



Building Blocks for PRU Development

Module 3

Designing a PRU Application

This session covers how to design a PRU application using a step-by-step design process. The module also walks through this process with an example PRU application, demonstrating how to design a Temperature Monitor application with the PRU.

Author: Texas Instruments®, Sitara™ ARM® Processors

Oct 2014

Creative Commons Attribution-ShareAlike 3.0 (CC BY-SA 3.0)



You are free:

to **Share** – to copy, distribute and transmit the work

to **Remix** – to adapt the work

to make commercial use of the work

Under the following conditions:



Attribution – You must give the original author(s) credit



Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

With the understanding that:

Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

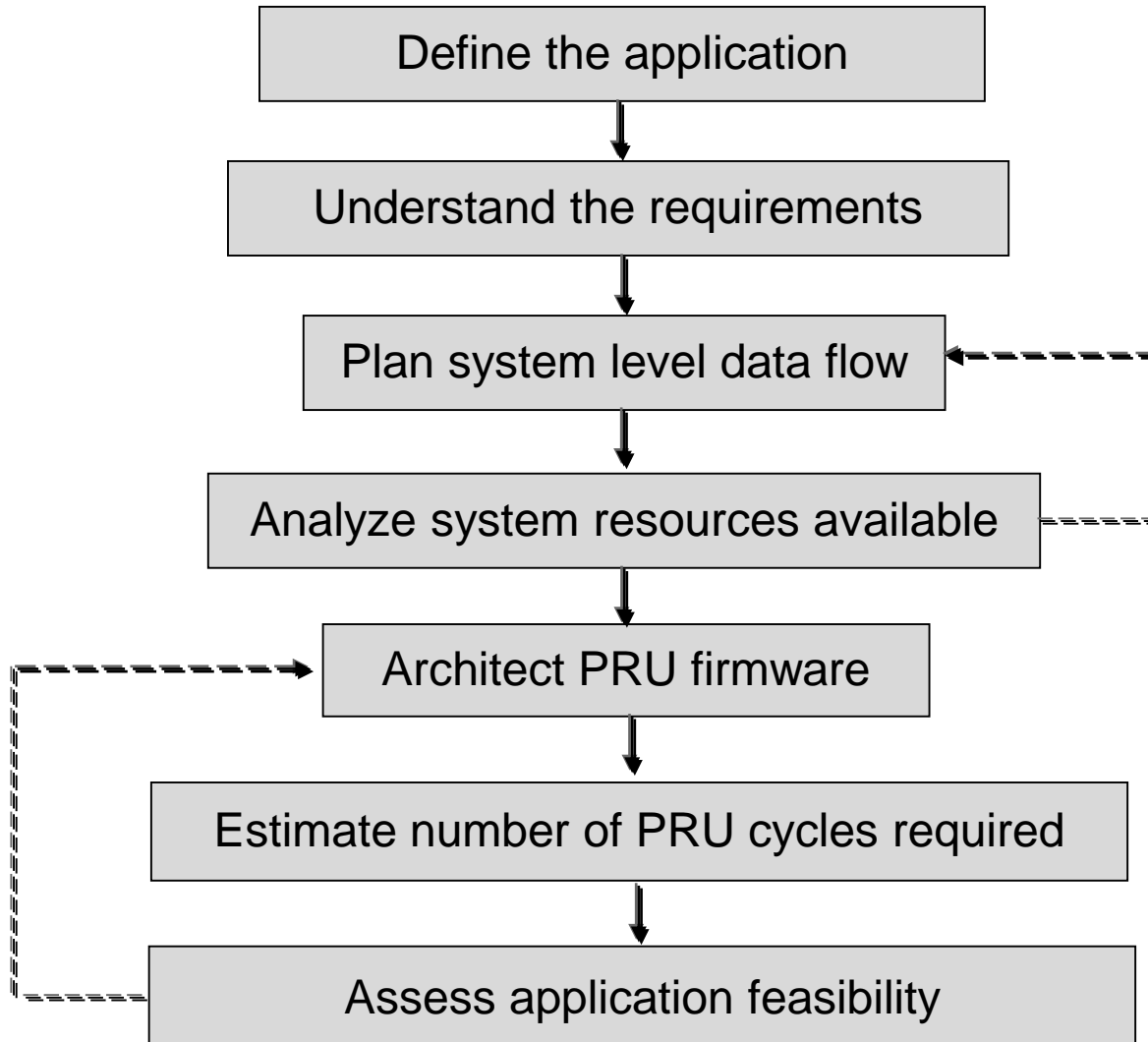


CC BY-SA 3.0 License:

<http://creativecommons.org/licenses/by-sa/3.0/us/legalcode>



PRU Design Steps



**Now let's design
a PRU application...**



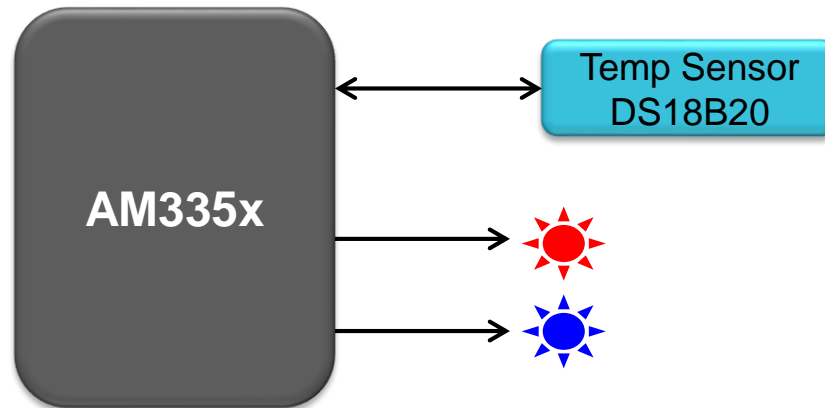
Step 1...



Step 1: Define the application

Temperature monitor

- Periodically monitor ambient temp using external sensor
- Illuminate blue or red LED based on temp change



Step 2...



Step 2: Understand the Requirements

1. Identify required I/Os

- * *How many?*
- * *What type? (i.e. uni or bidirectional, serial, parallel, etc)*
- * *Any special requirements? (i.e. open-drain, etc)*

2. Understand protocol basics

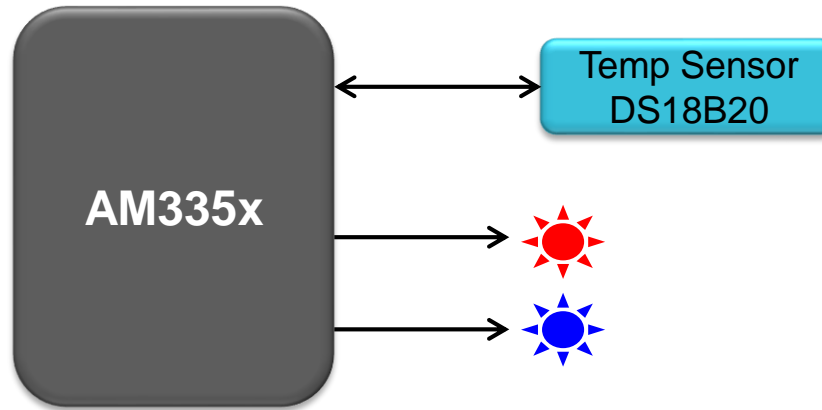
- * *Are any required protocols not available in hardware or not already implemented in software?*
- * *If so, what are basic building blocks of the protocol?*

3. Outline general data flow

- * *What tasks are required by the defined application?*

Step 2: Understand the Requirements

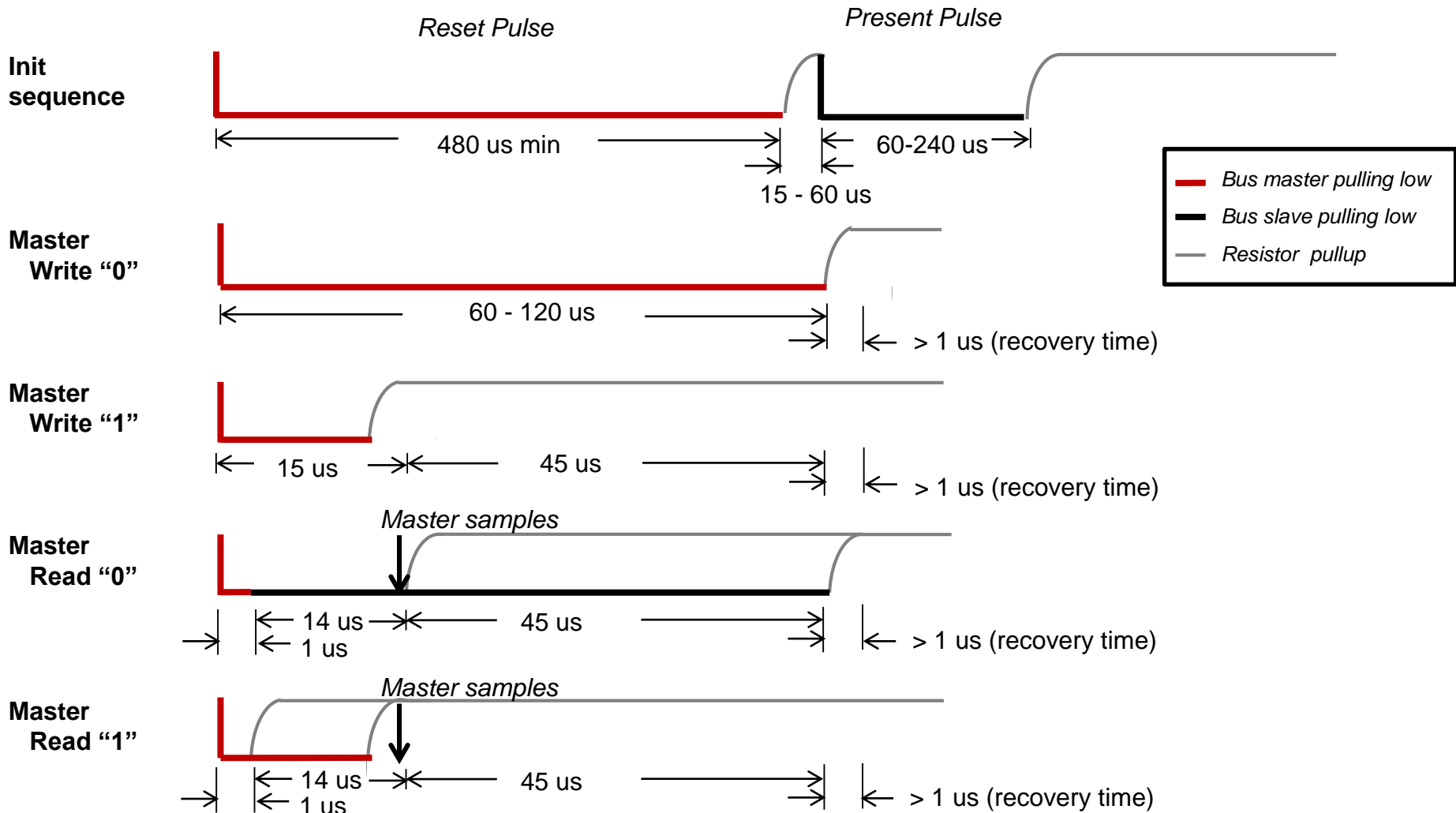
1. Identify required I/Os



I/O	I/O Type	#
Control & toggle LEDs	Output (unidirectional)	2
1-Wire interface with temp sensor	IO (bidirectional), open-drain	1

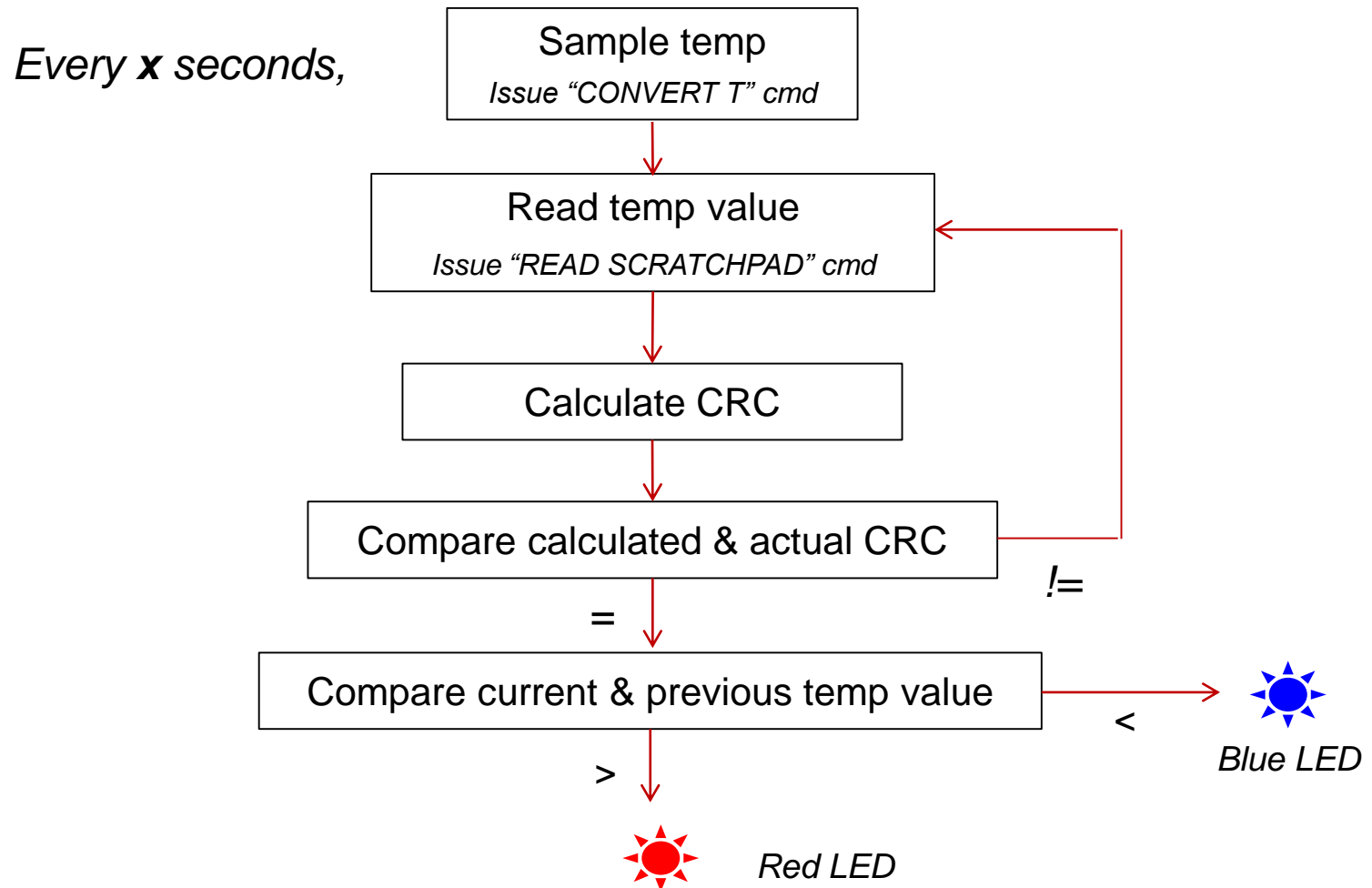
Step 2: Understand the Requirements

2. Understand protocol basics



Step 2: Understand the Requirements

2. Outline general data flow



Step 3...



Step 3: Plan system-level data flow

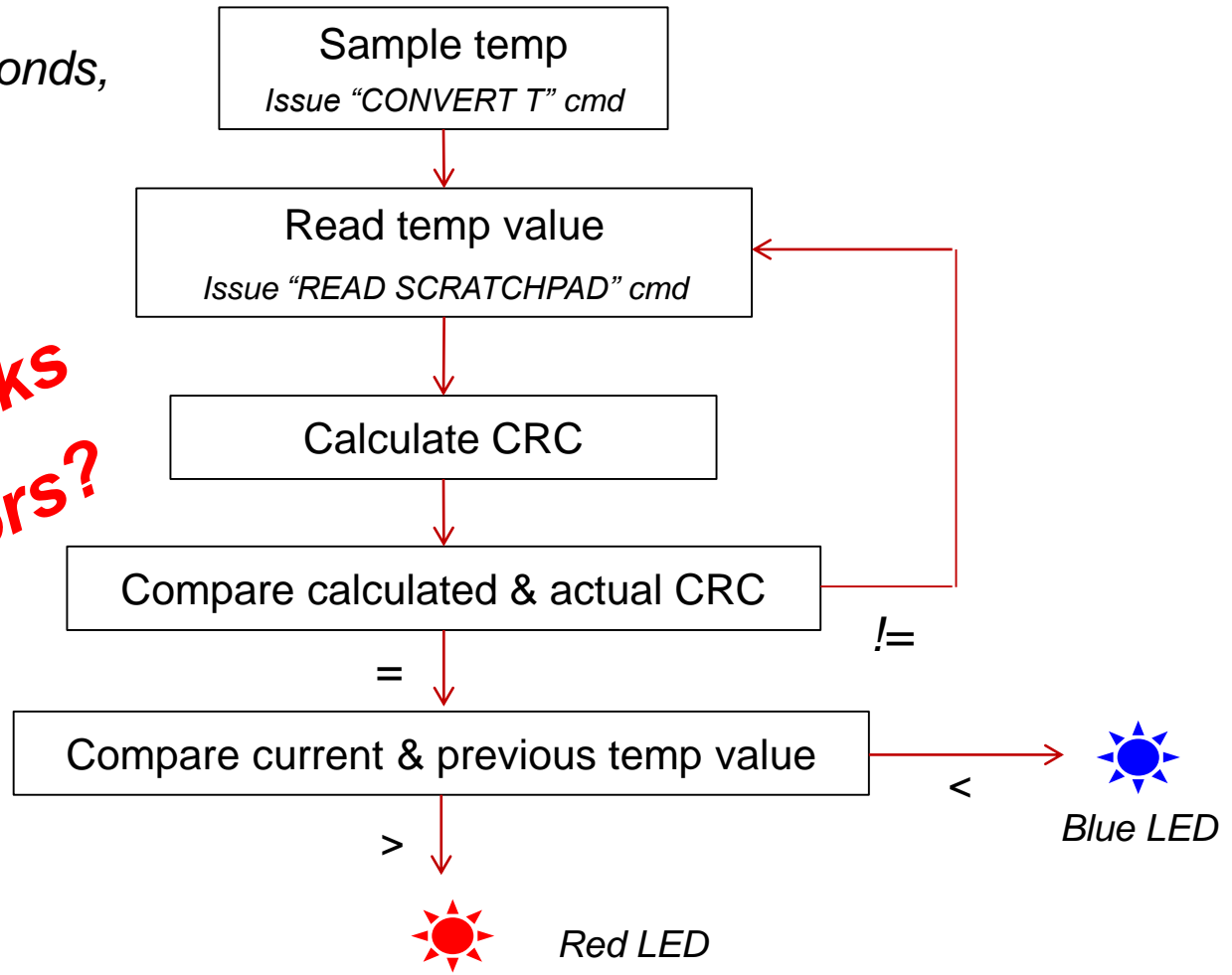
- Decide target processor(s) to implement Step 2 tasks

For example: ARM
 Single PRU core
 Dual PRU core

- May need to revisit decision after future steps

Step 3: Plan system-level data flow

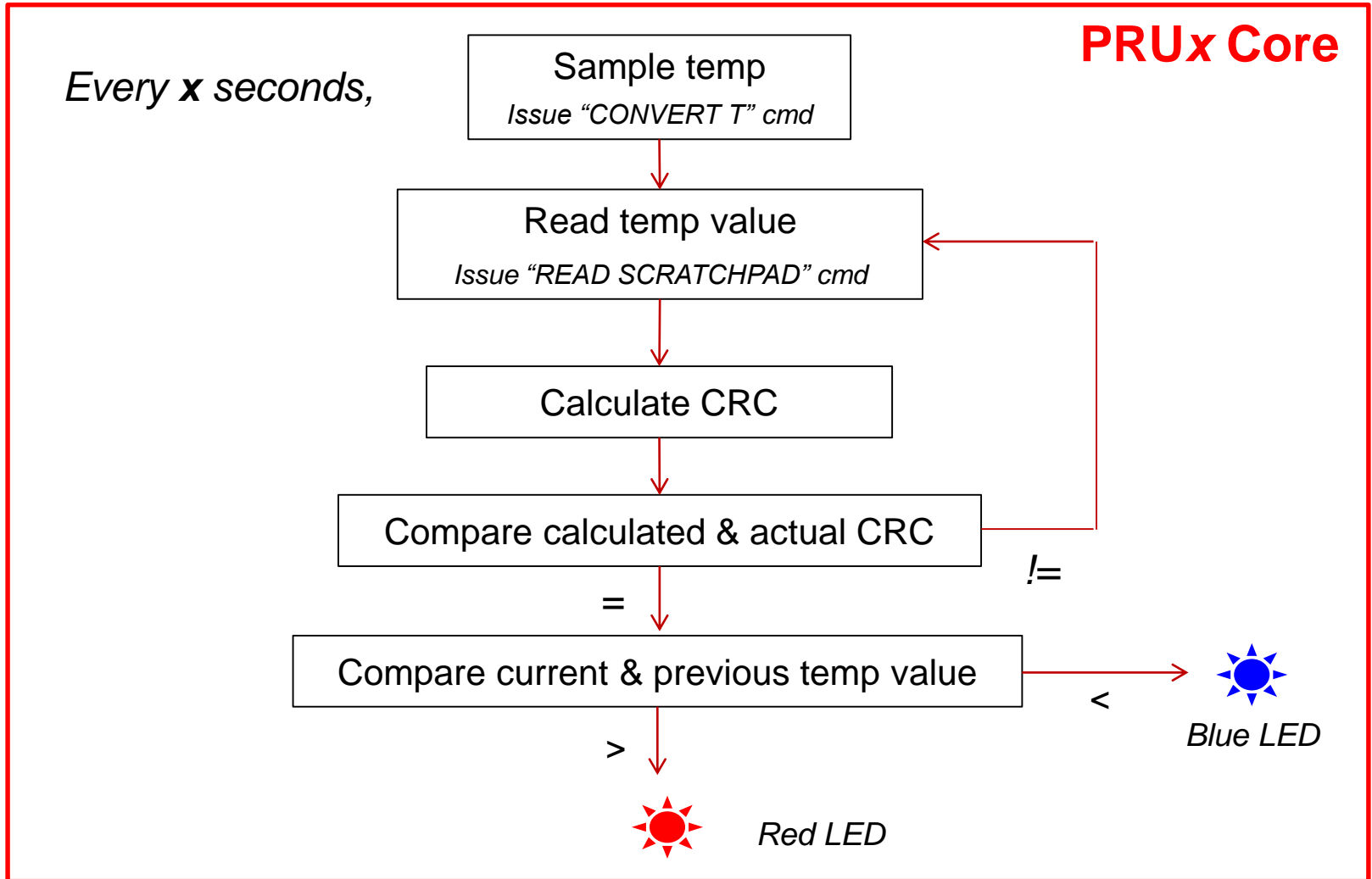
Every x seconds,



How would you divide these tasks btw processors?

Step 3: Plan system-level data flow

Here's one option...



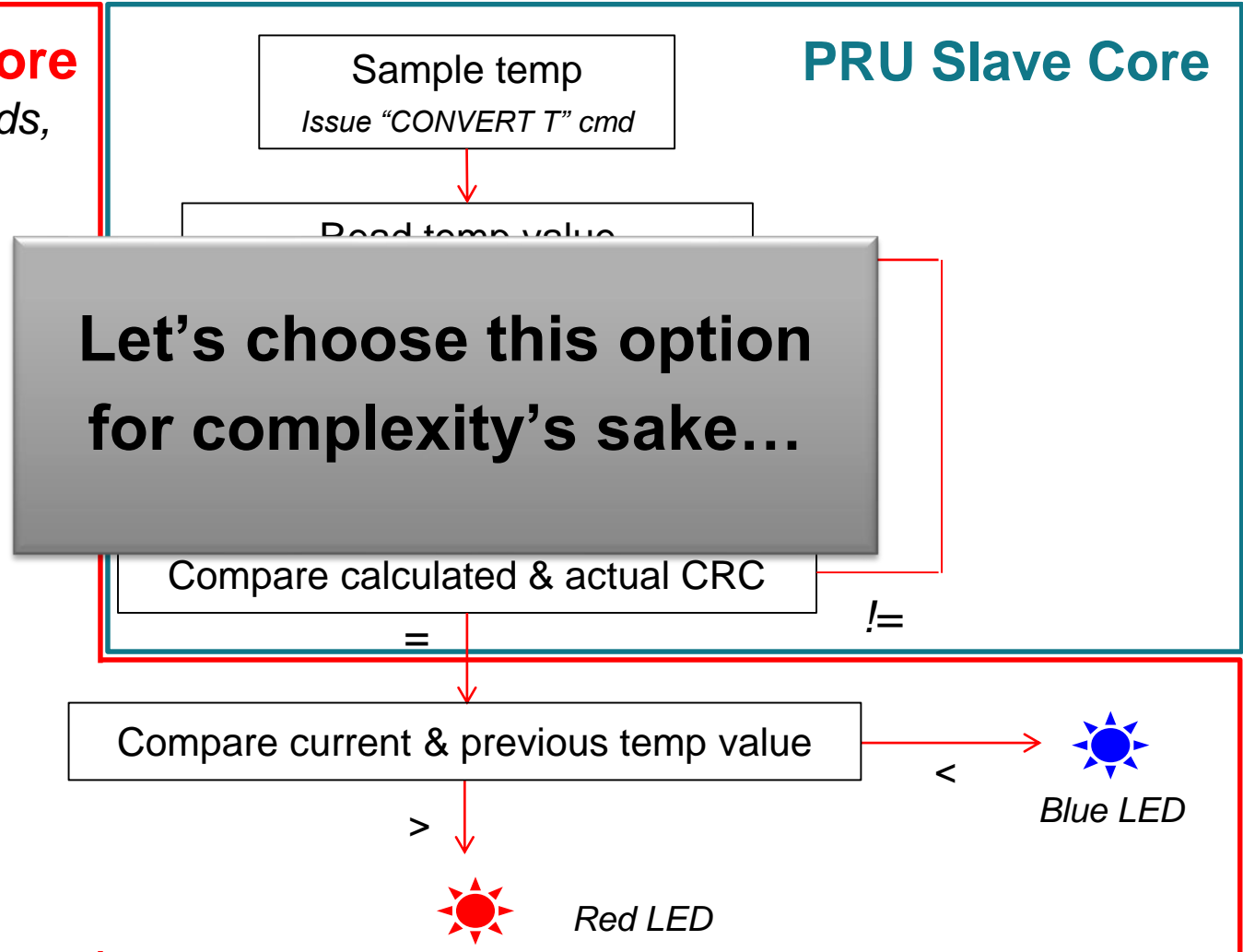
Step 3: Plan system-level data flow

Here's another option...

PRU Master Core

Every x seconds,

PRU Slave Core



Step 4...



Step 4: Analyze the available SOC resources

- **Peripherals & I/Os**
- **System Event & Interrupt Sources**
- **Memory Resources**

General Rule of Thumb:

Whenever possible, select resources within PRU subsystem to reduce access latency.

Step 4: Analyze the available SOC resources

- Peripherals & I/Os

#	I/O Type	Purpose
2	Output (unidirectional)	Control & toggle LEDs
1	IO (bidirectional), open-drain	1-Wire interface with temp sensor

***Which PRU / AM335x I/Os
would you choose
for each type of I/O?***

Step 4: Analyze the available SOC resources

- Peripherals & I/Os

#	I/O Type	Purpose
2	Output (unidirectional)	Control & toggle LEDs

- PRU GPO (R30)
 - Fast, ultra-low latency, simple to program
 - PRU core dependent
- PRU IEP Digital I/Os
 - Low latency, PRU core independent
 - Limited pins
- Standard AM335x GPIOs
 - PRU core independent, “unlimited” pins
 - High latency

Step 4: Analyze the available SOC resources

- Peripherals & I/Os

#	I/O Type	Purpose
2	Output (unidirectional)	Control & toggle LEDs

- PRU GPO (R30)
 - Fast, ultra-low latency, simple to program
 - PRU core dependent
- PRU IEP Digital I/Os
 - Low latency, PRU core independent
 - Limited pins
- Standard AM335x GPIOs
 - PRU core independent, “unlimited” pins
 - High latency

**Let's choose the PRU GPO
for its ultra-low latency
and simplicity to program...**

Step 4: Analyze the available SOC resources

- **Peripherals & I/Os**

#	I/O Type	Purpose
1	IO (bidirectional), open-drain	1-Wire interface with temp sensor

- **PRU GPO (R30)**
 - Ultra-low latency
 - Unidirectional, not open-drain, PRU core dependent
- **PRU IEP Digital I/Os**
 - Open-drain, low latency, PRU core independent
 - Unidirectional
- **Standard AM335x GPIOs**
 - Bidirectional, open-drain
 - High latency

Step 4: Analyze the available SOC resources

- **Peripherals & I/Os**

#	I/O Type	Purpose
1	IO (bidirectional), open-drain	1-Wire interface with temp sensor

- **PRU GPO (R30)**
 - Ultra-low latency
 - Unidirectional, not open-drain, PRU core dependent
- **PRU IEP Digital I/Os**
 - Open-drain, low latency, PRU core independent
 - Unidirectional

There isn't an option that fulfills all I/O requirements (bidirectional & open-drain).

However, we can achieve a bidirectional signal by externally tying together an input and an output signal.

Therefore, let's choose to use PRU IEP Digital I/Os for their open-drain capability...

Step 4: Analyze the available SOC resources

- Peripherals & I/Os

#	I/O Type	Purpose	Selected Resource
2	Output (unidirectional)	LED	PRU GPO (2)
1	IO (bidirectional), open-drain	Temp sensor	PRU IEP DigIO (2)

***Next, we need to check
if there are any I/O restrictions
due to pin muxing...***

Step 4: Analyze the available SOC resources

- Peripherals & I/Os

#	I/O Type	Purpose	Selected Resource
2	Output (unidirectional)	LED	PRU GPO (2)
1	IO (bidirectional), open-drain	Temp sensor	PRU IEP DigIO (2)

PRU IEP DigIO Pinmux Analysis

AM335x Pins	BBB Header label	MUX MODE A	MUX MODE B	PRU Cape function
B16	I2C1_SDA	PR1_EDIO_DATA_IN0	PR1_EDIO_DATA_OUT0	LCD_DATA4
			EDIO_DATA_OUT1	LCD_DATA5
			EDIO_DATA_OUT2	LCD_DATA6
			EDIO_DATA_OUT3	LCD_DATA7
			EDIO_DATA_OUT4	LDC_F
			EDIO_DATA_OUT5	
			EDIO_DATA_OUT6	eMMC - BBB
			EDIO_DATA_OUT6	LCD_RS
T4	LCD_DATA7	PR1_EDIO_DATA_IN7	PR1_EDIO_DATA_OUT7	Conflicts with bootpins

There is only **one** DigIO available, but we were needing **two** for the bidirectional Temp Sensor interface.

Time to re-evaluate what resources can be used instead...

Only 1 DigIO available

Step 4: Analyze the available SOC resources

- Peripherals & I/Os

#	I/O Type	Purpose
1	IO (bidirectional), open-drain	1-Wire interface with temp sensor

***Let's use the available DigIO pin
as the open-drain output of the
bidirectional signal.***

***Which PRU / AM335x I/O
would you choose for the input?***


Step 4: Analyze the available SOC resources

- Peripherals & I/Os

#	I/O Type	Purpose
1	IO (bidirectional), open-drain	1-Wire interface with temp sensor

- PRU GPI (R31)
 - Ultra-low latency
 - Unidirectional, not open-drain, PRU core dependent
- PRU IEP Digital Output
 - Open-drain, low latency, PRU core independent
 - Unidirectional
- Standard AM335x GPIOs
 - Bidirectional, open-drain
 - High latency

Pinmux restricts us to PRU1



Step 4: Analyze the available SOC resources

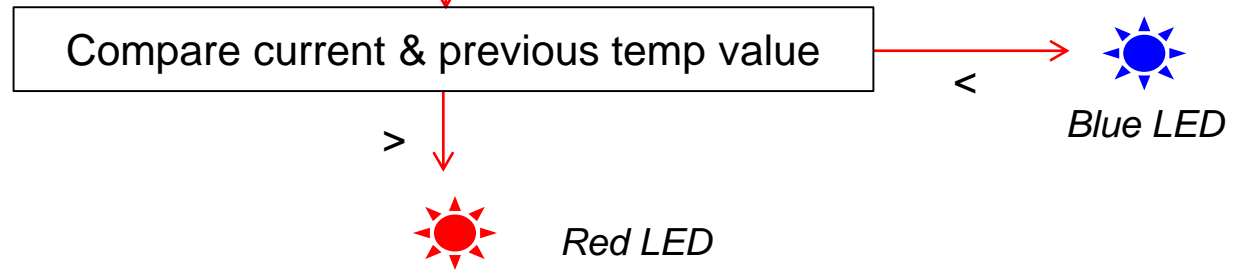
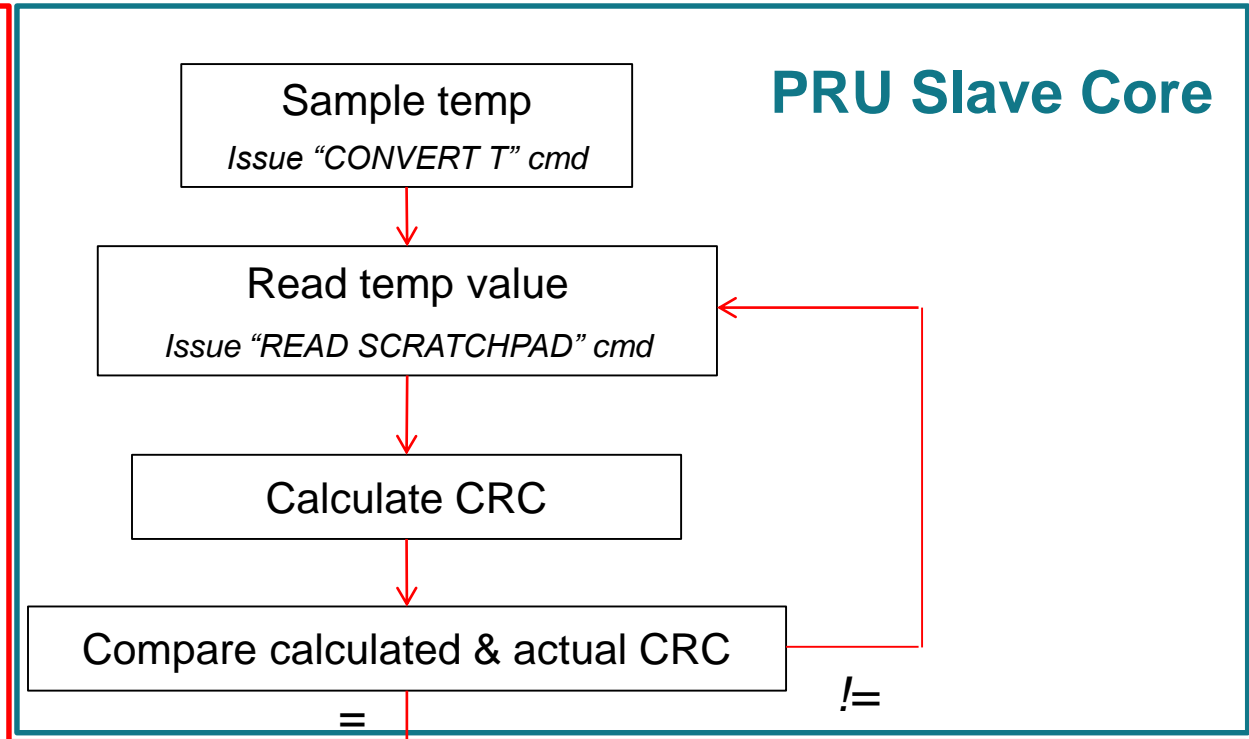
- System Event & Interrupt Sources

*What type of interrupts
are needed?*

Step 4: Analyze the available SOC resources

PRU Master Core

Every x seconds,



Step 4: Analyze the available SOC resources

- System Event & Interrupt Sources

Event Type	Details
Periodic system timer	x seconds (i.e. 3s)
1-wire timer	1, 15, 60, 240 us
PRU to PRU	

Step 4: Analyze the available SOC resources

System Event & Int Sources

Event Type

Periodic system timer

1-wire timer

What type of system event(s) would you choose?

Int Number	Signal Name (Standard Mode)	Source
63	tpcc_int_pend_po1	TPCC (EDMA)
62	tpcc_errint_pend_po	TPCC (EDMA)
61	tptc_erint_pend_po	TPTC0 (EDMA)
60	initiator_sinterrupt_q_n1	Mbox0 - mail_u1_irq (mailbox interrupt for pru0)
59	initiator_sinterrupt_q_n2	Mbox0 - mail_u2_irq (mailbox interrupt for pru1)
58	Emulation Suspend Signal (software use)	Debugss
57	POINTRPEND1	GPIO0
56	pwm_trip_zone	eHRPWM0/eHRPWM1/eHRPWM2
55	mcasp_x_intr_pend	McASP0 Tx
54	mcasp_r_intr_pend	McASP0 Rx
53	gen_intr_pend	ADC_TSC
52	nirq	UART2
51	nirq	UART0
50	c0_rx_thresh_pend	3PGSW (GEMAC)
49	c0_rx_pend	3PGSW (GEMAC)
48	c0_tx_pend	3PGSW (GEMAC)
47	c0_misc_pend	3PGSW (GEMAC)
46	epwm_intr_intr_pend	eHRPWM1
45	eqep_intr_intr_pend	eQEP0
44	SINTERRUPTN	McSPI0
43	epwm_intr_intr_pend	eHRPWM0
42	ecap_intr_intr_pend	eCAP0
41	POINTRPEND	I2C0
40	dcan_intr	DCAN0
39	dcan_int1	DCAN0
38	dcan_uerr	DCAN0
37	epwm_intr_intr_pend	eHRPWM2
36	ecap_intr_intr_pend	eCAP2
35	ecap_intr_intr_pend	eCAP1
34	mcasp_r_intr_pend	McASP1 Rx
33	mcasp_x_intr_pend	McASP1 Tx
32	nirq	UART1

17	pr1_pru_mst_intr[1]_intr_req	pru0 or pru1
16	pr1_pru_mst_intr[0]_intr_req	pru0 or pru1
15	pr1_ecap_intr_req	PRU-ICSS eCAP
14	Reserved	Reserved
13	Reserved	Reserved
12	Reserved	Reserved
11	Reserved	Reserved
10	Reserved	Reserved
9	Reserved	Reserved
8	pr1_digio_event_req	PRU-ICSS IEP
7	pr1_iep_tim_cap_cmp_pend	PRU-ICSS IEP
6	pr1_uart_uint_intr_req	PRU-ICSS UART
5	pr1_uart_utxevt_intr_req	PRU-ICSS UART
4	pr1_uart_urxevt_intr_req	PRU-ICSS UART
3	pr1_xfr_timeout	PRU-ICSS Scratch Pad
2	pr1_pru1_r31_status_cnt16	PRU-ICSS PRU1 (Shift Capture)
1	pr1_pru0_r31_status_cnt16	PRU-ICSS PRU0 (Shift Capture)
0	pr1_parity_err_intr_pend	PRU-ICSS Parity Logic

Step 4: Analyze the available SOC resources

- System Event & Interrupt Sources

Event Type	Details	Selected Resource
Periodic system timer	i.e. 3s	
1-wire timer	1, 15, 60, 240 us	

- PRU IEP Timer
 - Up to 8 compare values
- PRU eCAP (auxiliary PWM mode)
- Standard AM335x PWM
- Standard AM335x eCAP (APWM mode)
- Standard AM335x DMTimer
 - Manually poll by reading DMTimer status registers

Step 4: Analyze the available SOC resources

- System Event & Interrupt Sources

Event Type	Details	Selected Resource
Periodic system timer	i.e. 3s	PRU eCAP
1-wire timer	1, 15, 60, 240 us	PRU IEP Timer

- PRU IEP Timer
 - Up to 8 compare values
- PRU eCAP (auxiliary PWM mode)
- Standard AM335x PWM
- Standard AM335x eCAP (APWM mode)
- Standard AM335x DMTimer
 - Manually poll by reading DMTimer status registers

Mapped to PRU Master Core

Mapped to PRU Slave Core

Remember: System Events can only be routed to one Host Interrupt.

Step 4: Analyze the available SOC resources

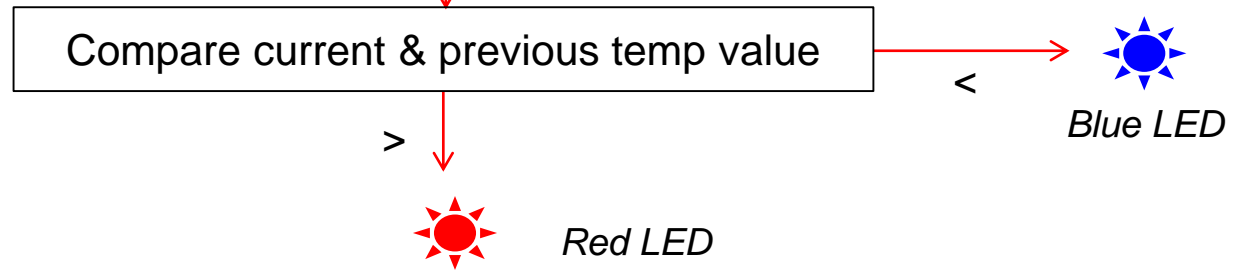
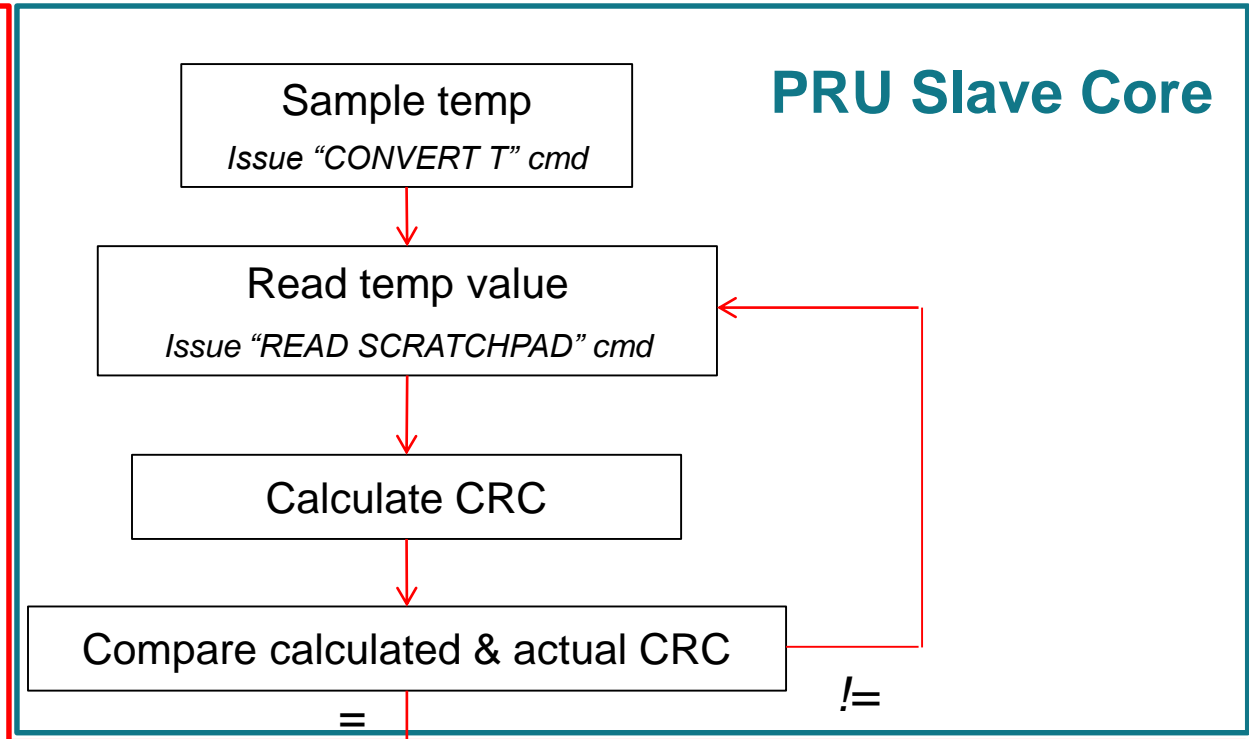
- Memory Resources

*What memory resources are needed
&
which PRU / AM335x memory space(s)
would you chose?*

Step 4: Analyze the available SOC resources

PRU Master Core

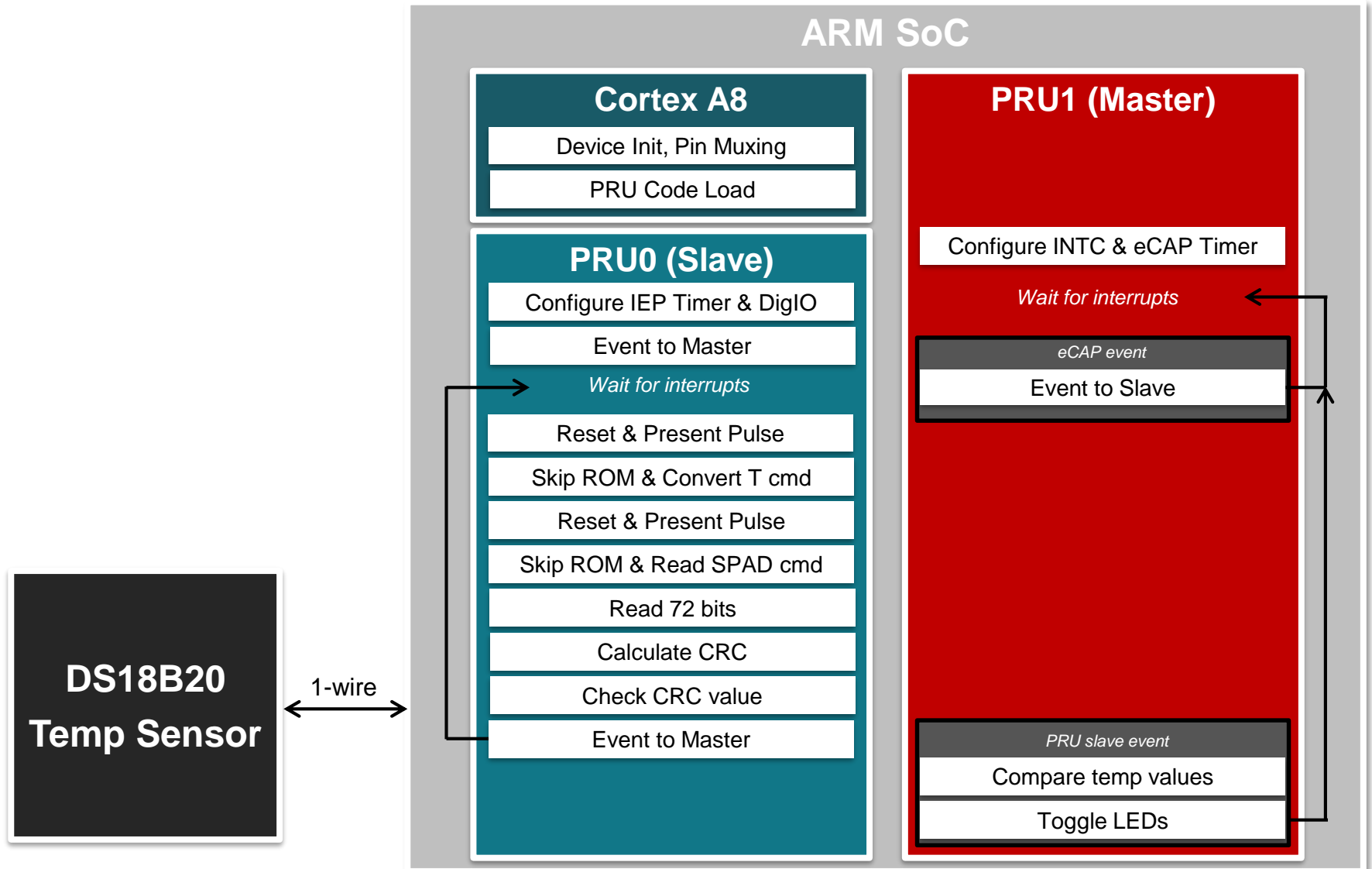
Every x seconds,



Step 5...



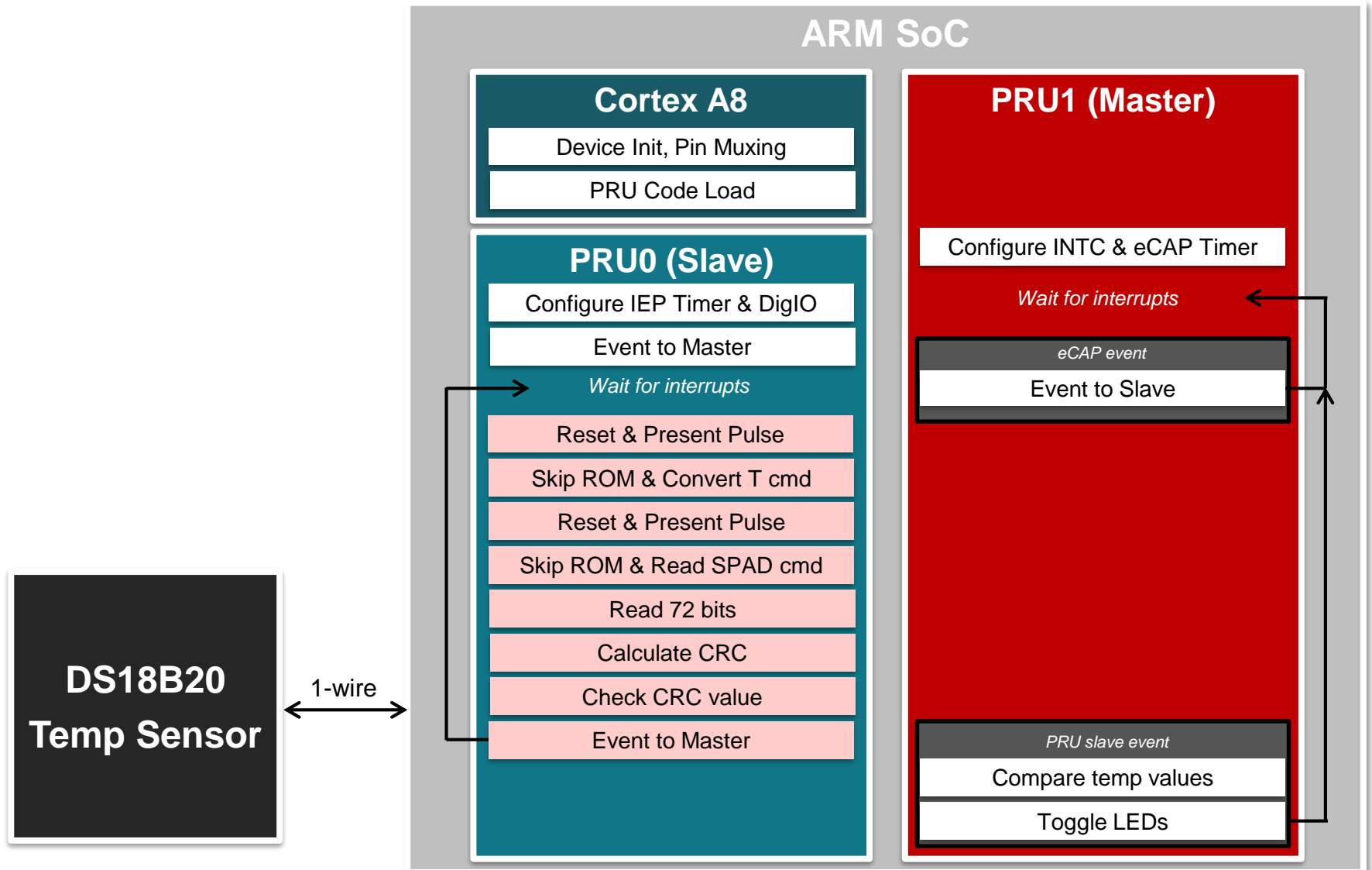
Step 5: Architect PRU firmware & ARM code



Step 6...

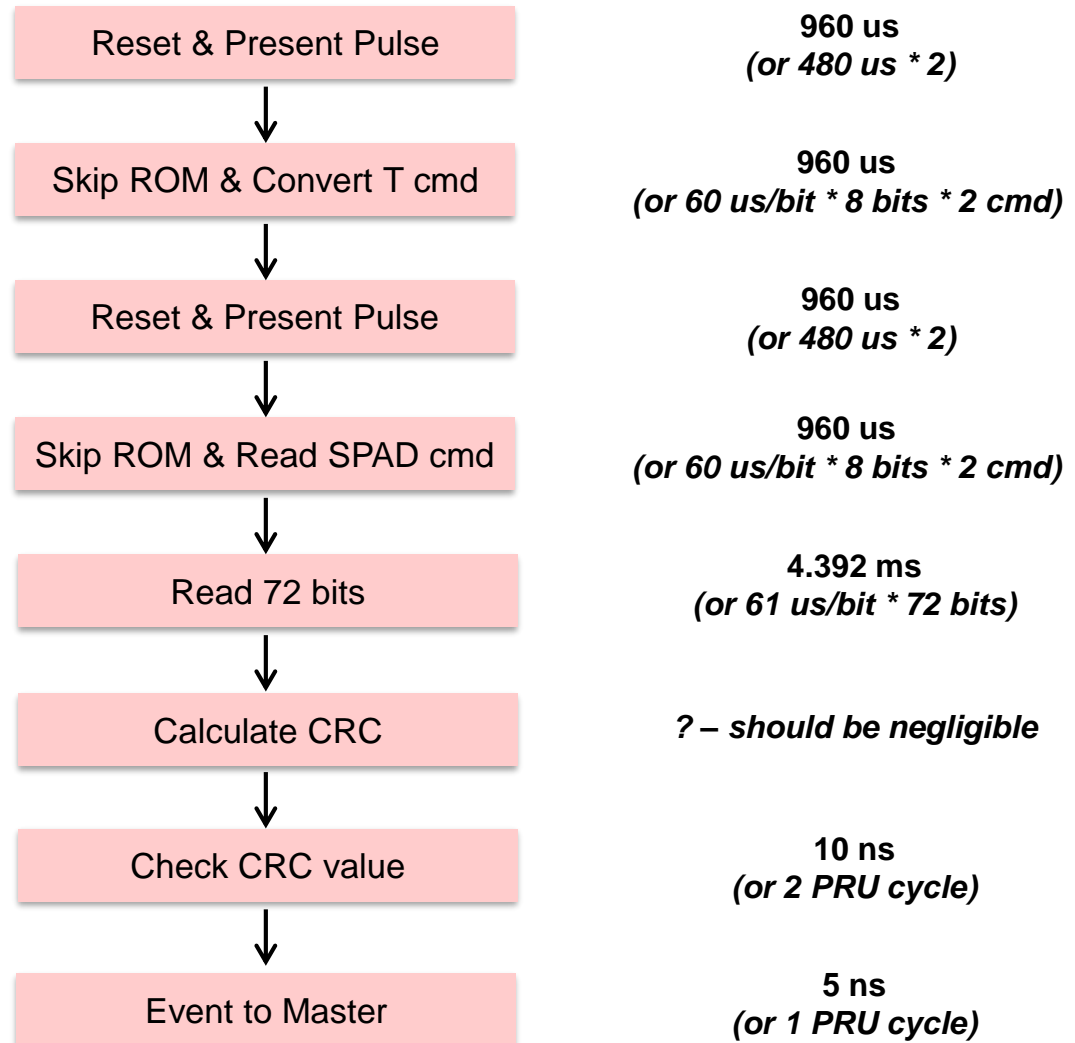


Step 6: Estimating PRU Cycles



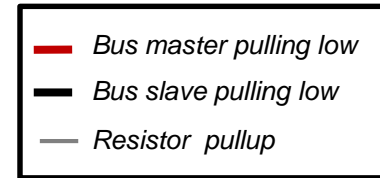
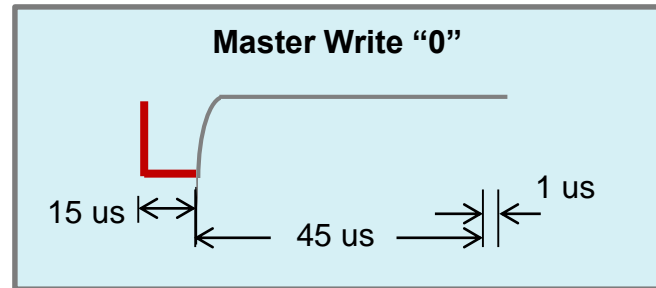
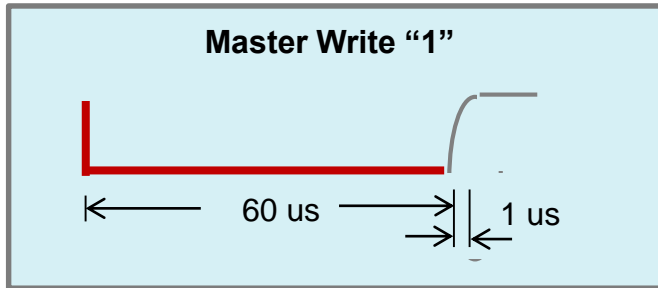
Step 6: Estimating PRU Cycles

Estimation

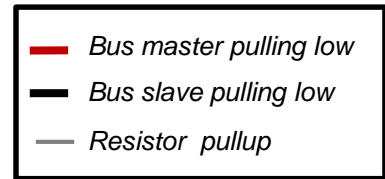
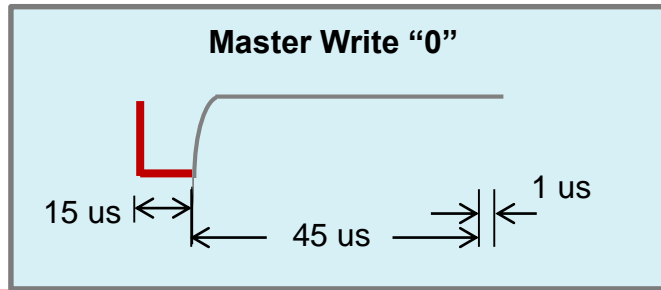
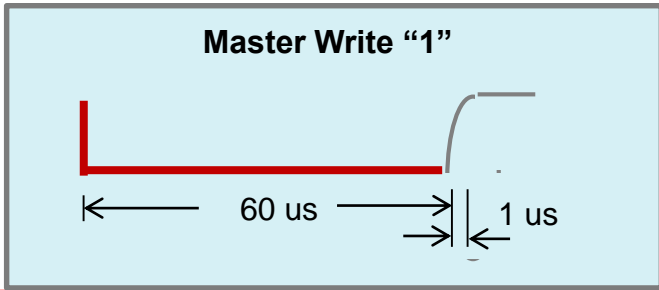
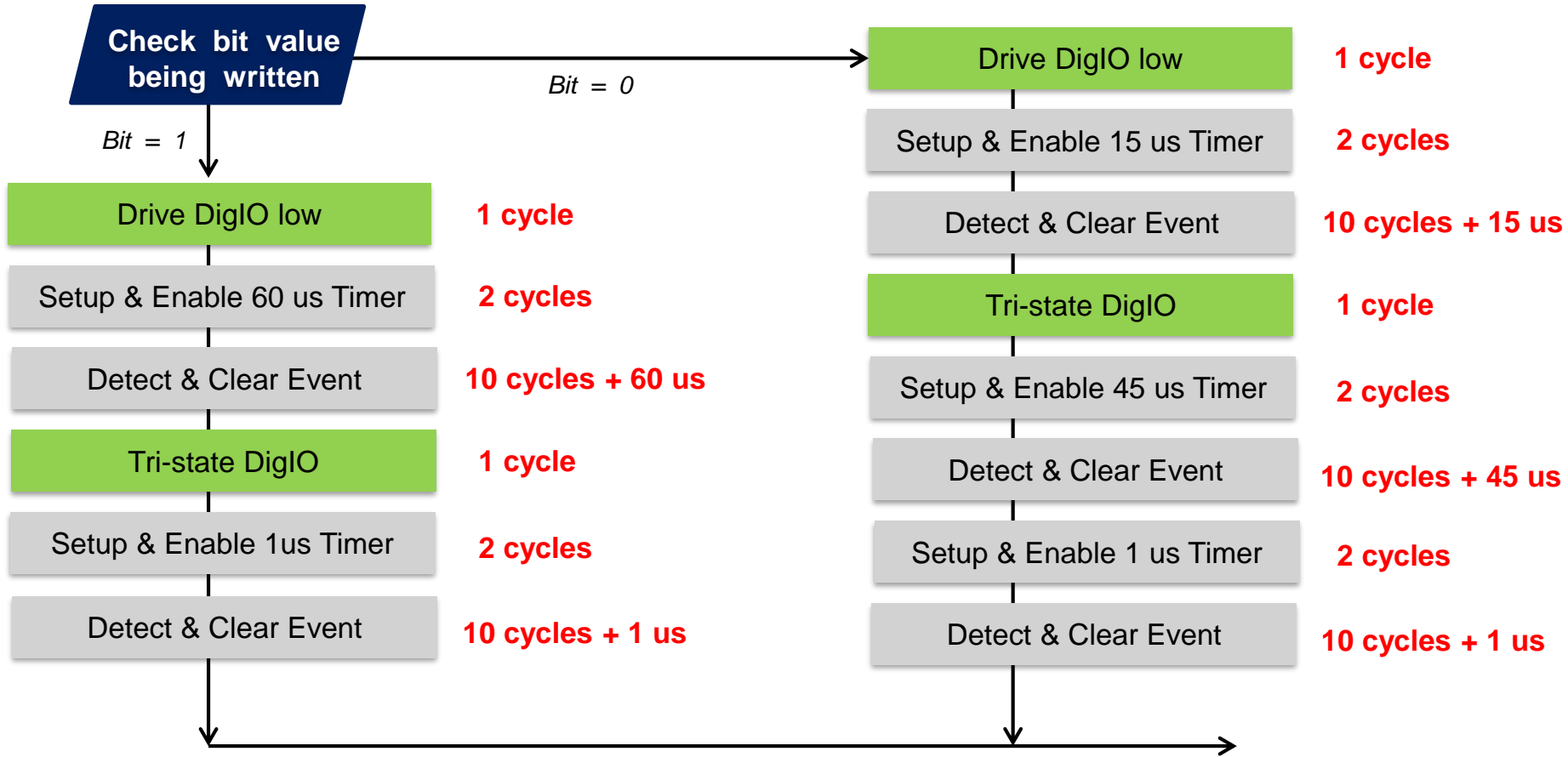


~ 8.232 ms + change

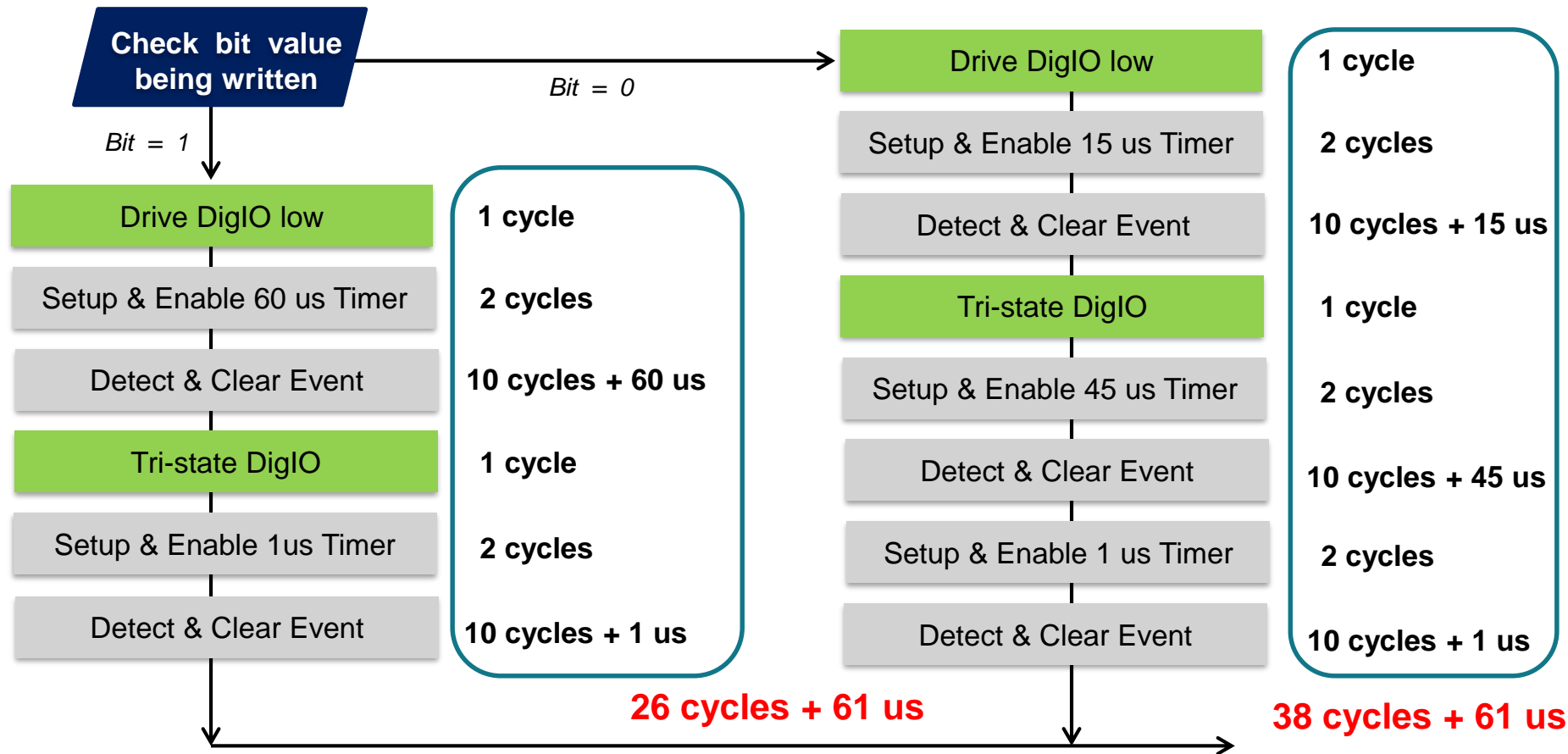
Step 6: Estimating PRU Cycles



Step 6: Estimating PRU Cycles



Step 6: Estimating PRU Cycles



PRU Read Latencies: Internal vs External MMRs

MMRs	Local MMR Access (PRU cycles @ 200MHz)	Global MMR Access (PRU cycles @ 200MHz)
PRU R31 (GPI)	1	N/A
PRU CTRL	4	36
PRU CFG	3	35
PRU INTC	3	35
PRU DRAM	3	35
PRU Shared DRAM	3	35
PRU ECAP	4	36
PRU UART	14	46
PRU IEP	12	44

Table 1: PRU-ICSS MMRs

Note: Latency values listed are “best-case” values.

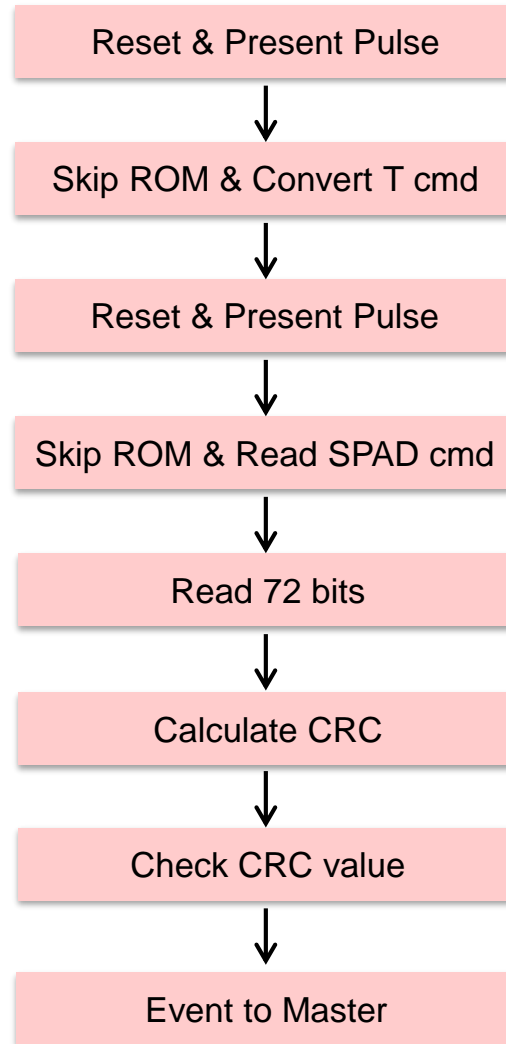
MMRs	Read Latency (PRU cycles @ 200MHz)
L30CMC	27
SRAM_INT	36
DDR	45
EDMA_TPCC	41
PWMSS	38
UART	34
DMTIMER	34
WDT	38
RTC	62
I2C	34
McASP	42
McSPI	34
DCAN	40
GPIO	34
ADC	42
LCDC	29
Ethernet	41
USB	36
MMC	36
EMIF	36
GPMC	38
PRCM	88
Control	34

Table 2: SoC MMRs

Step 7...



Step 7: Assessing Application Feasibility



PRU Cycle Estimate

~ 8.232 ms + change

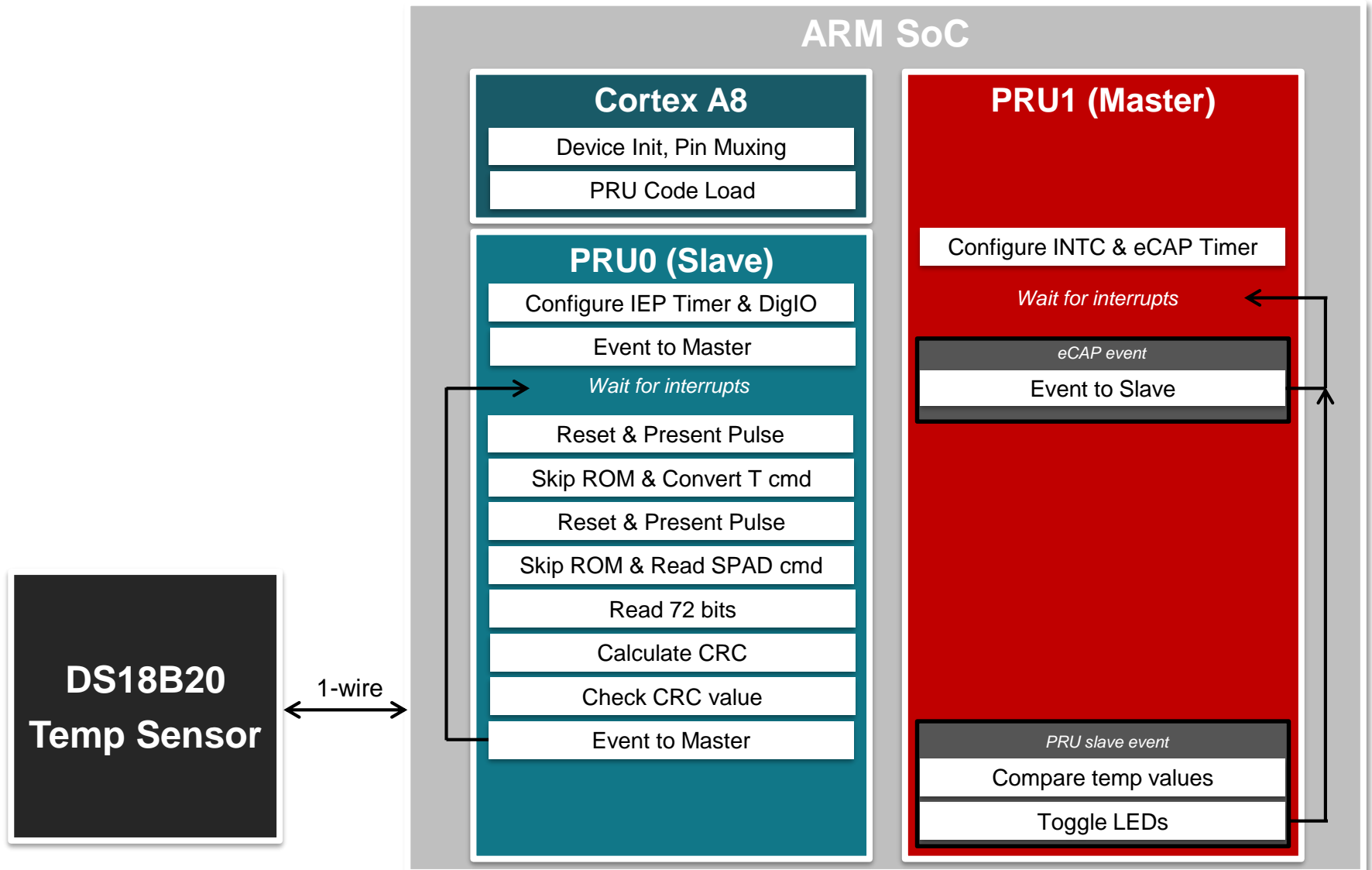
Application Requirement

3 s

Step 8...



Step 8: Start Coding!



**For Temp Monitor
source code and lab,
visit:**



[http://processors.wiki.ti.com/index.php/PRU Training: Hands-on Labs](http://processors.wiki.ti.com/index.php/PRU_Training:_Hands-on_Labs)

Thank you!



For more information about the PRU, visit:

Presentation Home – www.ti.com/sitarabootcamp

PRU-ICSS Wiki – <http://processors.wiki.ti.com/index.php/PRU-ICSS>

PRU Evaluation Hardware – <http://www.ti.com/tool/PRUCAPE>

Support – <http://e2e.ti.com>