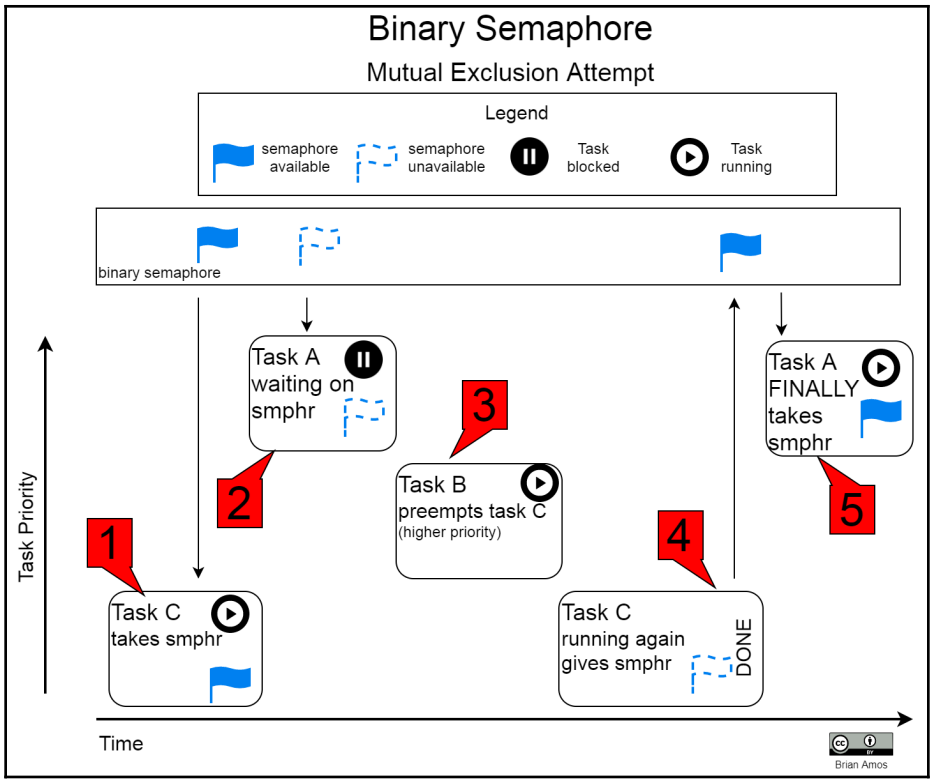
Chapter 8: Protecting Data and Synchronizing Tasks

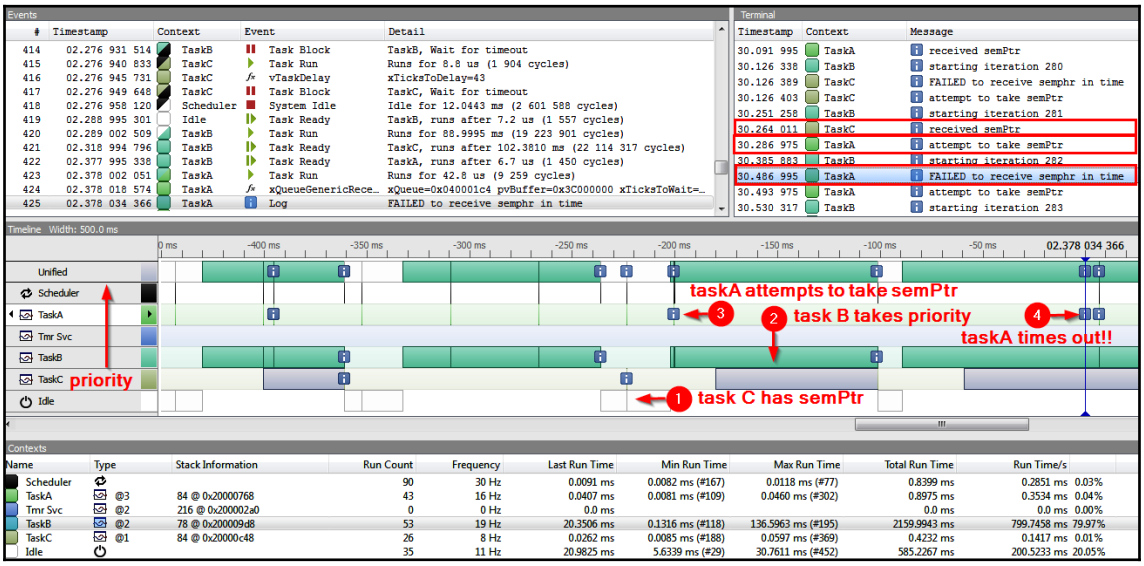


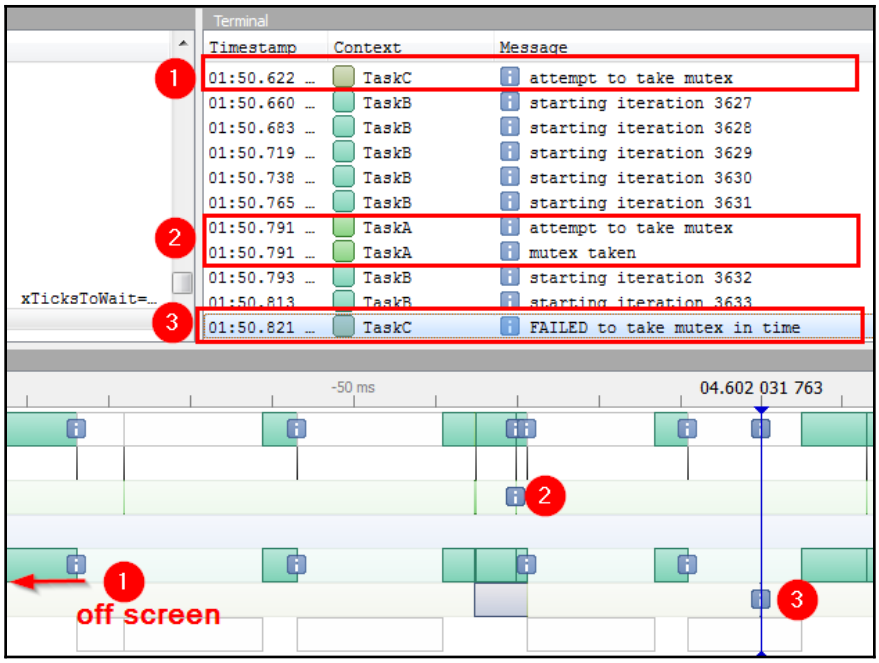


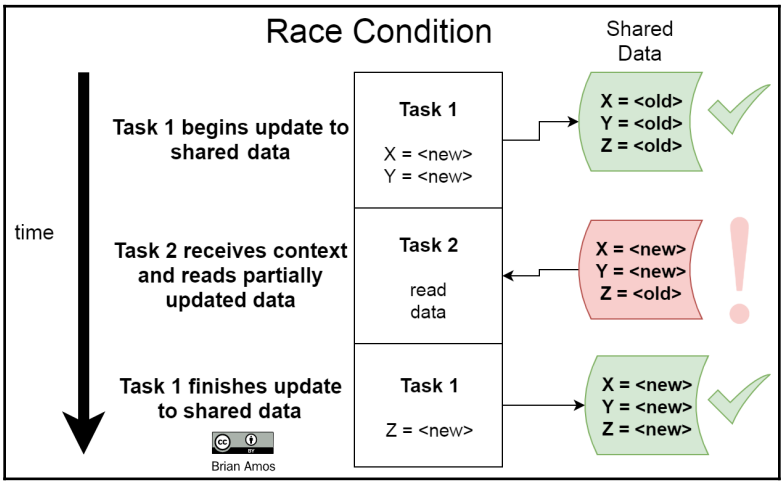

Timeline Width: 1.0 s
14.443 495 134
Unified
Scheduler
Tmr Svc
GreenTaskA
BlueTaskB
Idle
priority
BlueTaskB polling on flag
GreenTaskA interrupting TaskB to blink Green LED
70% CPU load
Contexts
Name Type Stack Information Run Count Frequency Last Run Time Min Run Time Max Run Time Total Run Time Run Time/s
Scheduler 252 16 Hz 0.0089 ms 0.0076 ms (#109) 0.0091 ms (#618) 2.1723 ms 0.1385 ms 0.01%
Tmr Svc @2 216 @ 0x200002d8 0 0 Hz 0.0 ms 0.0 ms 0.00%
GreenTaskA @2 80 @ 0x20000748 158 10 Hz 0.0081 ms 0.0080 ms (#49) 0.0286 ms (#90) 0.0976 ms 0.01%
BlueTaskB @1 80 @ 0x200009b8 205 13 Hz 99.9829 ms 0.0080 ms (#113) 100.0190 ms (#86) 10999.1356 ms 699.9436 ms 69.99%
Idle 95 6 Hz 49.9868 ms 49.8829 ms (#19) 50.0280 ms (#911) 4747.0397 ms 299.8201 ms 29.98%


Events
Timestamp Context Event Detail
171 01.699 995 773 Idle Task Ready TaskB, runs after 28.3 us (6 134 cycles)
172 01.700 000 824 Idle Task Ready GreenTaskA, runs after 6.0 us (1 310 cycle
173 01.700 006 889 GreenTas… Task Run Runs for 9.2 us (1 989 cycles)
174 01.700 012 412 GreenTas… vTaskDelay xTicksToDelay=100
175 01.700 016 097 GreenTas… Task Block GreenTaskA, Wait for timeout
176 01.700 024 171 TaskB Task Run Runs for 54.3 us (11 750 cycles)
177 01.700 040 292 TaskB xQueueGenericRece… xQueue=0x040001d4 pvBuffer=0x3C000000 xTic
178 01.700 053 241 TaskB Log FAILED to receive semphr in time
179 01.700 065 231 TaskB Log attempt to take semPtr
180 01.700 078 569 TaskB Task Block TaskB, Wait for timeout
181 01.700 088 343 Scheduler System Idle Idle for 99.9141 ms (21 581 466 cycles)
182 01.799 995 625 Idle Task Ready GreenTaskA, runs after 6.9 us (1 493 cycles)
1
2
3
Terminal
Timestamp Context Message
06.600 019 TaskB FAILED to receive semphr in time
06.600 031 TaskB attempt to take semPtr
06.799 981 GreenTaskA Task A (green LED) gives semPtr
06.800 030 TaskB received semPtr
07.099 999 TaskB attempt to take semPtr
07.600 055 TaskB FAILED to receive semphr in time
07.600 067 TaskB attempt to take semPtr
08.100 018 TaskB FAILED to receive semphr in time
08.100 030 TaskB attempt to take semPtr
08.199 981 GreenTaskA Task A (green LED) gives semPtr
08.200 029 TaskB received semPtr
Timeline Width: 1.0 s
01.700 040 292
Unified
Scheduler
Tmr Svc
GreenTaskA
TaskB
Idle
1
2
3
blue LED blinks


Binary Semaphore
Mutual Exclusion Attempt
Legend
semaphore available
semaphore unavailable
Task blocked
Task running
binary semaphore
Task Priority
Time
Task A waiting on smphr
Task B preempts task C (higher priority)
Task C takes smphr
Task C running again gives smphr
DONE
Task A FINALLY takes smphr
1
2
3
4
5
Brian Amos

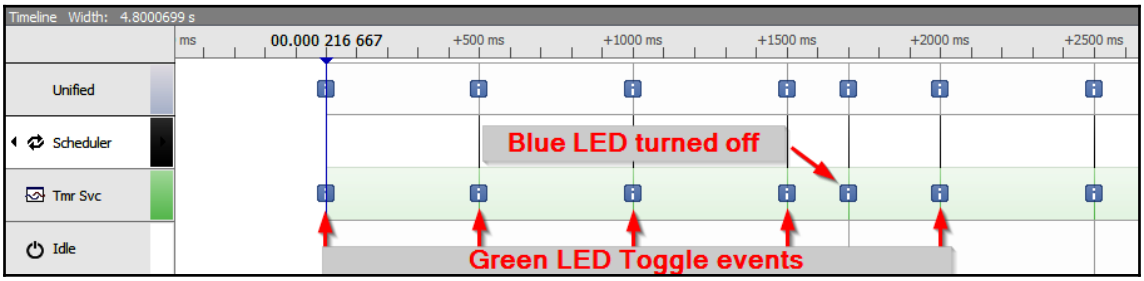

taskA attempts to take semPtr
task B takes priority
taskA times out!!
task C has semPtr
priority


Terminal
Timestamp
Context
Message
01:50.622 … TaskC attempt to take mutex
01:50.660 … TaskB starting iteration 3627
01:50.683 … TaskB starting iteration 3628
01:50.719 … TaskB starting iteration 3629
01:50.738 … TaskB starting iteration 3630
01:50.765 … TaskB starting iteration 3631
01:50.791 … TaskA attempt to take mutex
01:50.791 … TaskA mutex taken
01:50.793 … TaskB starting iteration 3632
xTicksToWait=…
01:50.821 … TaskC FAILED to take mutex in time
-50 ms
04.602 031 763
1
2
3
off screen

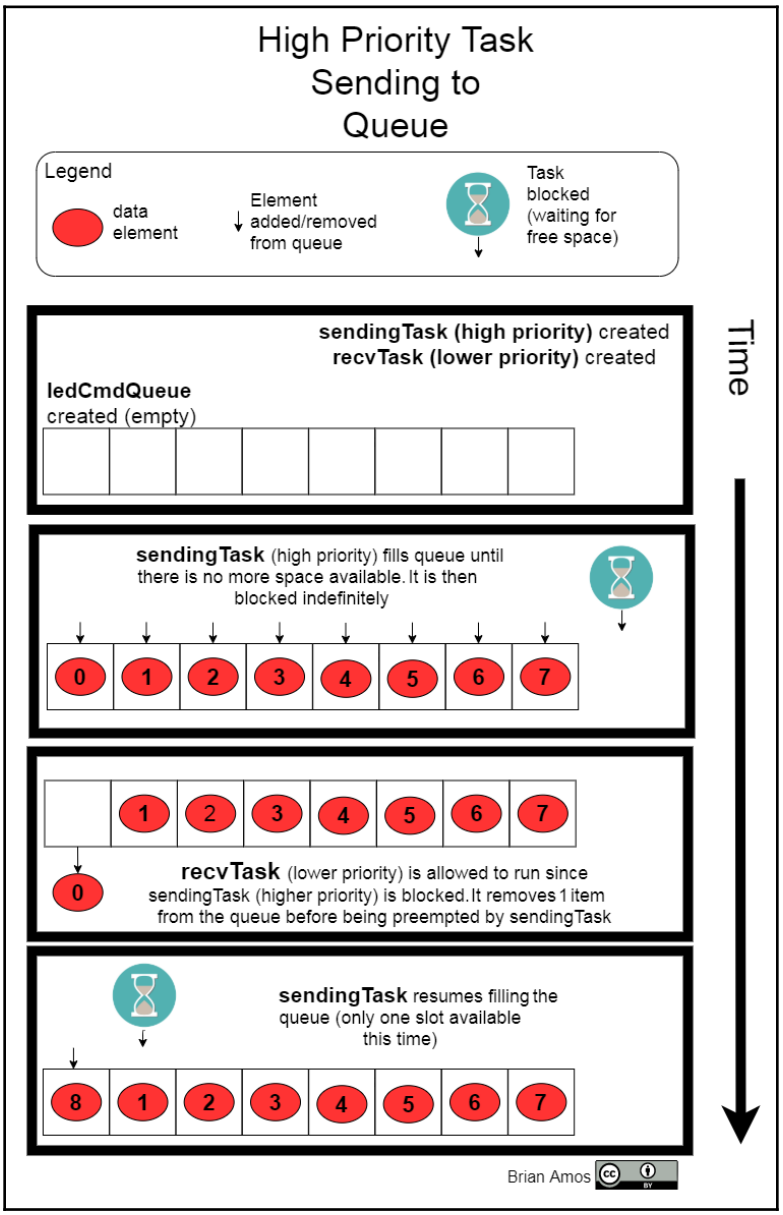

Race Condition
Shared Data
time
Task 1 begins update to shared data
Task 1
X = <new>
Y = <new>
X = <old>
Y = <old>
Z = <old>
Task 2 receives context and reads partially updated data
Task 2
read data
X = <new>
Y = <new>
Z = <old>
Task 1 finishes update to shared data
Task 1
Z = <new>
X = <new>
Y = <new>
Z = <new>
Brian Amos

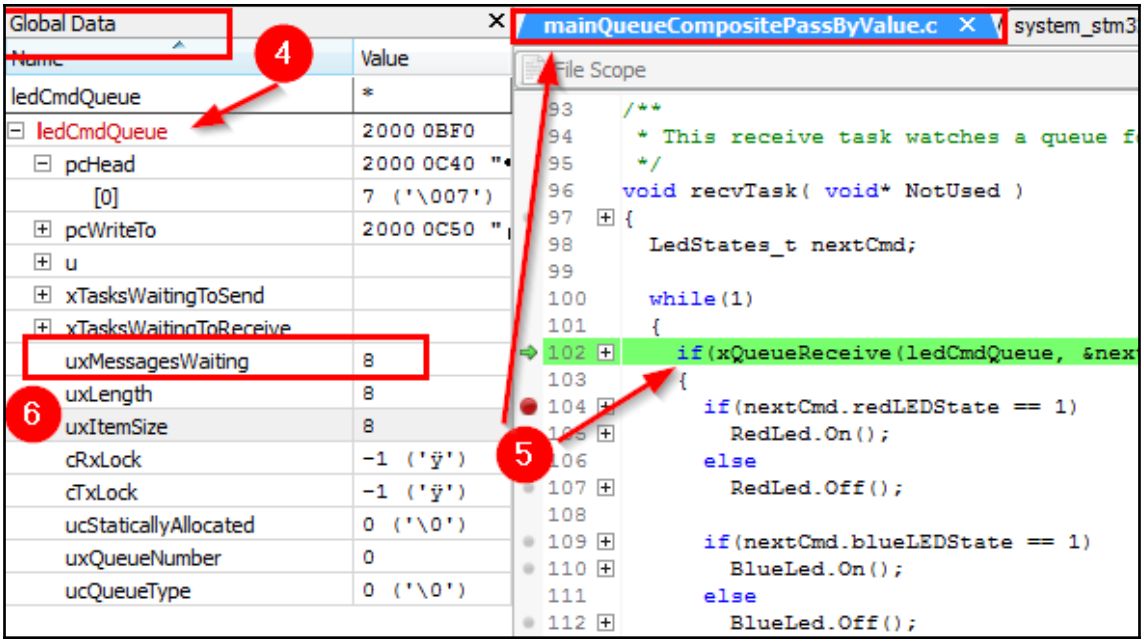
Terminal

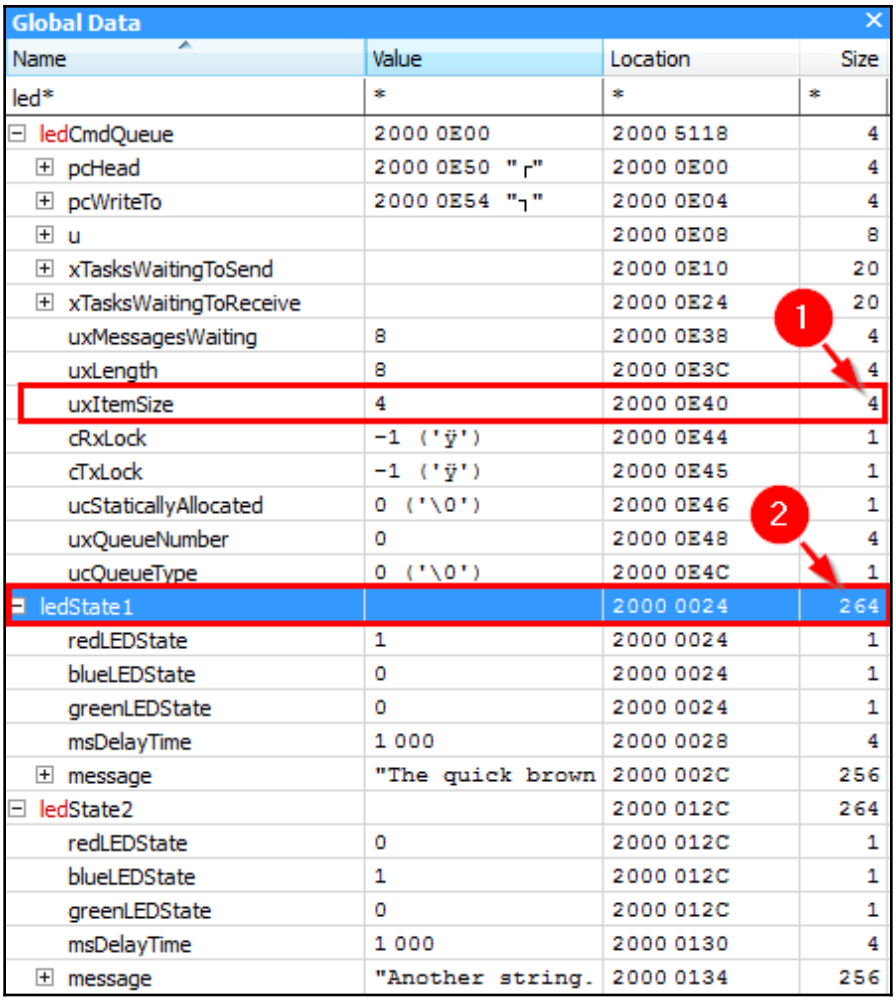
Timestamp	Context	Message
00.500 159	Tmr Svc	toggle Green LED
00.999 998	Tmr Svc	toggle Green LED
01.499 998	Tmr Svc	toggle Green LED
01.999 999	Tmr Svc	toggle Green LED
02.199 996	Tmr Svc	blue LED off
02.499 998	Tmr Svc	toggle Green LED
02.999 998	Tmr Svc	toggle Green LED
03.499 998	Tmr Svc	toggle Green LED
04.000 148	Tmr Svc	toggle Green LED
04.500 000	Tmr Svc	toggle Green LED

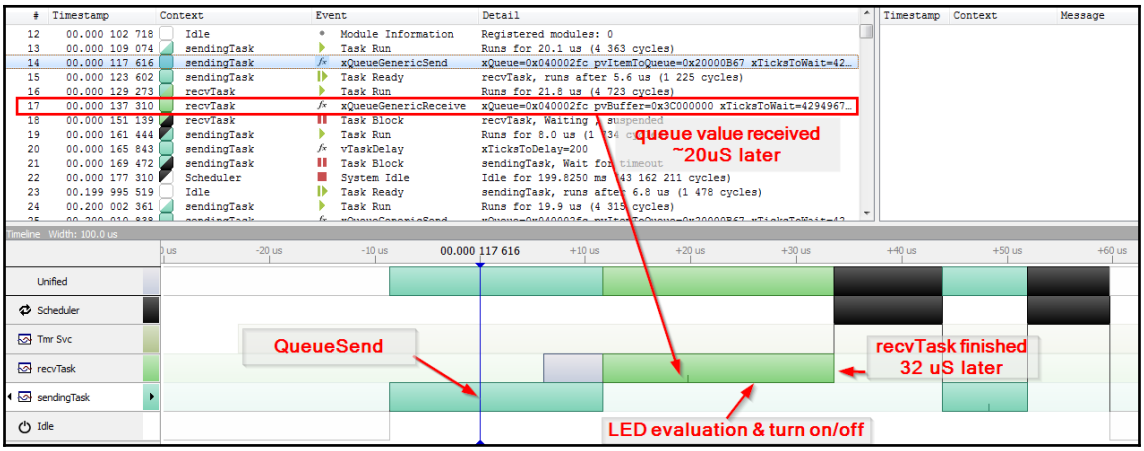


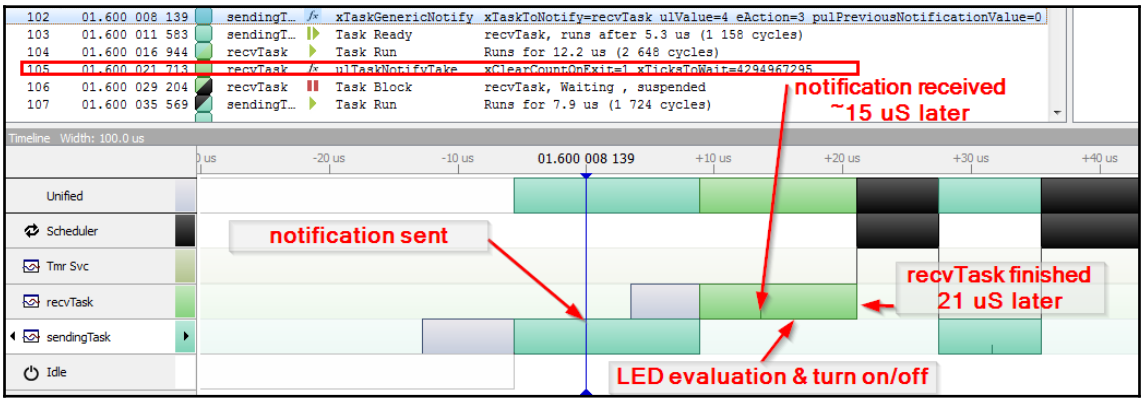
Chapter 9: Intertask Communication



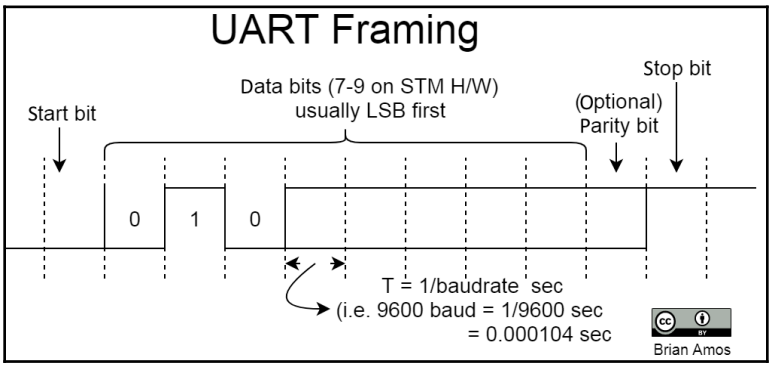

Global Data
Name
Value
ledCmdQueue
*
ledCmdQueue
2000 0BF0
pcHead
2000 0C40
[0]
7 ('\007')
pcWriteTo
2000 0C50
u
xTasksWaitingToSend
xTasksWaitingToReceive
uxMessagesWaiting
8
uxLength
8
uxItemSize
8
cRxLock
-1 ('ÿ')
cTxLock
-1 ('ÿ')
ucStaticallyAllocated
0 ('\0')
uxQueueNumber
0
ucQueueType
0 ('\0')
4
5
6
mainQueueCompositePassByValue.c
system_stm3
File Scope
93 /**
94 * This receive task watches a queue f
95 */
96 void recvTask(void* NotUsed)
97 {
98 LedStates_t nextCmd;
99
100 while(1)
101 {
102 if(xQueueReceive(ledCmdQueue, &nex
103 {
104 if(nextCmd.redLEDState == 1)
105 RedLed.On();
106 else
107 RedLed.Off();
108
109 if(nextCmd.blueLEDState == 1)
110 BlueLed.On();
111 else
112 BlueLed.Off();

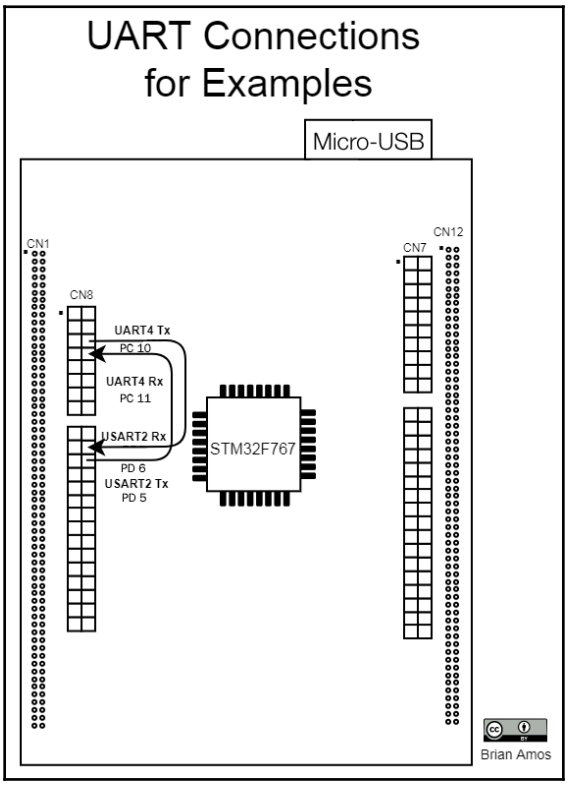

Global Data
Name
Value
Location
Size
led*
*
*
*
ledCmdQueue
2000 0E00
2000 5118
4
pcHead
2000 0E50 "r"
2000 0E00
4
pcWriteTo
2000 0E54 "ר"
2000 0E04
4
u
2000 0E08
8
xTasksWaitingToSend
2000 0E10
20
xTasksWaitingToReceive
2000 0E24
20
uxMessagesWaiting
8
2000 0E38
4
uxLength
8
2000 0E3C
4
uxItemSize
4
2000 0E40
4
cRxLock
-1 ('ÿ')
2000 0E44
1
cTxLock
-1 ('ÿ')
2000 0E45
1
ucStaticallyAllocated
0 ('\0')
2000 0E46
1
uxQueueNumber
0
2000 0E48
4
ucQueueType
0 ('\0')
2000 0E4C
1
ledState1
2000 0024
264
redLEDState
1
2000 0024
1
blueLEDState
0
2000 0024
1
greenLEDState
0
2000 0024
1
msDelayTime
1 000
2000 0028
4
message
"The quick brown
2000 002C
256
ledState2
2000 012C
264
redLEDState
0
2000 012C
1
blueLEDState
1
2000 012C
1
greenLEDState
0
2000 012C
1
msDelayTime
1 000
2000 0130
4
message
"Another string.
2000 0134
256
1
2


queue value received
~20uS later
QueueSend
recvTask finished
32 uS later
LED evaluation & turn on/off


notification received
~15 uS later
notification sent
recvTask finished
21 uS later
LED evaluation & turn on/off

Chapter 10: Drivers and ISRs





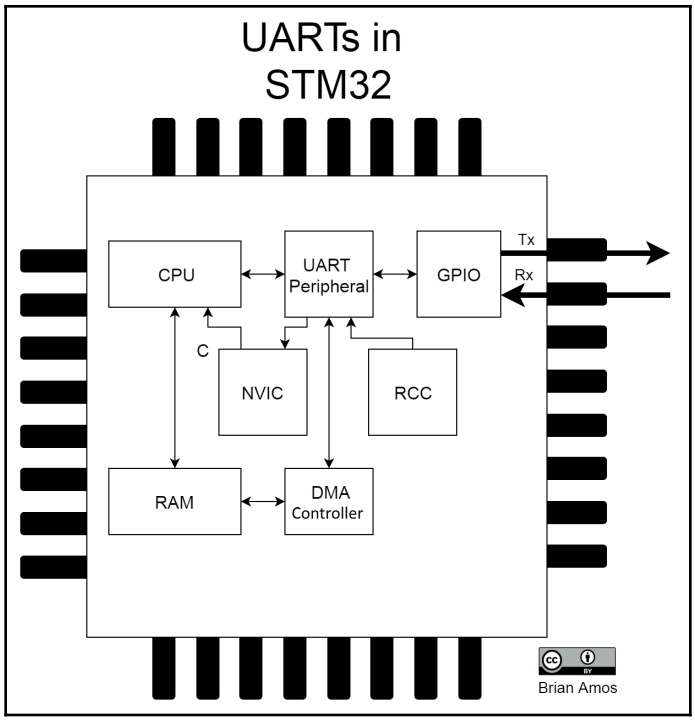
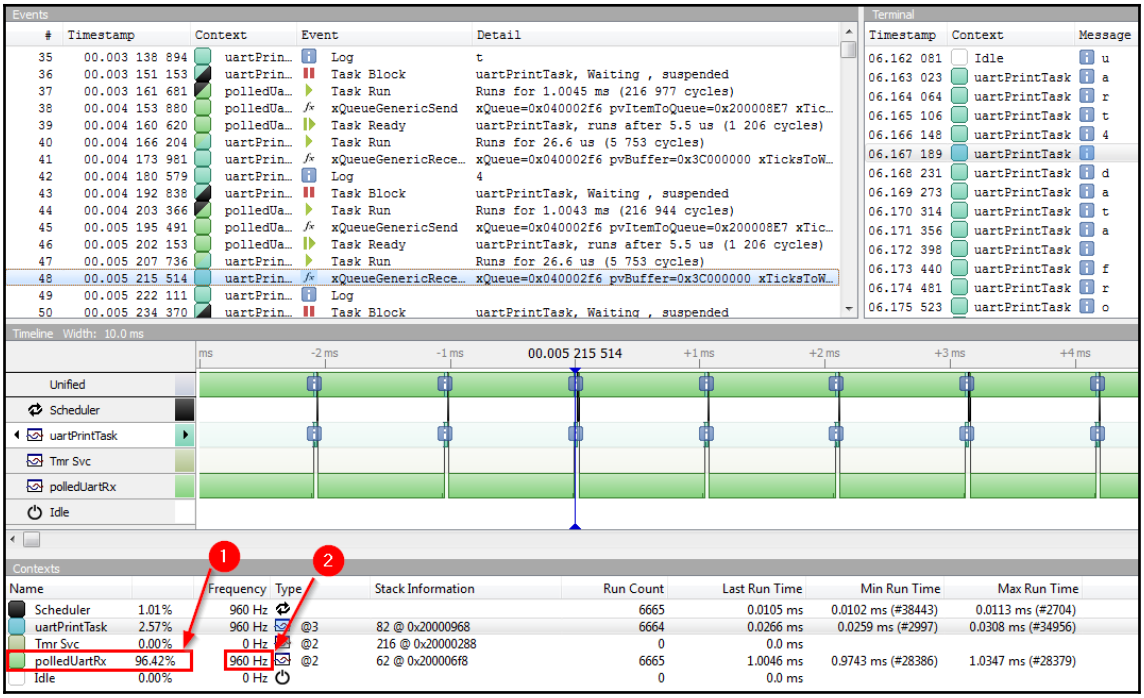


Table 13. STM32F765xx, STM32F767xx, STM32F768Ax and STM32F769xx alternate function mapping (continued)

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF
		SYS	I2C4/UART5/TIM1/2	TIM3/4/5	TIM8/9/10/11/LPTIM1/DFSDM1/CEC	I2C1/2/3/4/USART1/CEC	SPI1/I2S1/SPI2/I2S2/SPI3/I2S3/SPI4/5/6	SPI2/I2S2/SPI3/I2S3/SAI1/I2C4/UART4/DFSDM1	SPI[illegible]/USART1/2/3/UART5/DFSDM1/SPDIF	SPI6/SAI2/USART6/UART4/5/7/8/OTG_FS/SPDIF	CAN1/2/TIM12/13/14/QUADSPI/FMC/LCD	SAI2/QUADSPI/SDMMC2/DFSDM1/OTG2_HS/OTG1_FS/LCD	I2C4/CAN3/SDMMC2/ETH	UAR FMC/ MMC DIOS G2_
Port C	PC11	-	-	-	DFSDM1_DATAIN5	-	-	SPI3_MISO	USART3_RX	UART4_RX	QUADSPI_BK2_NCS	-	-	SDM _D
	PC12	TRACED3	-	-	-	-	-	SPI3_MOSI/I2S3_SD	U[illegible]	UART5_TX	-	-	-	SDM _C
	PC13	-	-	-	-	-	-	-	-	-	-	-	-	-


Events
Timestamp Context Event Detail
35 00.003 138 894 uartPrin… Log t
36 00.003 151 153 uartPrin… Task Block uartPrintTask, Waiting , suspended
37 00.003 161 681 polledUa… Task Run Runs for 1.0045 ms (216 977 cycles)
38 00.004 153 880 polledUa… xQueueGenericSend xQueue=0x040002f6 pvItemToQueue=0x200008E7 xTic…
39 00.004 160 620 polledUa… Task Ready uartPrintTask, runs after 5.5 us (1 206 cycles)
40 00.004 166 204 uartPrin… Task Run Runs for 26.6 us (5 753 cycles)
41 00.004 173 981 uartPrin… xQueueGenericRece… xQueue=0x040002f6 pvBuffer=0x3C000000 xTicksToW…
42 00.004 180 579 uartPrin… Log 4
43 00.004 192 838 uartPrin… Task Block uartPrintTask, Waiting , suspended
44 00.004 203 366 polledUa… Task Run Runs for 1.0043 ms (216 944 cycles)
45 00.005 195 491 polledUa… xQueueGenericSend xQueue=0x040002f6 pvItemToQueue=0x200008E7 xTic…
46 00.005 202 153 polledUa… Task Ready uartPrintTask, runs after 5.5 us (1 206 cycles)
47 00.005 207 736 uartPrin… Task Run Runs for 26.6 us (5 753 cycles)
48 00.005 215 514 uartPrin… xQueueGenericRece… xQueue=0x040002f6 pvBuffer=0x3C000000 xTicksToW…
49 00.005 222 111 uartPrin… Log
50 00.005 234 370 uartPrin… Task Block uartPrintTask, Waiting , suspended
Terminal
Timestamp Context Message
06.162 081 Idle u
06.163 023 uartPrintTask a
06.164 064 uartPrintTask r
06.165 106 uartPrintTask t
06.166 148 uartPrintTask 4
06.167 189 uartPrintTask
06.168 231 uartPrintTask d
06.169 273 uartPrintTask a
06.170 314 uartPrintTask t
06.171 356 uartPrintTask a
06.172 398 uartPrintTask
06.173 440 uartPrintTask f
06.174 481 uartPrintTask r
06.175 523 uartPrintTask o
Timeline Width: 10.0 ms
ms -2 ms -1 ms 00.005 215 514 +1 ms +2 ms +3 ms +4 ms
Unified
Scheduler
uartPrintTask
Tmr Svc
polledUartRx
Idle
1
2
Contexts
Name Frequency Type Stack Information Run Count Last Run Time Min Run Time Max Run Time
Scheduler 1.01% 960 Hz 6665 0.0105 ms 0.0102 ms (#38443) 0.0113 ms (#2704)
uartPrintTask 2.57% 960 Hz @3 82 @ 0x20000968 6664 0.0266 ms 0.0259 ms (#2997) 0.0308 ms (#34956)
Tmr Svc 0.00% 0 Hz @2 216 @ 0x20000288 0 0.0 ms
polledUartRx 96.42% 960 Hz @2 62 @ 0x200006f8 6665 1.0046 ms 0.9743 ms (#28386) 1.0347 ms (#28379)
Idle 0.00% 0 Hz 0 0.0 ms

STM32F767 Interrupt Priorities and FreeRTOS API calls

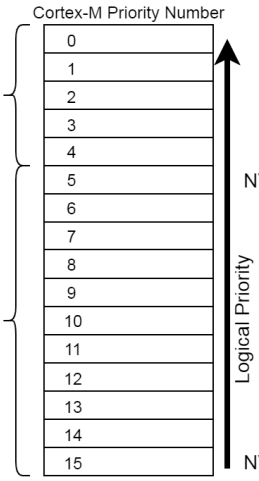
Example Setup Requirements:

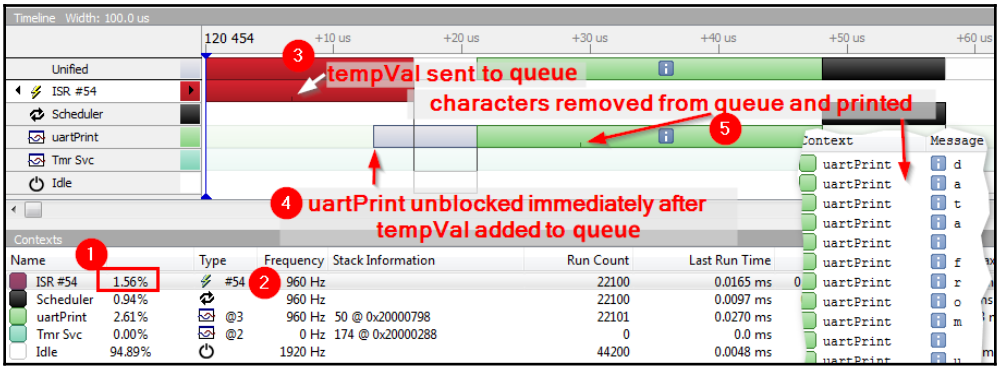
- NVIC_SetPriorityGrouping(0);
(allows all 4 NVIC priority bits used for priority numbers (16 unique priorities)

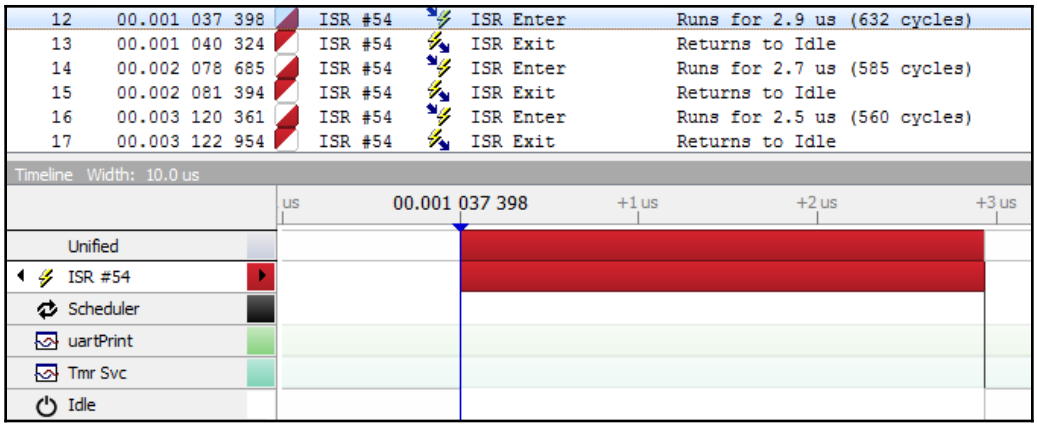
- FreeRTOSConfig.h:
configMAX_SYSCALL_INTERRUPT_PRIORITY
= 5 << (8 - 4) = 80

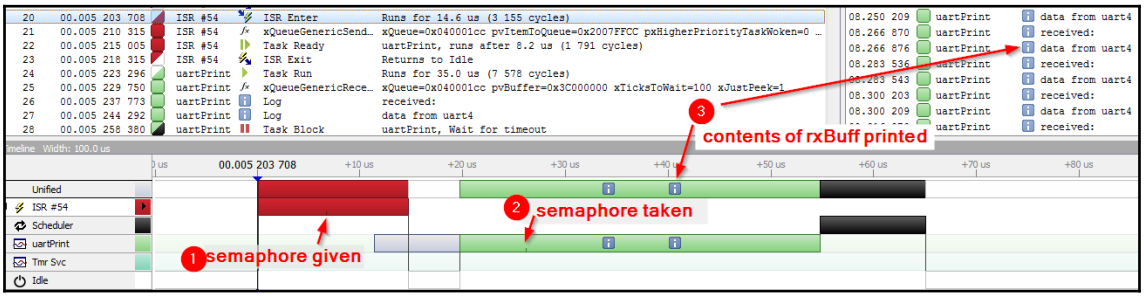
ISRs with Cortex M priorities 0-4:
- No interruptions by lower priority interrupts, tasks, or RTOS scheduler
- Not permitted to make any FreeRTOS API calls

ISRs with Cortex M Priorities 5-15:
- may call *FromISR* FreeRTOS API functions






12 00.001 037 398 ISR #54 ISR Enter Runs for 2.9 us (632 cycles)
13 00.001 040 324 ISR #54 ISR Exit Returns to Idle
14 00.002 078 685 ISR #54 ISR Enter Runs for 2.7 us (585 cycles)
15 00.002 081 394 ISR #54 ISR Exit Returns to Idle
16 00.003 120 361 ISR #54 ISR Enter Runs for 2.5 us (560 cycles)
17 00.003 122 954 ISR #54 ISR Exit Returns to Idle
Timeline Width: 10.0 us
us
00.001 037 398
+1 us
+2 us
+3 us
Unified
ISR #54
Scheduler
uartPrint
Tmr Svc
Idle


20 00.005 203 708 ISR #54 ISR Enter Runs for 14.6 us (3 155 cycles)
21 00.005 210 315 ISR #54 xQueueGenericSend… xQueue=0x040001cc pvItemToQueue=0x2007FFCC pxHigherPriorityTaskWoken=0 …
22 00.005 215 005 ISR #54 Task Ready uartPrint, runs after 8.2 us (1 791 cycles)
23 00.005 218 315 ISR #54 ISR Exit Returns to Idle
24 00.005 223 296 uartPrint Task Run Runs for 35.0 us (7 578 cycles)
25 00.005 229 750 uartPrint xQueueGenericRece… xQueue=0x040001cc pvBuffer=0x3C000000 xTicksToWait=100 xJustPeek=1
26 00.005 237 773 uartPrint Log received:
27 00.005 244 292 uartPrint Log data from uart4
28 00.005 258 380 uartPrint Task Block uartPrint, Wait for timeout
08.250 209 uartPrint data from uart4
08.266 870 uartPrint received:
08.266 876 uartPrint data from uart4
08.283 536 uartPrint received:
08.283 543 uartPrint data from uart4
08.300 203 uartPrint received:
08.300 209 uartPrint data from uart4
uartPrint received:
3
contents of rxBuff printed
00.005 203 708
+10 us
+20 us
+30 us
+40 us
+50 us
+60 us
+70 us
+80 us
Unified
ISR #54
Scheduler
uartPrint
Tmr Svc
Idle
2
semaphore taken
1
semaphore given

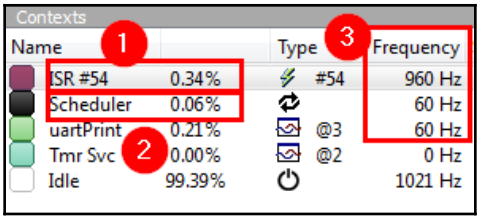
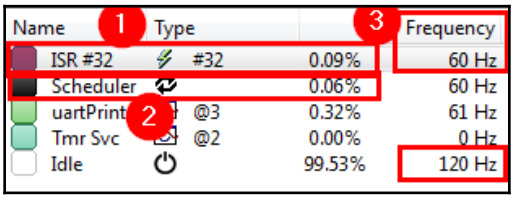
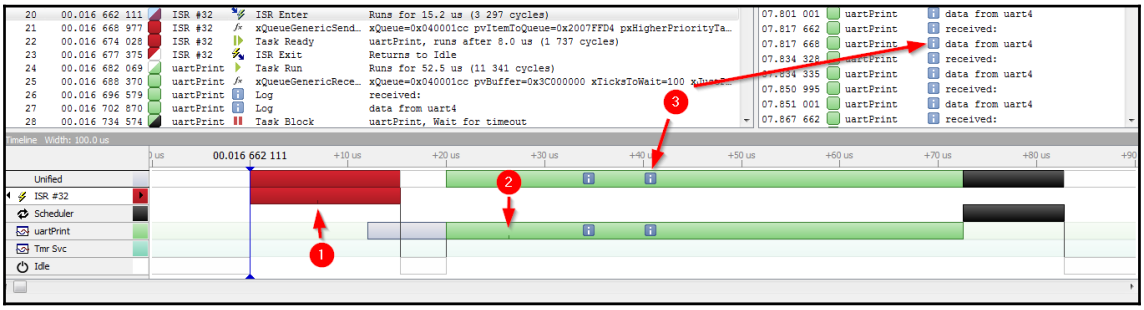
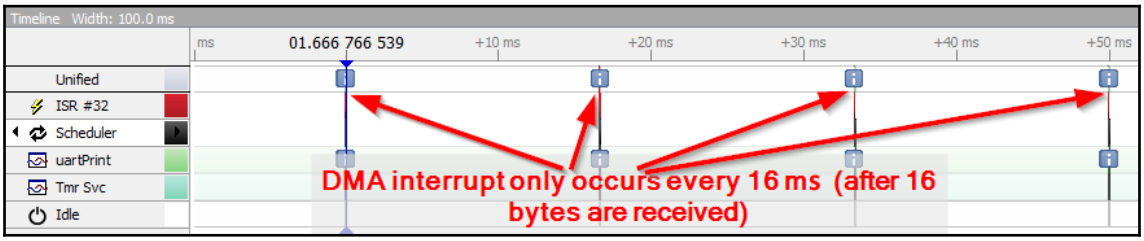

Contexts
Name
1
Type
3
Frequency
ISR #54 0.34% #54 960 Hz
Scheduler 0.06% 60 Hz
uartPrint 0.21% @3 60 Hz
Tmr Svc 2 0.00% @2 0 Hz
Idle 99.39% 1021 Hz

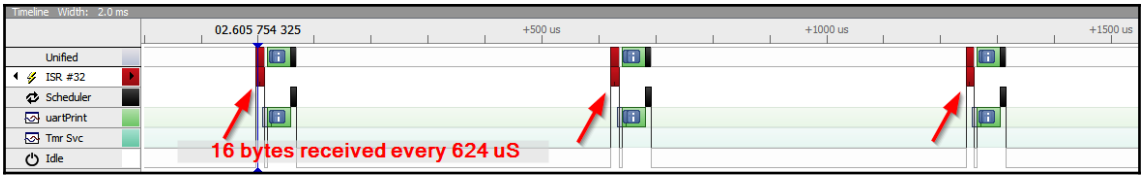
Table 27. DMA1 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	SPDIFRX_DT	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	SPDIFRX_CS	SPI3_TX
Channel 1	I2C1_RX	I2C3_RX	TIM7_UP	-	TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2C4 _RX	TIM4_CH2	-	I2C4 _RX	TIM4_UP	TIM4_CH3
Channel 3	-	TIM2_UP TIM2_CH3	I2C3_RX	-	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Channel 5	UART8_TX	UART7_TX	TIM3_CH4 TIM3_UP	UART7_RX	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX	TIM3_CH3
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	-	TIM5_UP	-
Channel 7	-	TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX
Channel 8	I2C3_TX	I2C4_RX	-	-	I2C2_TX	-	I2C4_TX	-
Channel 9	-	SPI2_RX	-	-	-	-	SPI2_TX	-

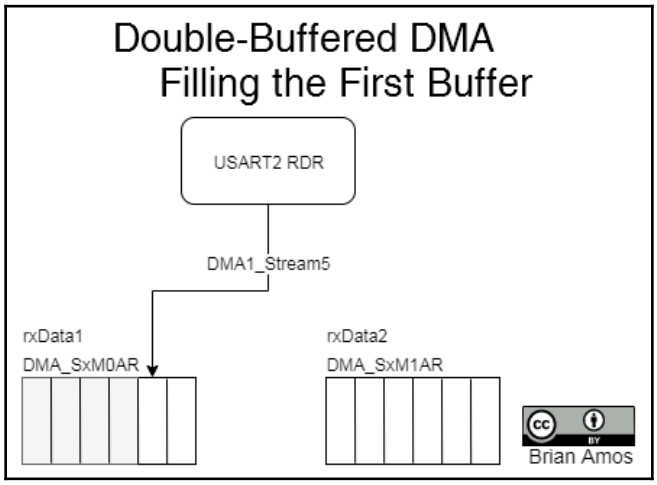


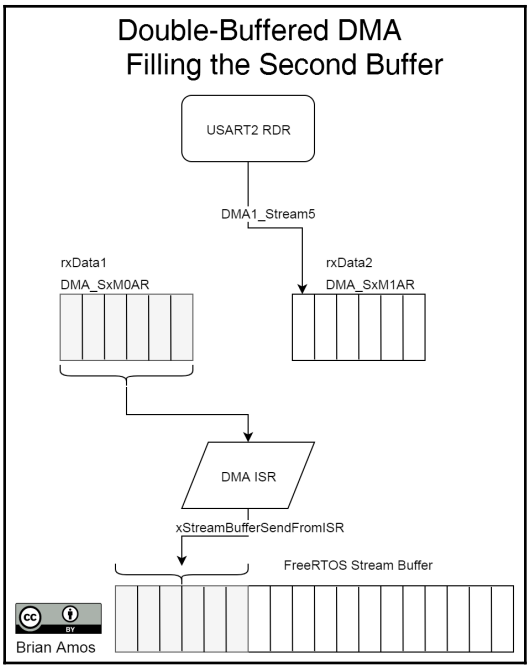


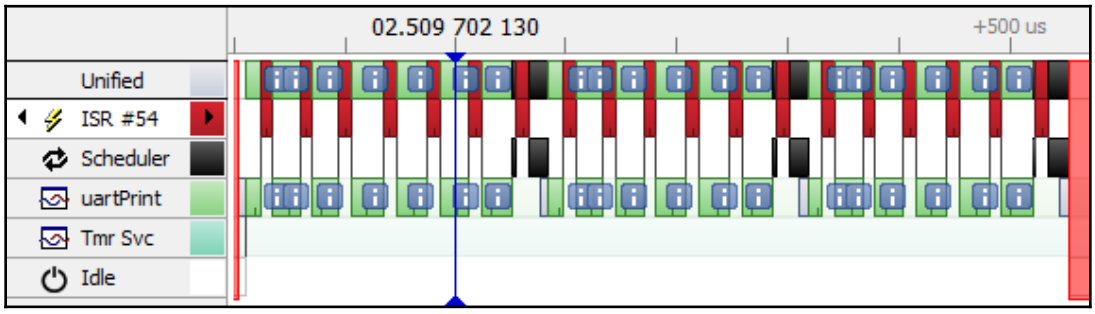




Name	Type		Frequency
ISR #32	#32	2.29%	1604 Hz
Scheduler		1.69%	1604 Hz
uartPrint	@3	6.24%	1604 Hz
Tmr Svc	@2	0.00%	0 Hz
Idle		89.78%	3208 Hz



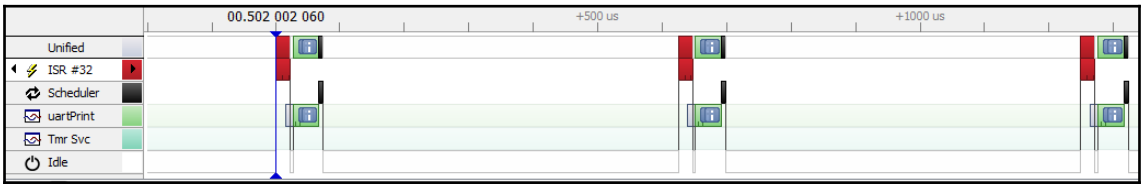

Double-Buffered DMA
Filling the Second Buffer
USART2 RDR
DMA1_Stream5
rxData1
DMA_SxM0AR
rxData2
DMA_SxM1AR
DMA ISR
xStreamBufferSendFromISR
FreeRTOS Stream Buffer
Brian Amos


02.509 702 130
+500 us
Unified
ISR #54
Scheduler
uartPrint
Tmr Svc
Idle

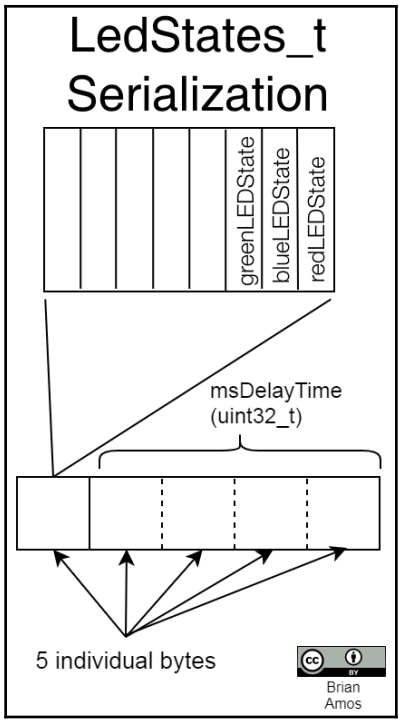
Terminal

Timestamp	Context	Message
34.294 774	uartPrint	
34.294 815	uartPrint	d
34.294 846	uartPrint	a
34.294 920	uartPrint	t
34.294 936	uartPrint	a
34.301 508	uartPrint	a
34.301 524	uartPrint	r
34.301 555	uartPrint	t
34.301 595	uartPrint	4
34.301 636	uartPrint	
34.301 677	uartPrint	d
34.301 707	uartPrint	a
34.301 781	uartPrint	t
34.301 797	uartPrint	a
34.301 828	uartPrint	

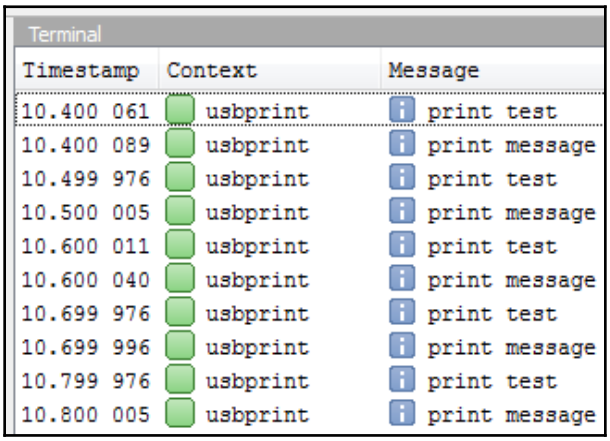
missing 'u'

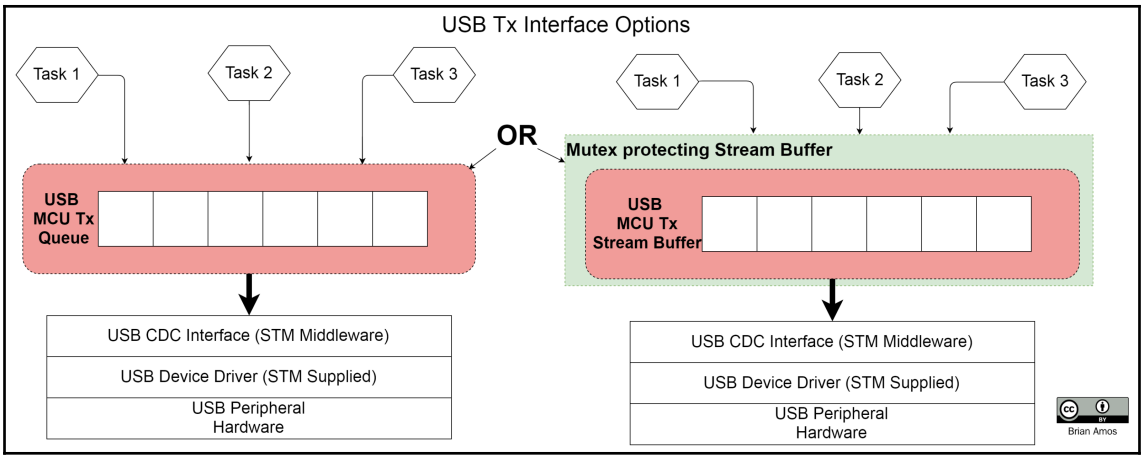


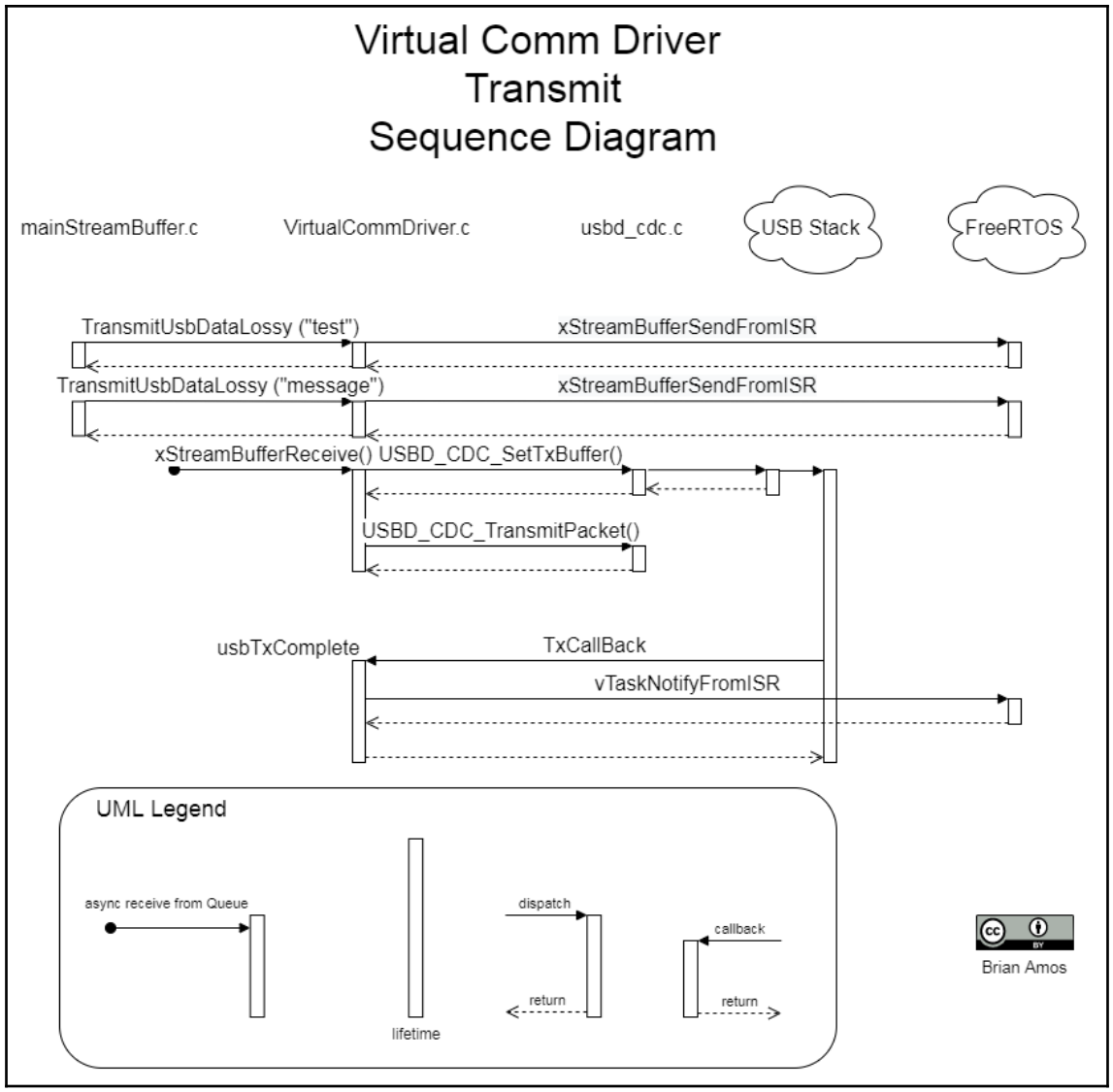
Name	Type		Frequency
ISR #32	#32	3.35%	1430 Hz
Scheduler		0.95%	1424 Hz
uartPrint	@3	6.09%	1426 Hz
Tmr Svc	@2	0.00%	0 Hz
Idle		89.61%	2851 Hz

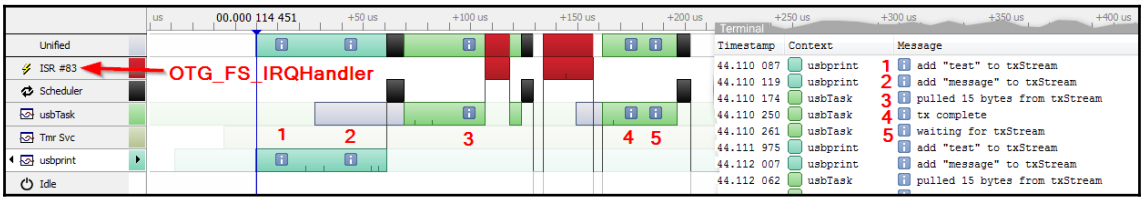

LedStates_t
Serialization
greenLEDState
blueLEDState
redLEDState
msDelayTime
(uint32_t)
5 individual bytes
Brian
Amos

Chapter 11: Sharing Hardware Peripherals across Tasks

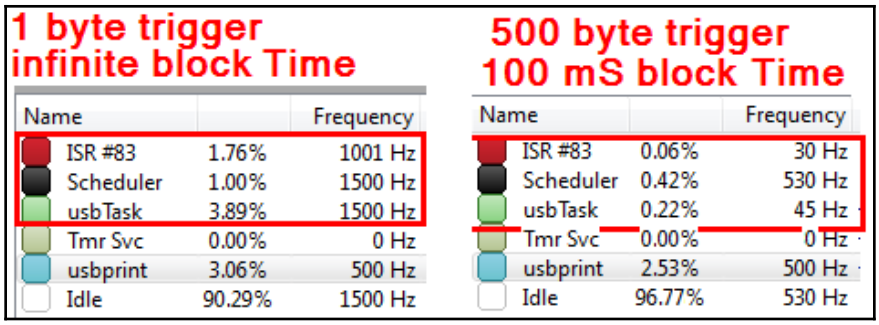


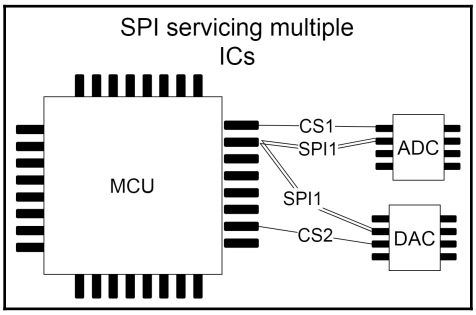

USB Tx Interface Options
Task 1
Task 2
Task 3
OR
Task 1
Task 2
Task 3
Mutex protecting Stream Buffer
USB
MCU Tx
Queue
USB
MCU Tx
Stream Buffer
USB CDC Interface (STM Middleware)
USB Device Driver (STM Supplied)
USB Peripheral
Hardware
USB CDC Interface (STM Middleware)
USB Device Driver (STM Supplied)
USB Peripheral
Hardware
Brian Amos


Virtual Comm Driver
Transmit
Sequence Diagram
mainStreamBuffer.c
VirtualCommDriver.c
usbd_cdc.c
USB Stack
FreeRTOS
TransmitUsbDataLossy ("test")
xStreamBufferSendFromISR
TransmitUsbDataLossy ("message")
xStreamBufferSendFromISR
xStreamBufferReceive()
USBD_CDC_SetTxBuffer()
USBD_CDC_TransmitPacket()
usbTxComplete
TxCallBack
vTaskNotifyFromISR
UML Legend
async receive from Queue
lifetime
dispatch
return
callback
return
Brian Amos

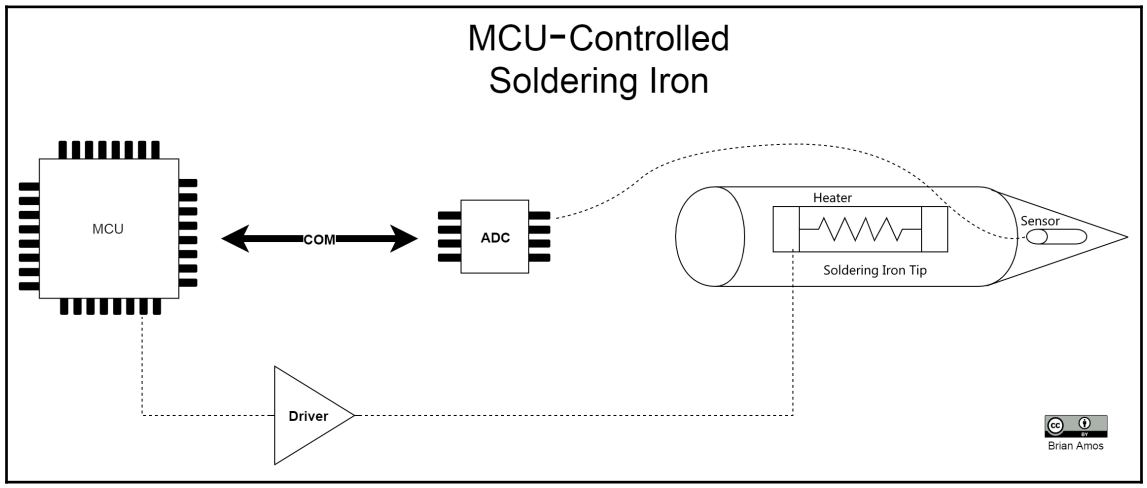


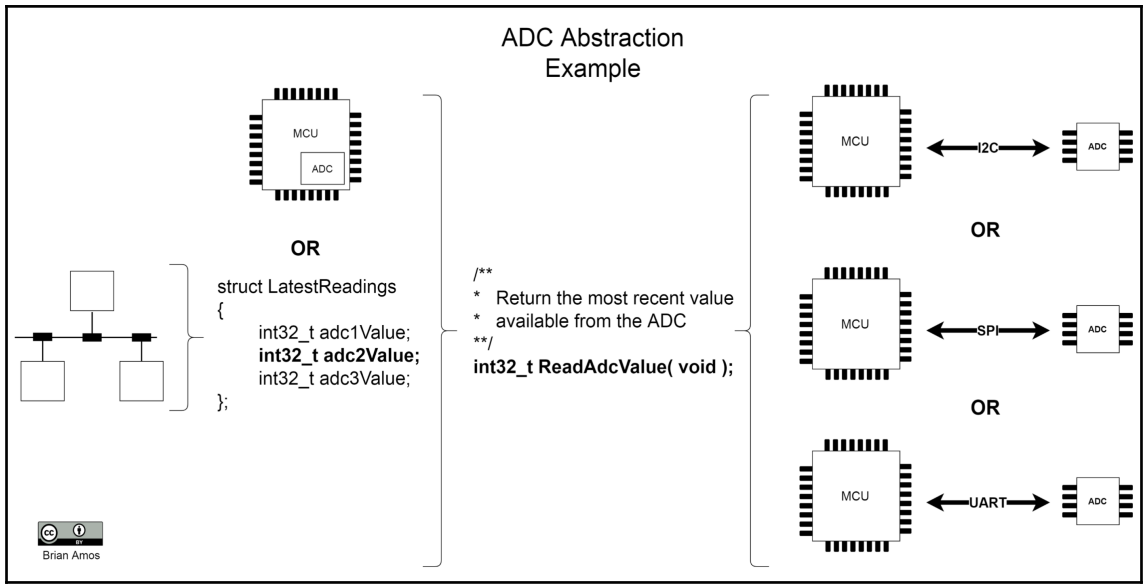
Name		Frequency
ISR #83	1.76%	1001 Hz
Scheduler	1.00%	1500 Hz
usbTask	3.89%	1500 Hz
Tmr Svc	0.00%	0 Hz
usbprint	3.06%	500 Hz
Idle	90.29%	1500 Hz

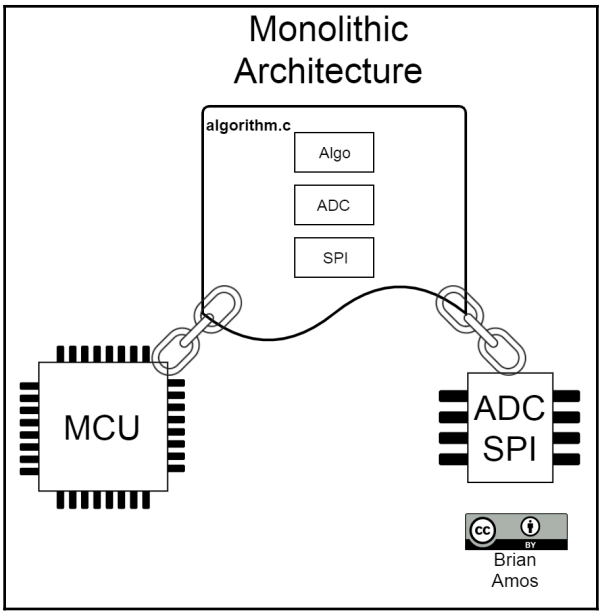

1 byte trigger
infinite block Time
Name
Frequency
ISR #83 1.76% 1001 Hz
Scheduler 1.00% 1500 Hz
usbTask 3.89% 1500 Hz
Tmr Svc 0.00% 0 Hz
usbprint 3.06% 500 Hz
Idle 90.29% 1500 Hz
500 byte trigger
100 mS block Time
Name
Frequency
ISR #83 0.06% 30 Hz
Scheduler 0.42% 530 Hz
usbTask 0.22% 45 Hz
Tmr Svc 0.00% 0 Hz
usbprint 2.53% 500 Hz
Idle 96.77% 530 Hz

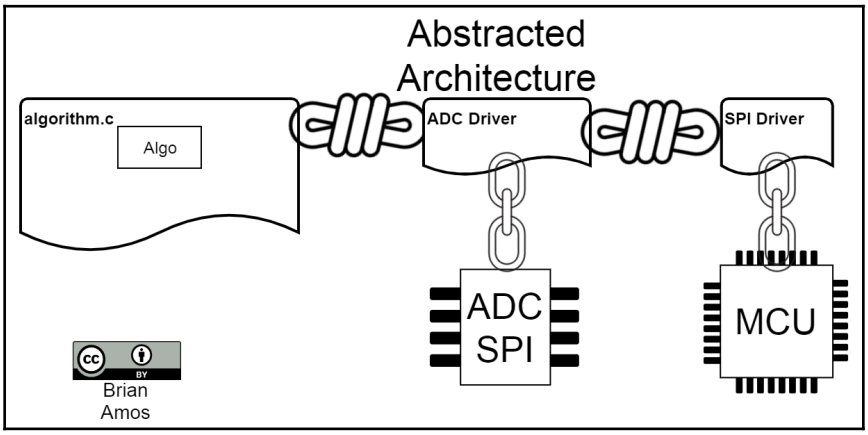

SPI servicing multiple ICs
MCU
CS1
SPI1
ADC
SPI1
CS2
DAC

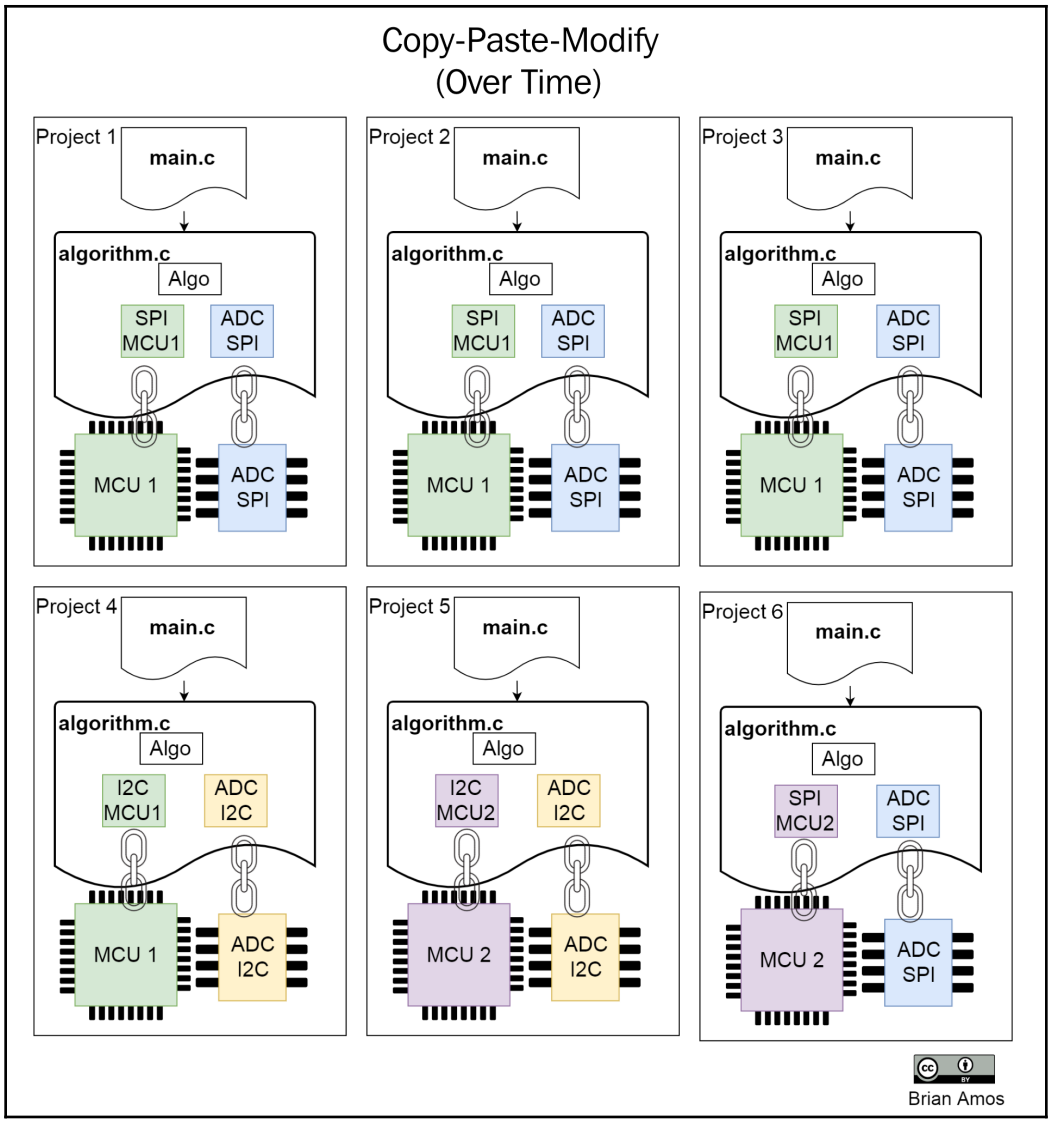
Chapter 12: Tips for Creating a Well-Abstracted Architecture

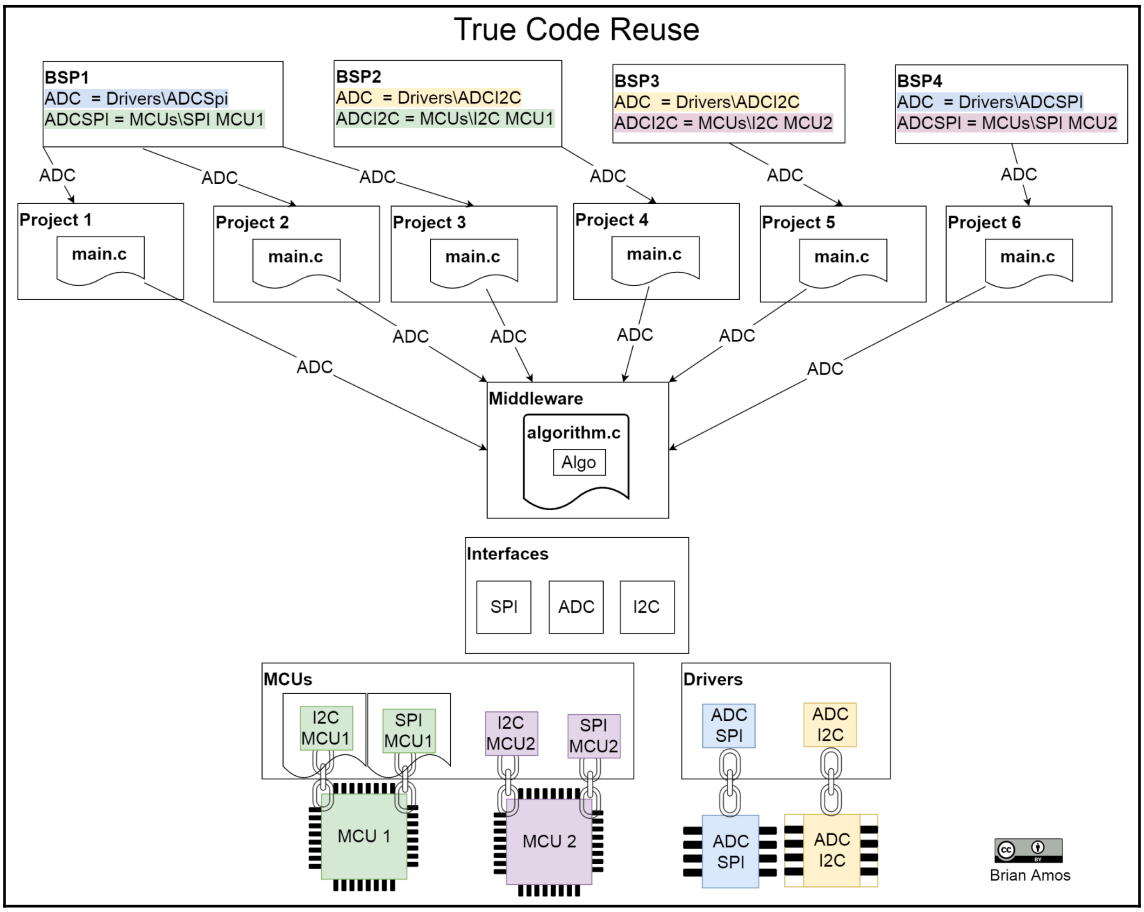


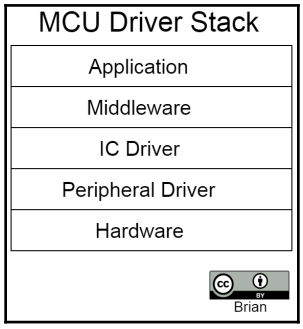



Monolithic
Architecture
algorithm.c
Algo
ADC
SPI
MCU
ADC
SPI
Brian
Amos

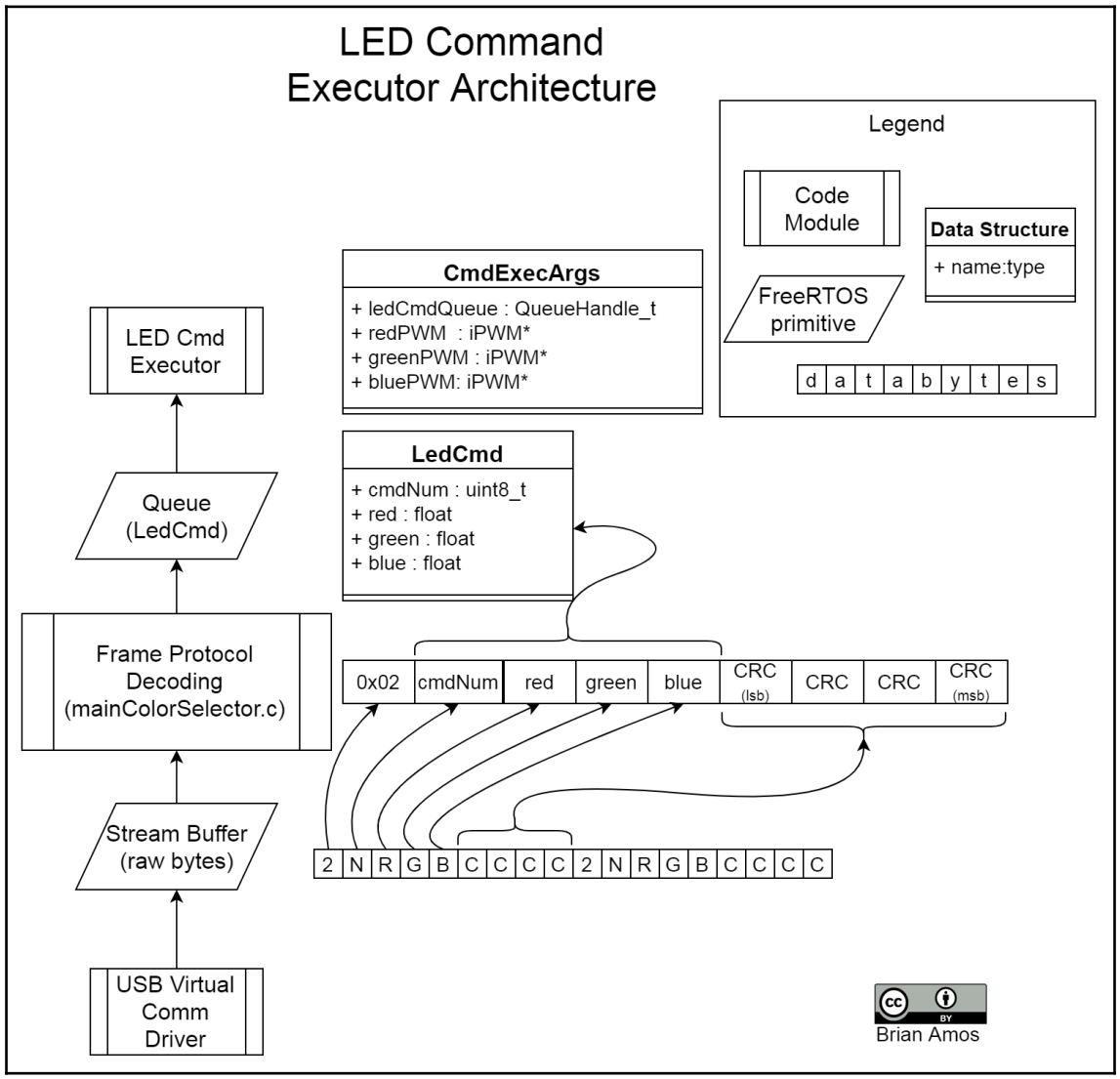

Abstracted
Architecture
algorithm.c
Algo
ADC Driver
SPI Driver
ADC
SPI
MCU
Brian
Amos


Copy-Paste-Modify
(Over Time)
Project 1
main.c
algorithm.c
Algo
SPI MCU1
ADC SPI
MCU 1
ADC SPI
Project 2
main.c
algorithm.c
Algo
SPI MCU1
ADC SPI
MCU 1
ADC SPI
Project 3
main.c
algorithm.c
Algo
SPI MCU1
ADC SPI
MCU 1
ADC SPI
Project 4
main.c
algorithm.c
Algo
I2C MCU1
ADC I2C
MCU 1
ADC I2C
Project 5
main.c
algorithm.c
Algo
I2C MCU2
ADC I2C
MCU 2
ADC I2C
Project 6
main.c
algorithm.c
Algo
SPI MCU2
ADC SPI
MCU 2
ADC SPI
Brian Amos

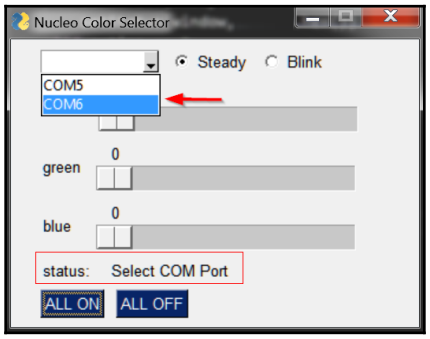

True Code Reuse
BSP1
ADC = Drivers\ADCSpi
ADCSPI = MCUs\SPI MCU1
BSP2
ADC = Drivers\ADCI2C
ADCI2C = MCUs\I2C MCU1
BSP3
ADC = Drivers\ADCI2C
ADCI2C = MCUs\I2C MCU2
BSP4
ADC = Drivers\ADCSPI
ADCSPI = MCUs\SPI MCU2
ADC
Project 1
main.c
Project 2
main.c
Project 3
main.c
Project 4
main.c
Project 5
main.c
Project 6
main.c
Middleware
algorithm.c
Algo
Interfaces
SPI
ADC
I2C
MCUs
I2C MCU1
SPI MCU1
I2C MCU2
SPI MCU2
MCU 1
MCU 2
Drivers
ADC SPI
ADC I2C
ADC SPI
ADC I2C
Brian Amos

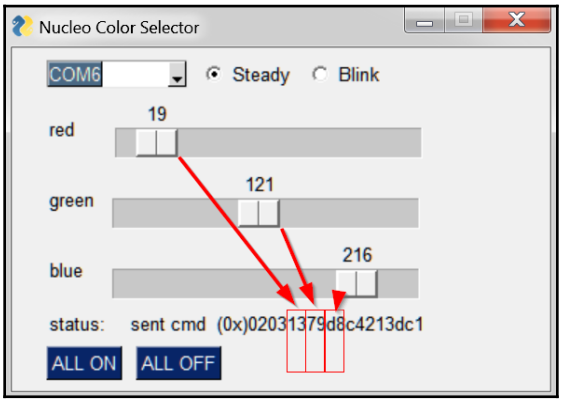

MCU Driver Stack
Application
Middleware
IC Driver
Peripheral Driver
Hardware
Brian

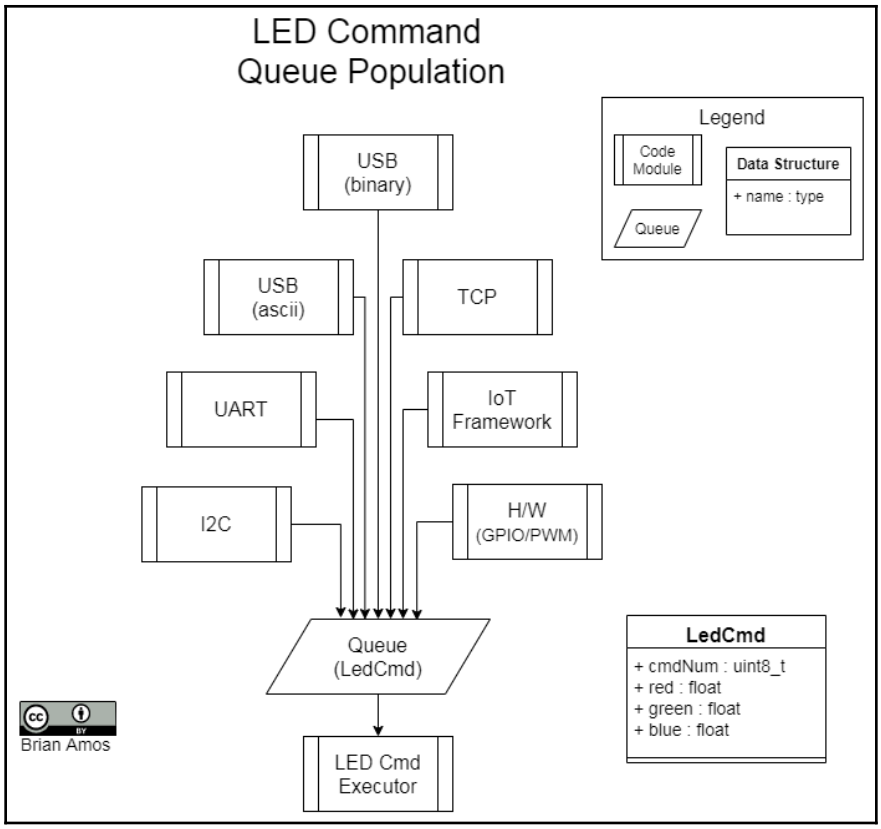
Chapter 13: Creating Loose Coupling with Queues



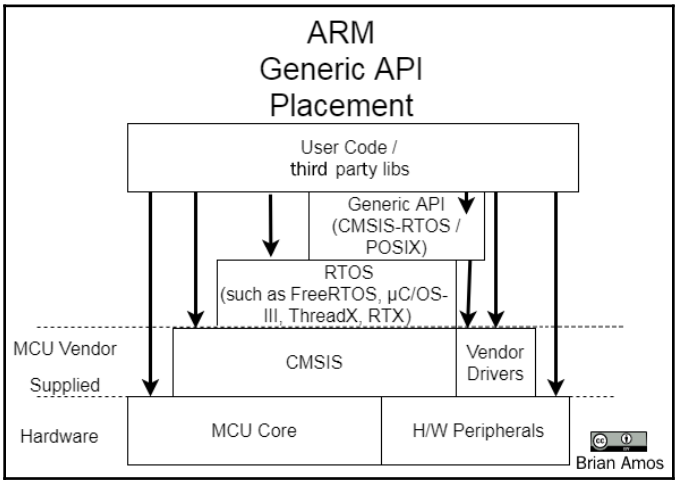
LED Command Frame								
0x02	cmdNum	red	green	blue	CRC (lsb)	CRC	CRC	CRC (msb)

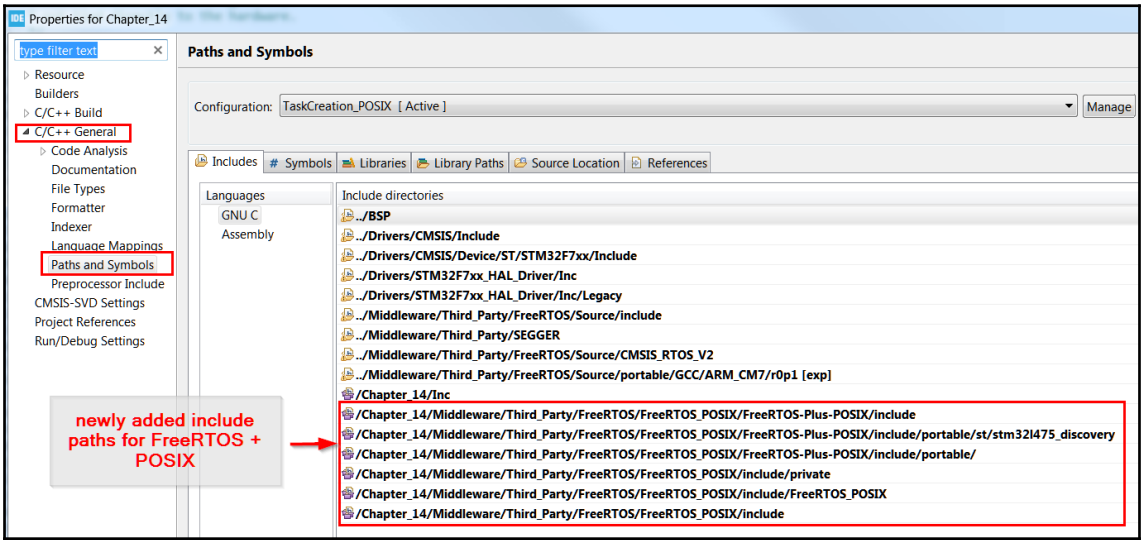




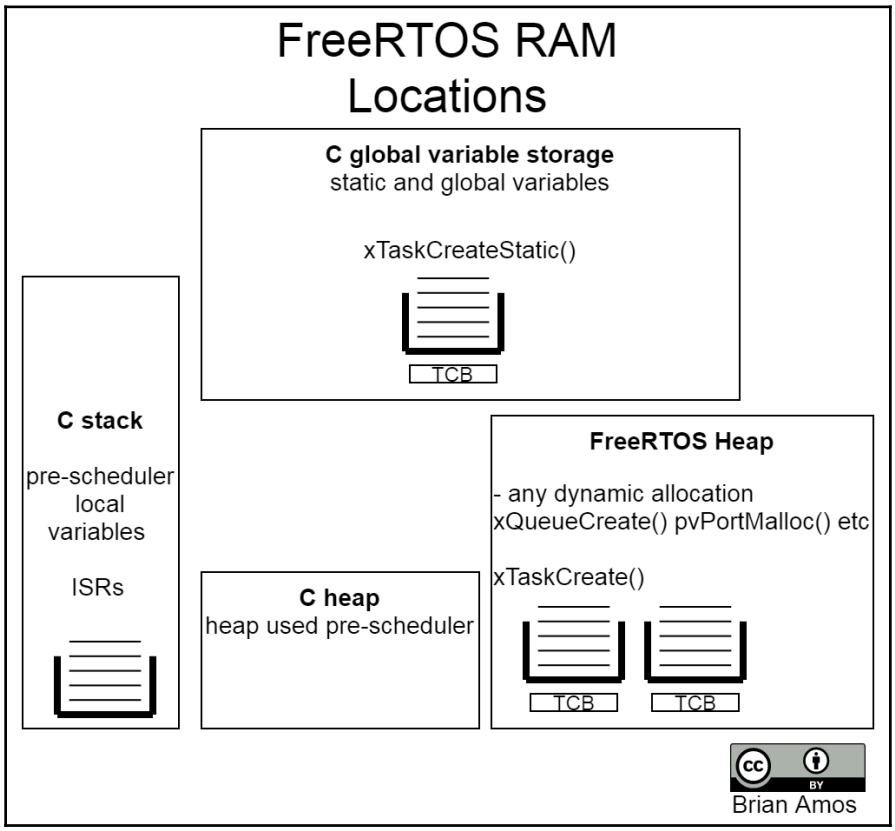

LED Command
Queue Population
Legend
Code Module
Data Structure
+ name : type
Queue
USB (binary)
USB (ascii)
TCP
UART
IoT Framework
I2C
H/W (GPIO/PWM)
Queue (LedCmd)
LED Cmd Executor
LedCmd
+ cmdNum : uint8_t
+ red : float
+ green : float
+ blue : float
Brian Amos

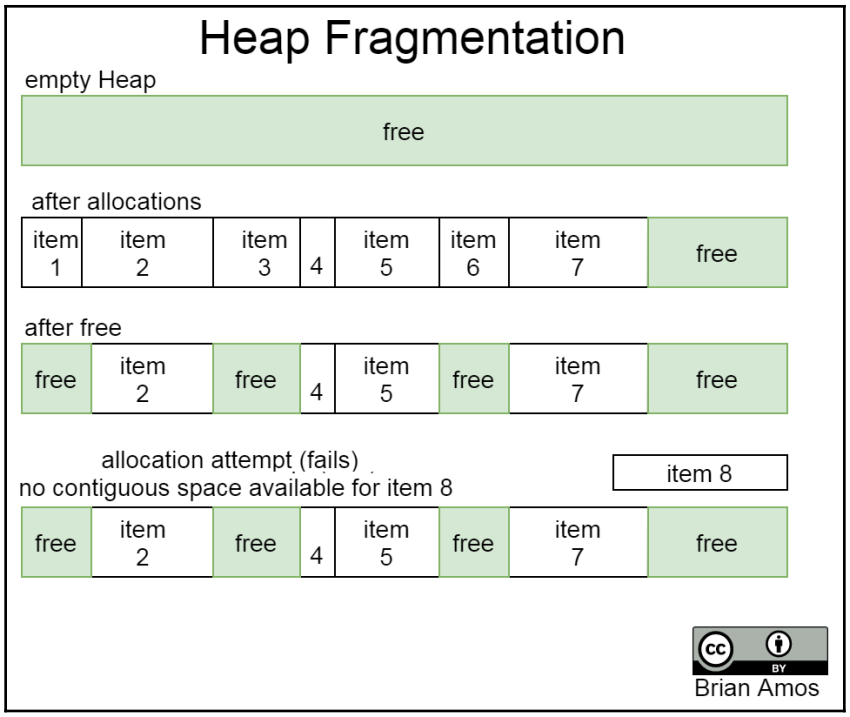
Chapter 14: Choosing an RTOS API

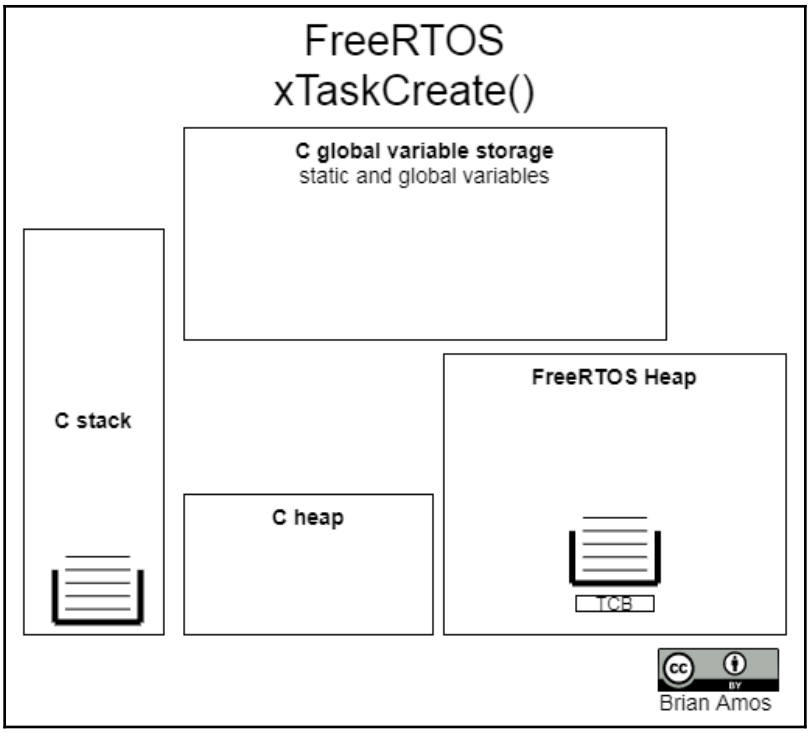


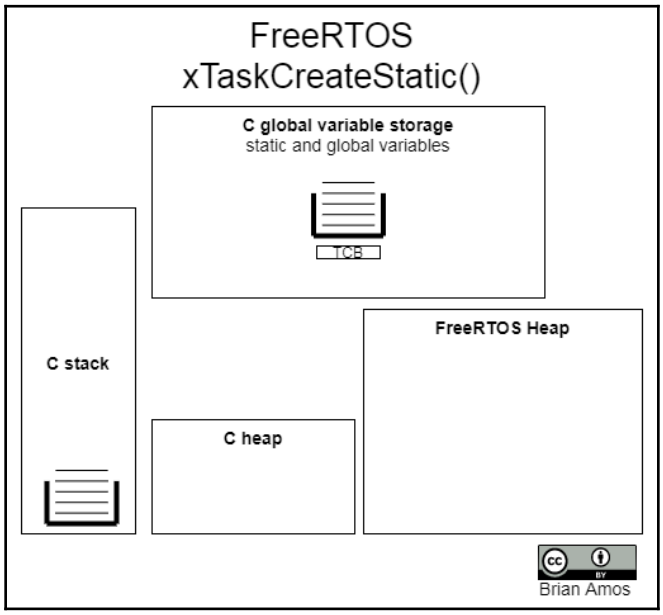


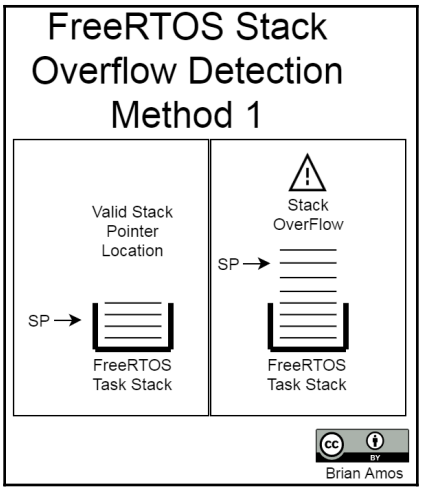
Chapter 15: FreeRTOS Memory Management

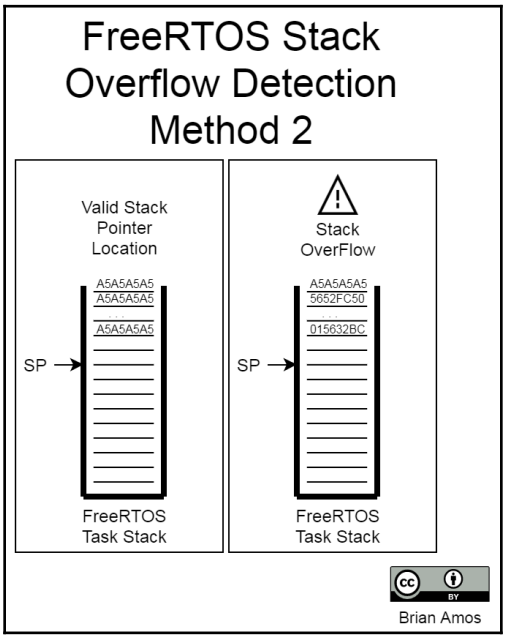



Heap Fragmentation
empty Heap
free
after allocations
item 1
item 2
item 3
4
item 5
item 6
item 7
free
after free
free
item 2
free
4
item 5
free
item 7
free
allocation attempt (fails)
no contiguous space available for item 8
item 8
free
item 2
free
4
item 5
free
item 7
free
Brian Amos

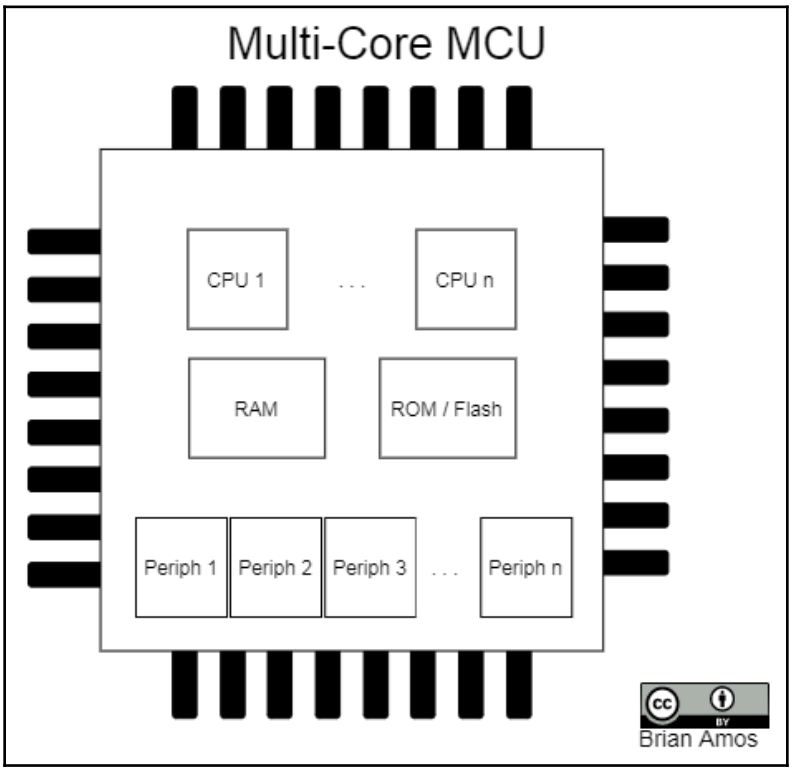

FreeRTOS
xTaskCreate()
C global variable storage
static and global variables
C stack
FreeRTOS Heap
C heap
TCB
Brian Amos

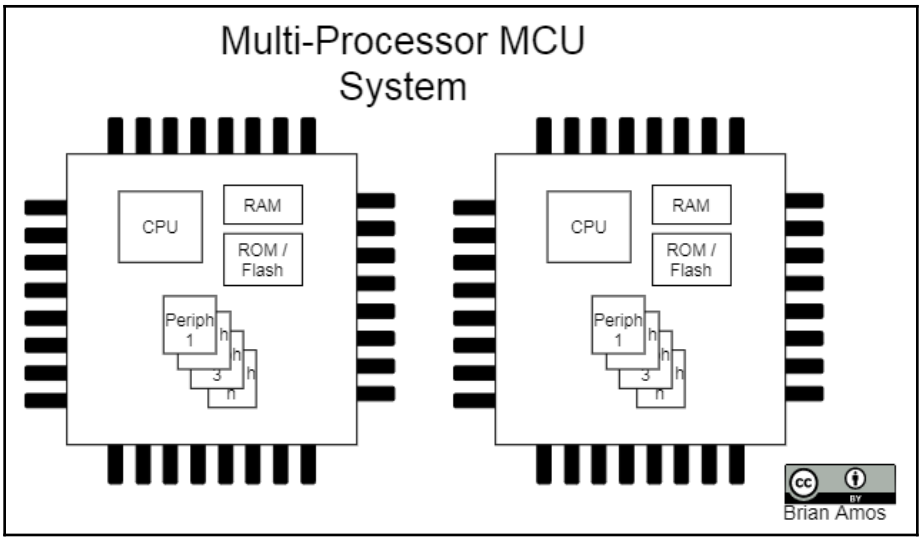

FreeRTOS
xTaskCreateStatic()
C global variable storage
static and global variables
TCB
C stack
FreeRTOS Heap
C heap
Brian Amos

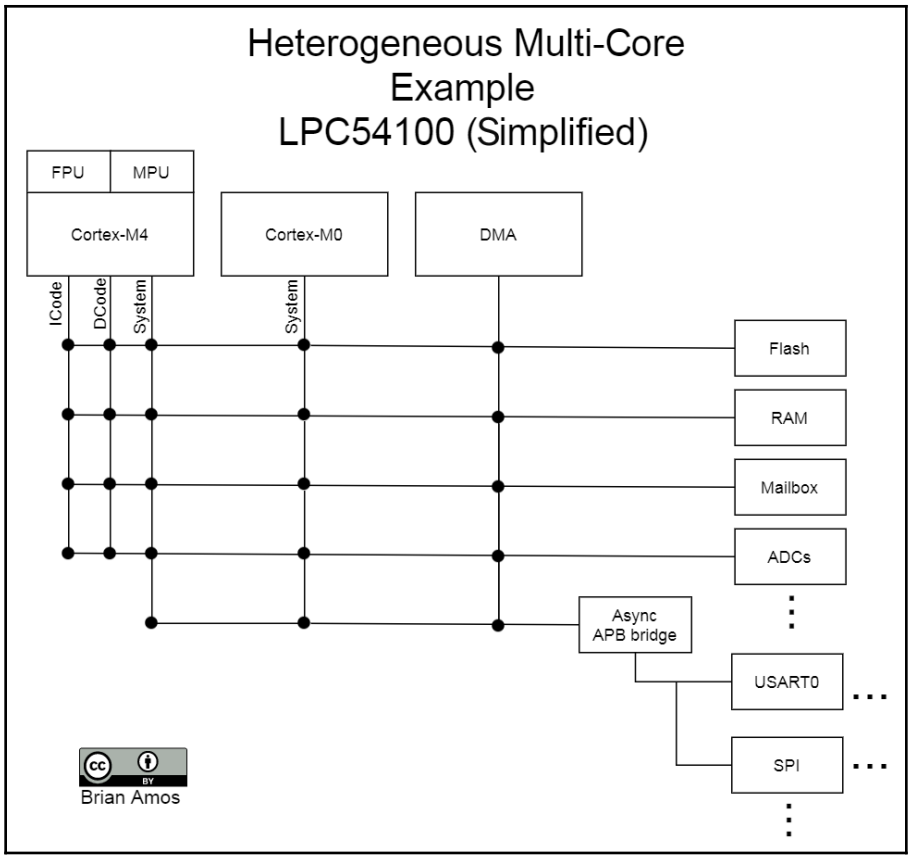

FreeRTOS Stack
Overflow Detection
Method 1
Valid Stack
Pointer
Location
SP
FreeRTOS
Task Stack
Stack
OverFlow
SP
FreeRTOS
Task Stack
Brian Amos

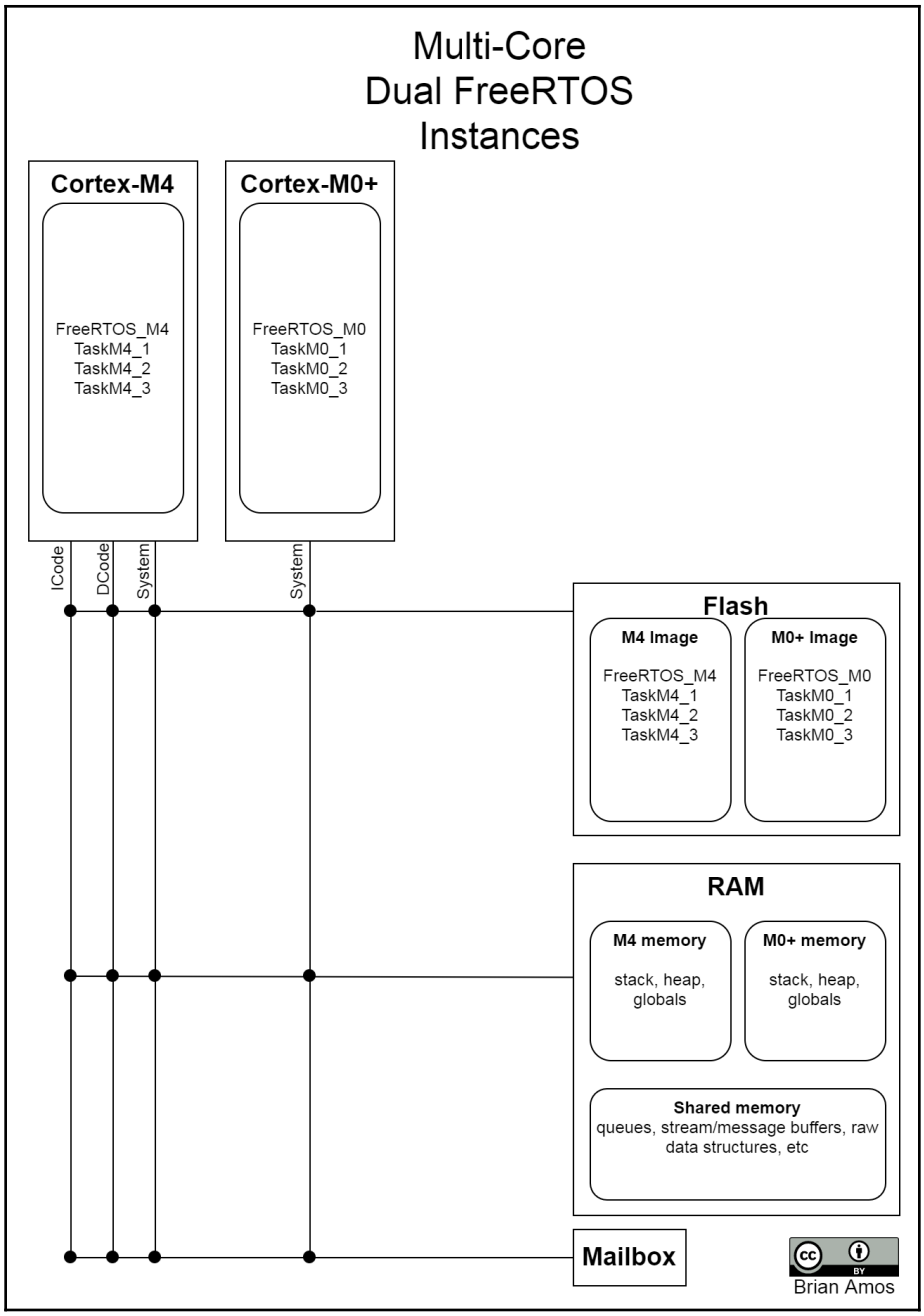

FreeRTOS Stack
Overflow Detection
Method 2
Valid Stack
Pointer
Location
A5A5A5A5
A5A5A5A5
A5A5A5A5
SP
FreeRTOS
Task Stack
Stack
OverFlow
A5A5A5A5
5652FC50
015632BC
SP
FreeRTOS
Task Stack
Brian Amos

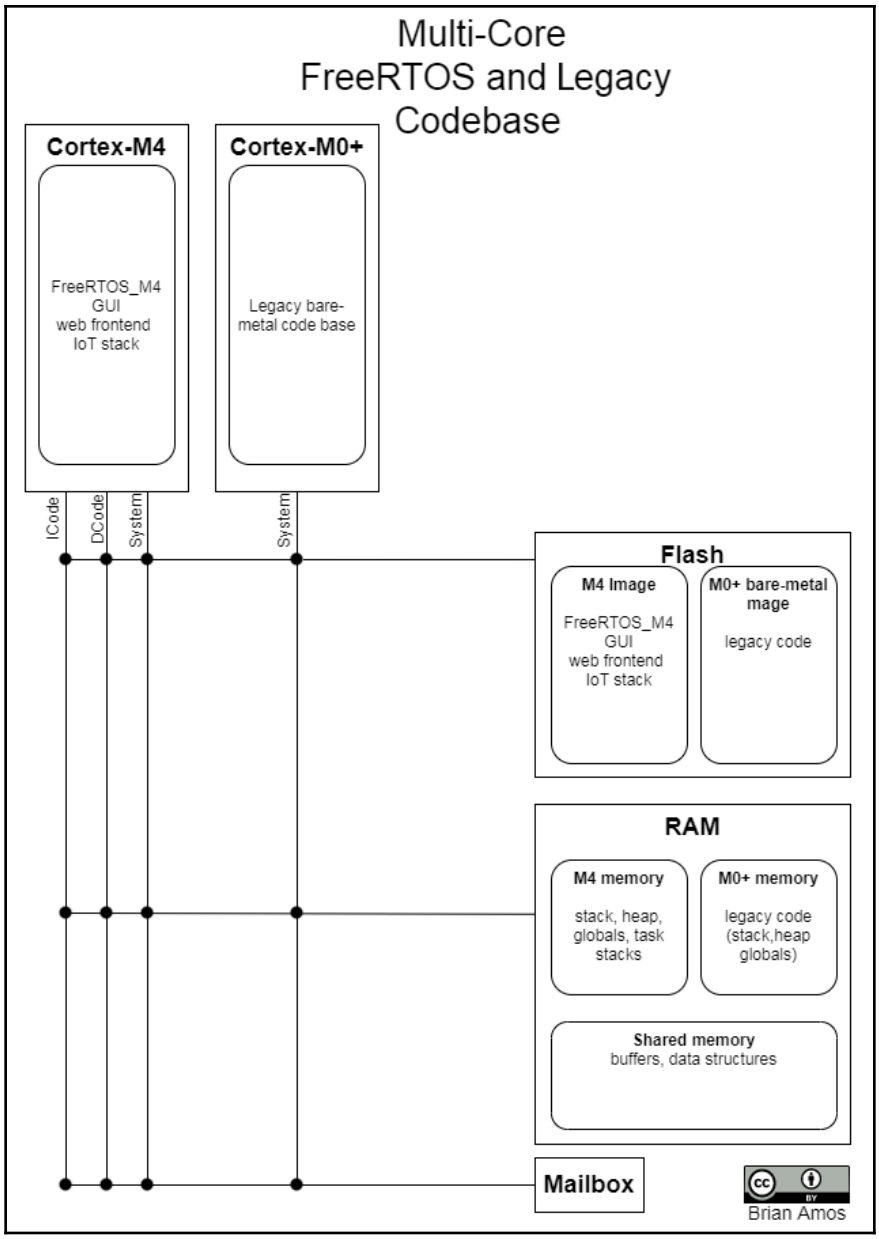
Chapter 16: Multi-Processor and Multi-Core Systems

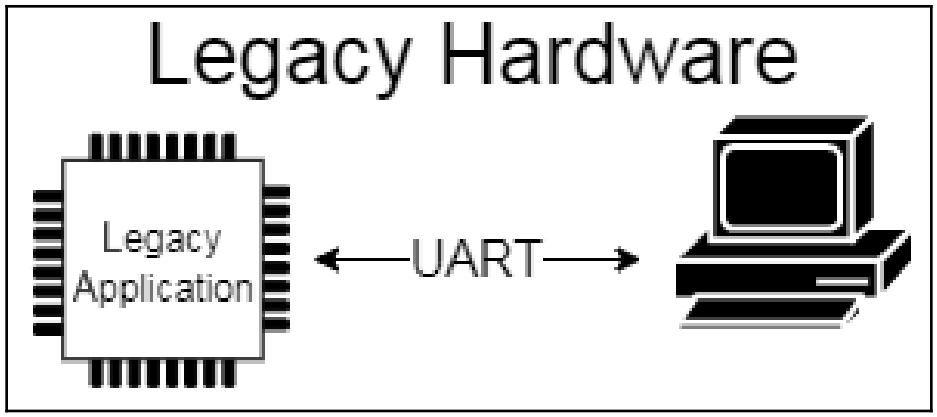


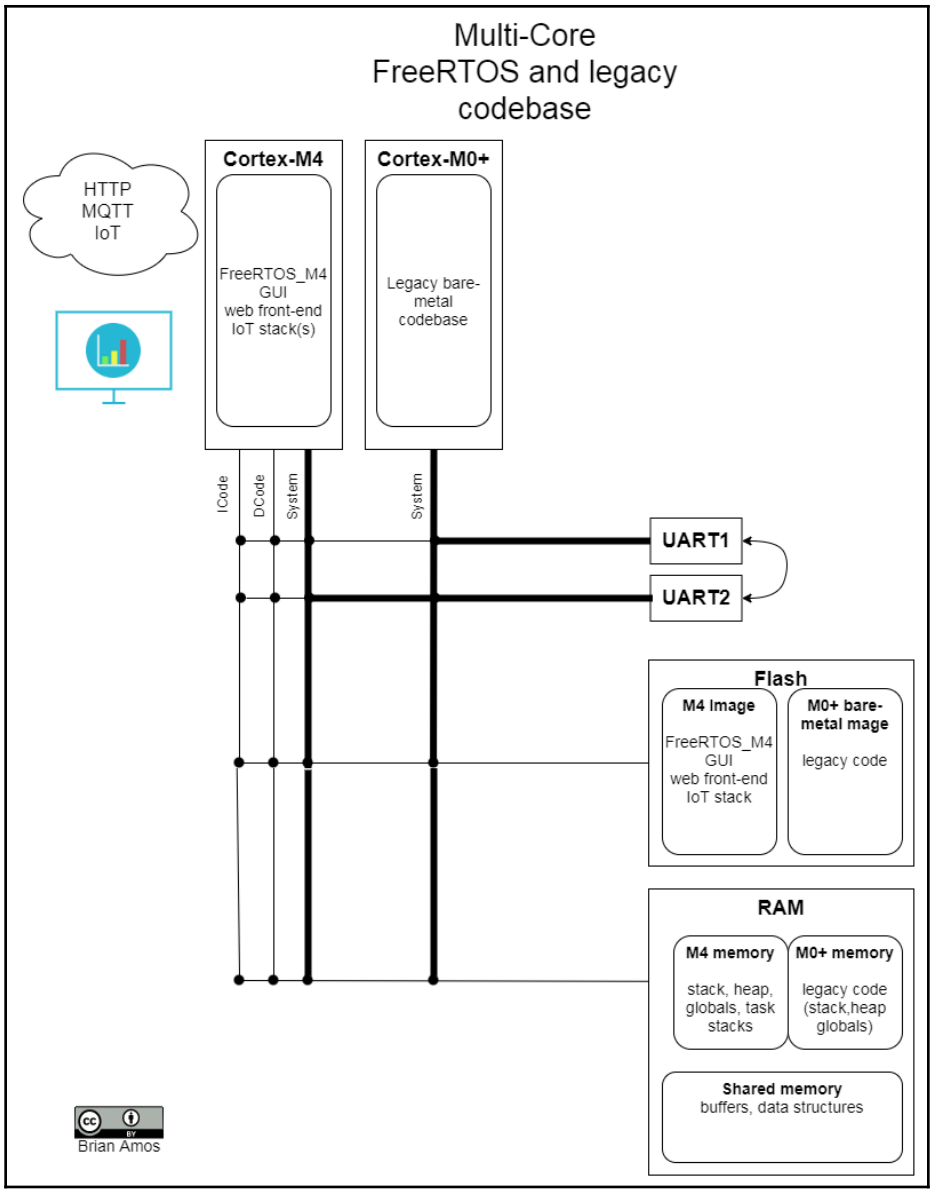

Multi-Processor MCU System
CPU
RAM
ROM / Flash
Periph 1
2
3
n
CPU
RAM
ROM / Flash
Periph 1
2
3
n
Brian Amos

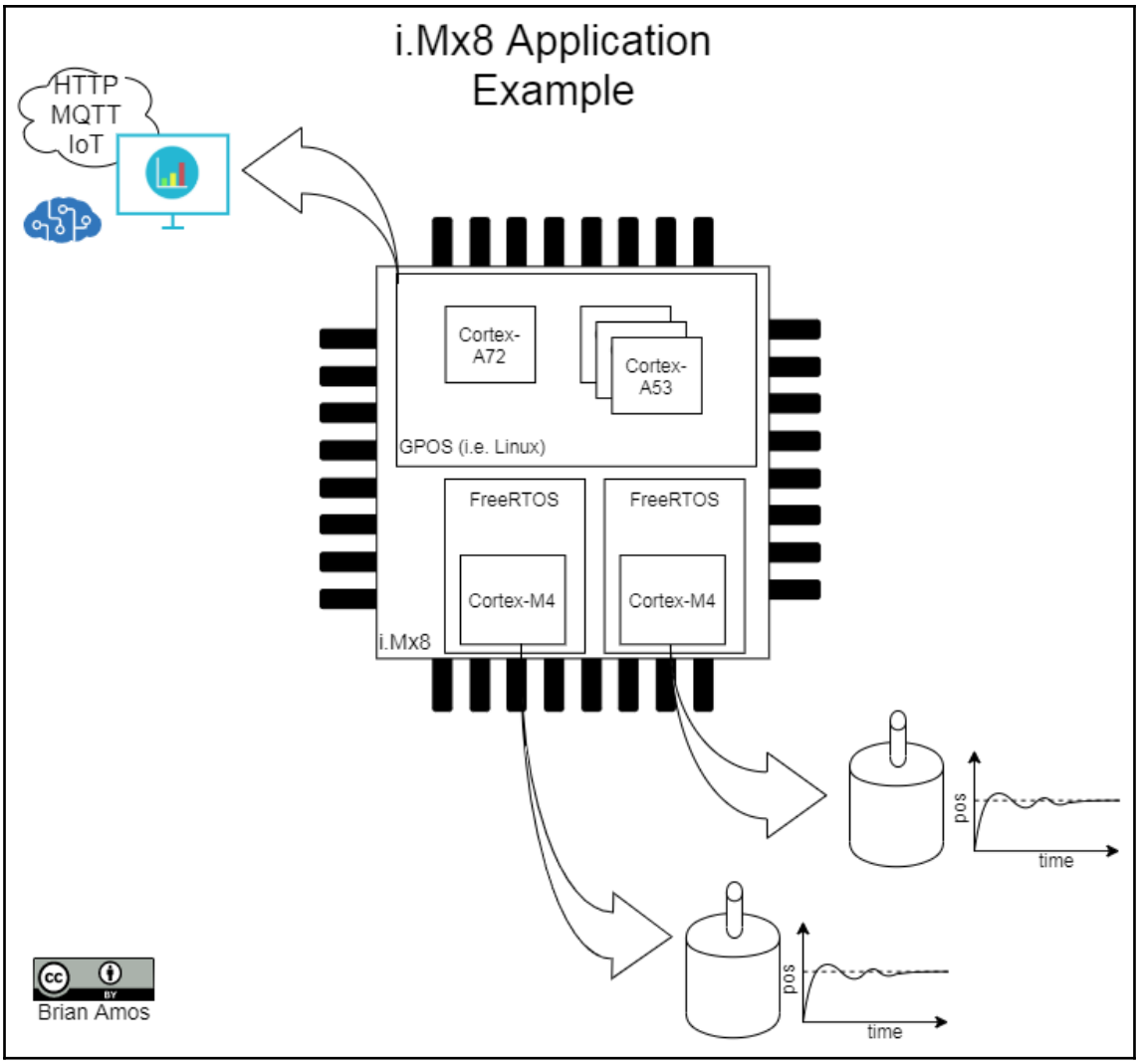

Heterogeneous Multi-Core
Example
LPC54100 (Simplified)
FPU
MPU
Cortex-M4
Cortex-M0
DMA
ICode
DCode
System
System
Flash
RAM
Mailbox
ADCs
Async
APB bridge
USART0
SPI
Brian Amos

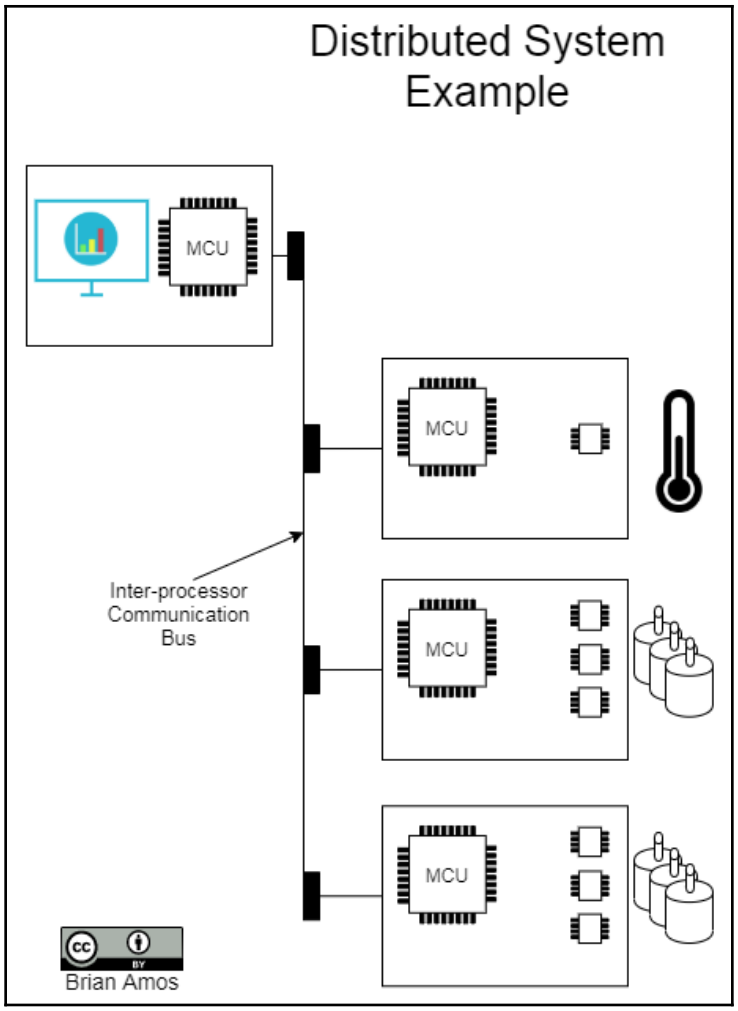

Multi-Core
Dual FreeRTOS
Instances
Cortex-M4
FreeRTOS_M4
TaskM4_1
TaskM4_2
TaskM4_3
Cortex-M0+
FreeRTOS_M0
TaskM0_1
TaskM0_2
TaskM0_3
ICode
DCode
System
System
Flash
M4 Image
FreeRTOS_M4
TaskM4_1
TaskM4_2
TaskM4_3
M0+ Image
FreeRTOS_M0
TaskM0_1
TaskM0_2
TaskM0_3
RAM
M4 memory
stack, heap, globals
M0+ memory
stack, heap, globals
Shared memory
queues, stream/message buffers, raw data structures, etc
Mailbox
Brian Amos

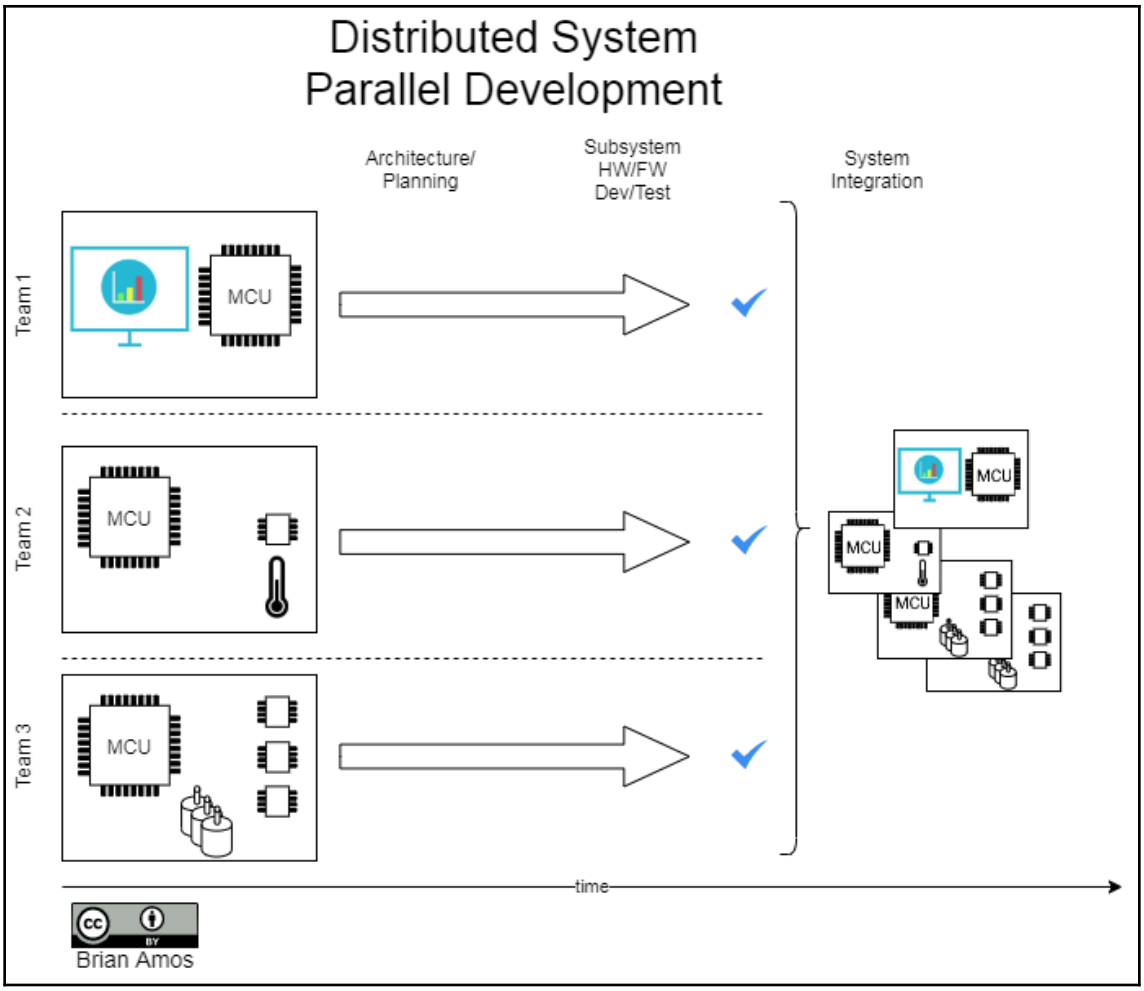

Multi-Core
FreeRTOS and Legacy
Codebase
Cortex-M4
FreeRTOS_M4
GUI
web frontend
IoT stack
Cortex-M0+
Legacy bare-
metal code base
ICode
DCode
System
System
Flash
M4 Image
FreeRTOS_M4
GUI
web frontend
IoT stack
M0+ bare-metal
mage
legacy code
RAM
M4 memory
stack, heap,
globals, task
stacks
M0+ memory
legacy code
(stack,heap
globals)
Shared memory
buffers, data structures
Mailbox
Brian Amos


Legacy Hardware
Legacy
Application
UART

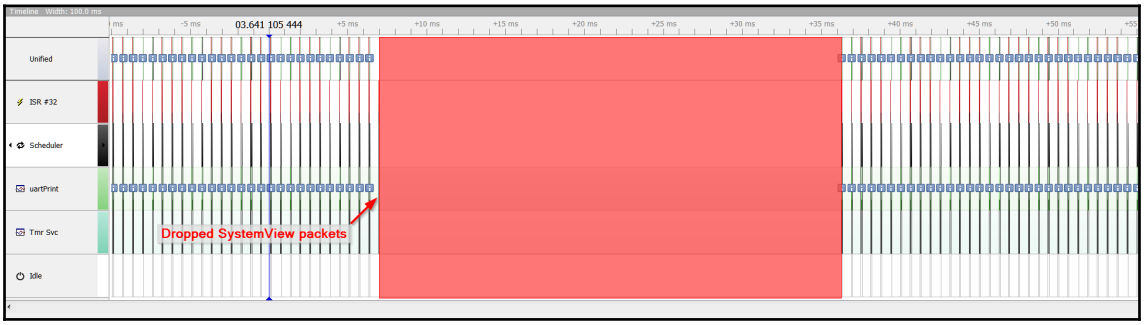

Multi-Core
FreeRTOS and legacy
codebase
HTTP
MQTT
IoT
Cortex-M4
FreeRTOS_M4
GUI
web front-end
IoT stack(s)
Cortex-M0+
Legacy bare-
metal
codebase
ICode
DCode
System
System
UART1
UART2
Flash
M4 Image
FreeRTOS_M4
GUI
web front-end
IoT stack
M0+ bare-
metal mage
legacy code
RAM
M4 memory
stack, heap,
globals, task
stacks
M0+ memory
legacy code
(stack,heap
globals)
Shared memory
buffers, data structures
Brian Amos

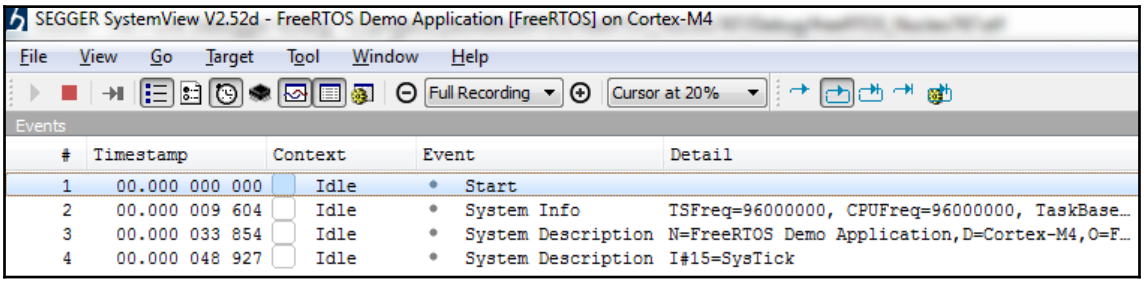

i.Mx8 Application Example
HTTP
MQTT
IoT
Cortex-A72
Cortex-A53
GPOS (i.e. Linux)
FreeRTOS
FreeRTOS
Cortex-M4
Cortex-M4
i.Mx8
pos
time
pos
time
Brian Amos

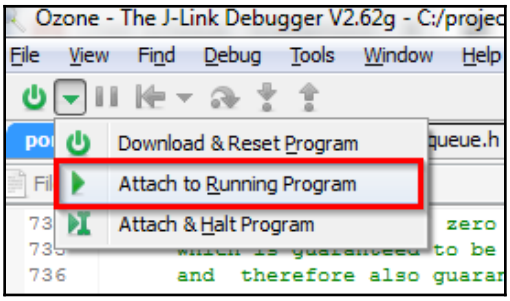

Distributed System
Example
MCU
MCU
MCU
MCU
Inter-processor
Communication
Bus
Brian Amos

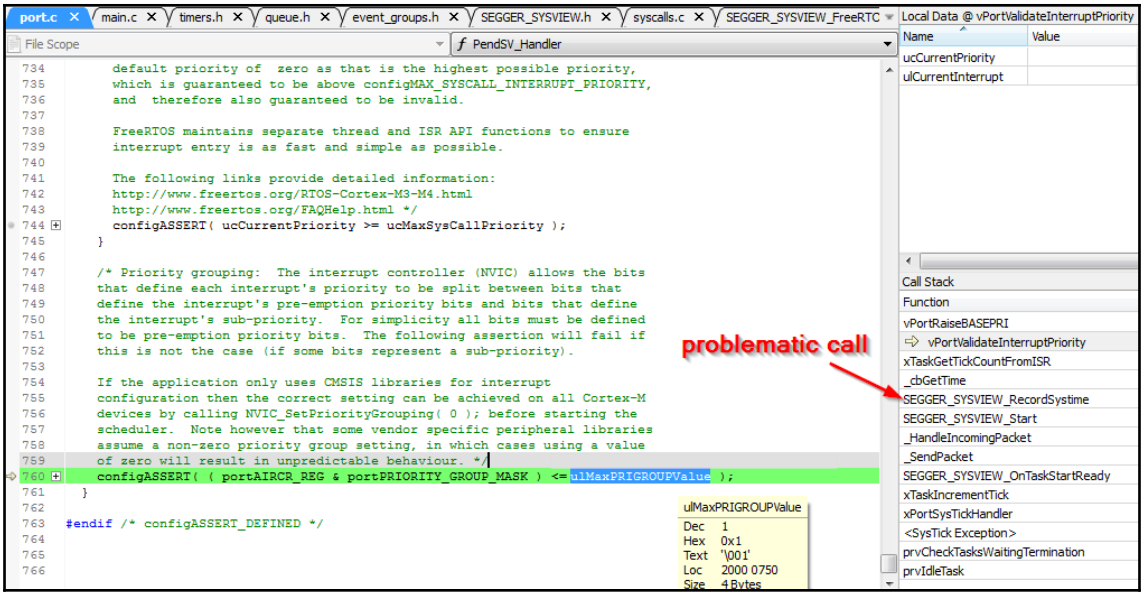

Distributed System
Parallel Development
Architecture/
Planning
Subsystem
HW/FW
Dev/Test
System
Integration
Team 1
Team 2
Team 3
MCU
MCU
MCU
MCU
MCU
MCU
time
Brian Amos

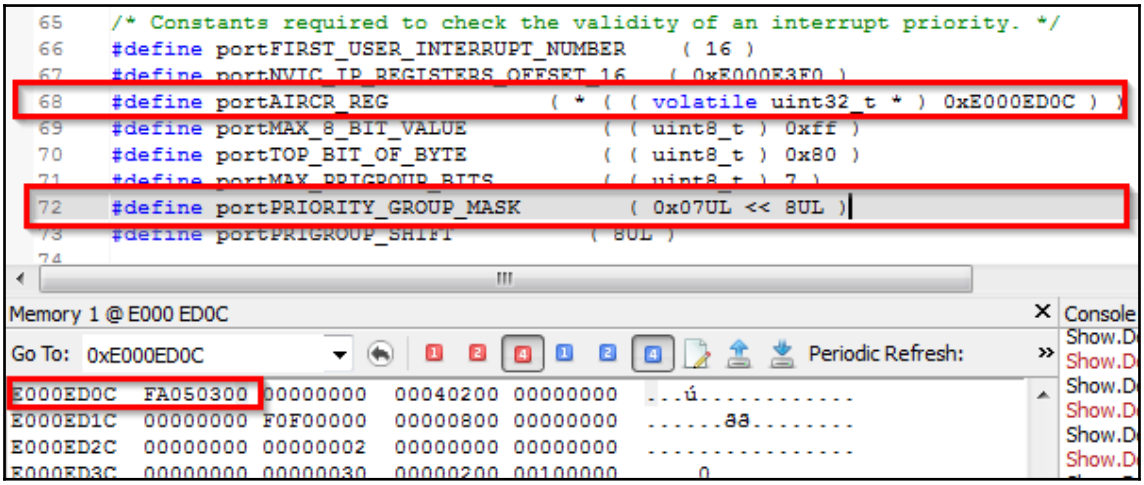
Chapter 17: Troubleshooting Tips and Next Steps

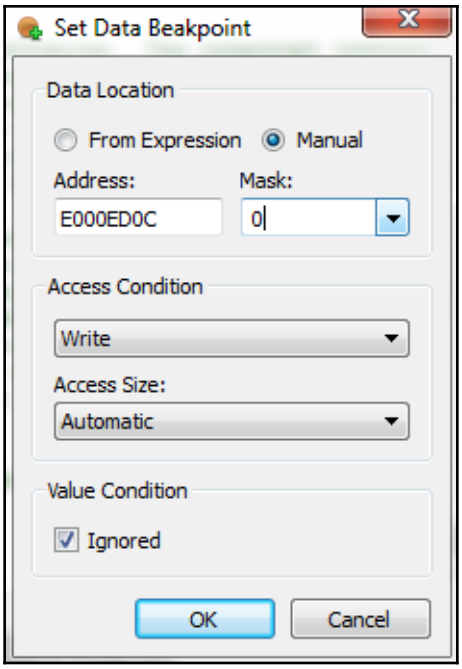


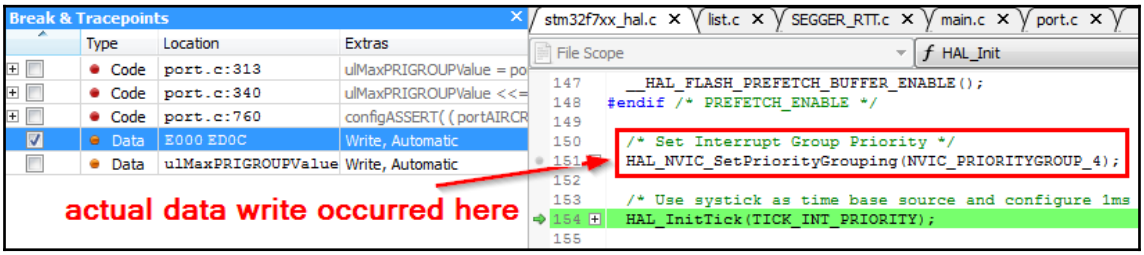





port.c
main.c
timers.h
queue.h
event_groups.h
SEGGER_SYSVIEW.h
syscalls.c
SEGGER_SYSVIEW_FreeRTC
File Scope
PendSV_Handler
default priority of zero as that is the highest possible priority,
which is guaranteed to be above configMAX_SYSCALL_INTERRUPT_PRIORITY,
and therefore also guaranteed to be invalid.
FreeRTOS maintains separate thread and ISR API functions to ensure
interrupt entry is as fast and simple as possible.
The following links provide detailed information:
http://www.freertos.org/RTOS-Cortex-M3-M4.html
http://www.freertos.org/FAQHelp.html */
configASSERT(ucCurrentPriority >= ucMaxSysCallPriority);
}
/* Priority grouping: The interrupt controller (NVIC) allows the bits
that define each interrupt's priority to be split between bits that
define the interrupt's pre-emption priority bits and bits that define
the interrupt's sub-priority. For simplicity all bits must be defined
to be pre-emption priority bits. The following assertion will fail if
this is not the case (if some bits represent a sub-priority).
If the application only uses CMSIS libraries for interrupt
configuration then the correct setting can be achieved on all Cortex-M
devices by calling NVIC_SetPriorityGrouping(0); before starting the
scheduler. Note however that some vendor specific peripheral libraries
assume a non-zero priority group setting, in which cases using a value
of zero will result in unpredictable behaviour. */
configASSERT((portAIRCR_REG & portPRIORITY_GROUP_MASK) <= ulMaxPRIGROUPValue);
}
#endif /* configASSERT_DEFINED */
ulMaxPRIGROUPValue
Dec 1
Hex 0x1
Text '\001'
Loc 2000 0750
Size 4 Bytes
Local Data @ vPortValidateInterruptPriority
Name
Value
ucCurrentPriority
ulCurrentInterrupt
Call Stack
Function
vPortRaiseBASEPRI
vPortValidateInterruptPriority
xTaskGetTickCountFromISR
_cbGetTime
SEGGER_SYSVIEW_RecordSystime
SEGGER_SYSVIEW_Start
_HandleIncomingPacket
_SendPacket
SEGGER_SYSVIEW_OnTaskStartReady
xTaskIncrementTick
xPortSysTickHandler
<SysTick Exception>
prvCheckTasksWaitingTermination
prvIdleTask
problematic call


/* Constants required to check the validity of an interrupt priority. */
#define portFIRST_USER_INTERRUPT_NUMBER (16)
#define portNVIC_IP_REGISTERS_OFFSET_16 (0xE000E3F0)
#define portAIRCR_REG (* ((volatile uint32_t *) 0xE000ED0C))
#define portMAX_8_BIT_VALUE ((uint8_t) 0xff)
#define portTOP_BIT_OF_BYTE ((uint8_t) 0x80)
#define portMAX_PRIGROUP_BITS ((uint8_t) 7)
#define portPRIORITY_GROUP_MASK (0x07UL << 8UL)
#define portPRIGROUP_SHIFT (8UL)
Memory 1 @ E000 ED0C
Go To: 0xE000ED0C
Periodic Refresh:
E000ED0C FA050300 00000000 00040200 00000000 ...ú.....
E000ED1C 00000000 F0F00000 00000800 00000000 .....ðð.....
E000ED2C 00000000 00000002 00000000 00000000 .....
E000ED3C 00000000 00000030 00000200 00100000 ...0





ARM Cortex-M7 Devices Generic User Guide

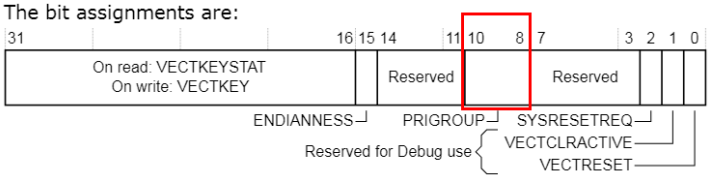
Home > Cortex-M7 Peripherals > System control block > Application Interrupt and Reset Control Register

4.3.5. Application Interrupt and Reset Control Register

The AIRCR provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. See the register summary in Table 4.12 and Table 4.17 for its attributes.

To write to this register, you must write `0x5FA` to the VECTKEY field, otherwise the processor ignores the write.

The bit assignments are:



Binary point

The PRIGROUP field indicates the position of the binary point that splits the PRI_*n* fields in the Interrupt Priority Registers into separate *group priority* and *subpriority* fields. Table 4.18 shows how the PRIGROUP value controls this split. Implementations having PRI_*n* fields of less than 8 bits treat the least-significant bits as zero.

Table 4.18. Priority grouping

	Interrupt priority level value, PRI_*N*[7:0]			Number of	
PRIGROUP	**Binary point[a]**	**Group priority bits**	**Subpriority bits**	**Group priorities[b]**	**Subpriorities[b]**
0b000[c]	bxxxxxxx.y	[7:1]	[0]	128	2
0b001[c]	bxxxxxx.yy	[7:2]	[1:0]	64	4
0b010[c]	bxxxxx.yyy	[7:3]	[2:0]	32	8
0b011[c]	bxxxx.yyyy	[7:4]	[3:0]	16	16
0b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
0b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
0b110	bx.yyyyyyy	[7]	[6:0]	2	128
0b111	b.yyyyyyyy	None	[7:0]	1	256

[a] PRI_*n*[7:0] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit.