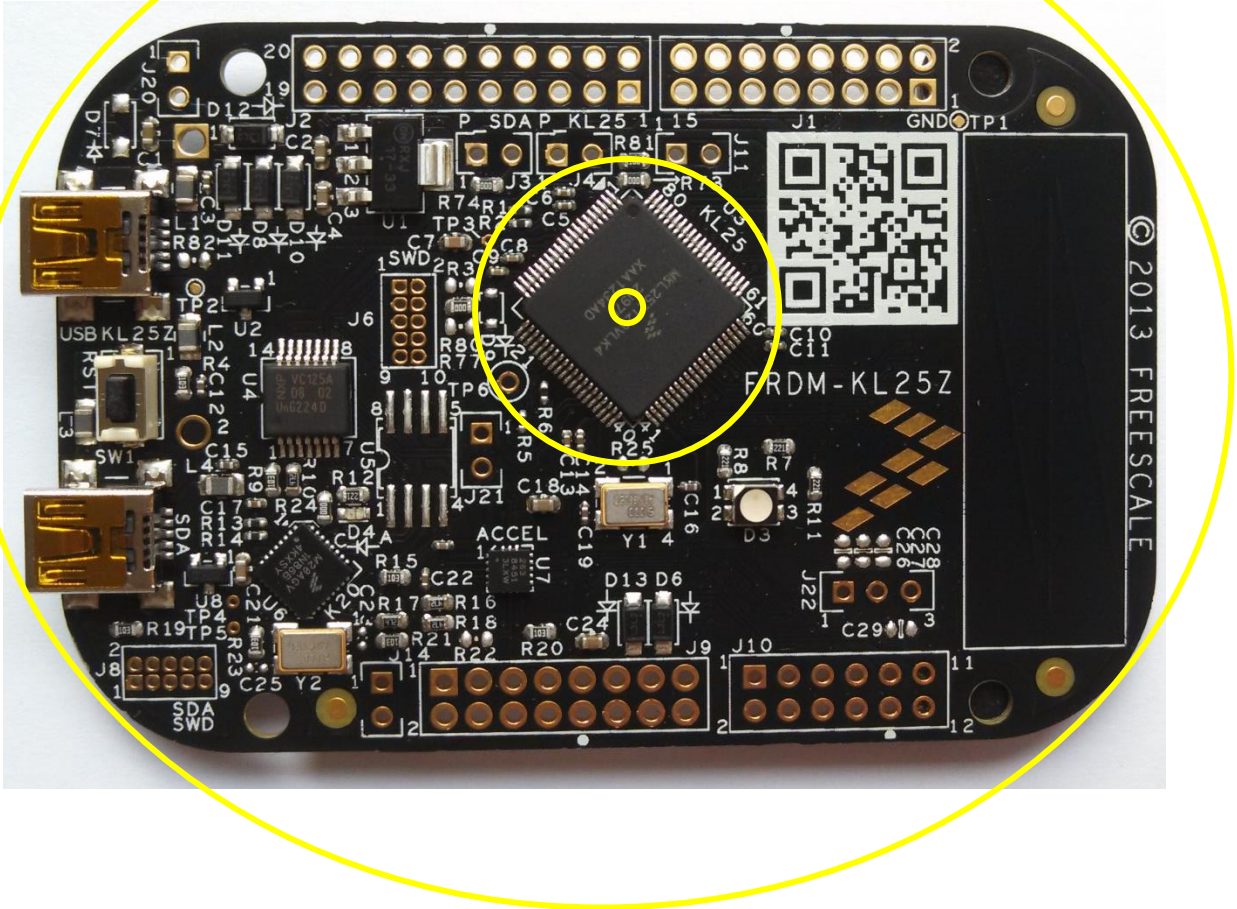
CPU, MCU and Board Features for Low Power and Energy

Overview

- Cortex-M0+ CPU Core
- KL25Z MCU and Peripherals
- FRDM-KL25Z Development Board



Information Source – Reference Manual

- Chapter 7: Power Management
  - Describes different modes available
- Chapter 13: System Mode Controller
  - Switches system between different power modes
- Chapter 5: Clock Distribution
  - Clock sources
  - Controls 1.0 V band-gap reference (for ADC, CMP)
- Chapter 12: System Integration Module
  - Selection of clock sources for configurable peripherals
- Chapter 15: Low-Leakage Wakeup Unit
- Chapter 24: Multipurpose Clock Generator
- Chapter 33: Low-Power Timer
- Chapter 14: Power Management Controller
  - Detects low operating voltage
    - Low Voltage Warning interrupts
    - Power-On Reset

Application Notes

- AN4503
- KLQRUG

Freescale Semiconductor
Users Guide

KLQRUG
Rev. 0, 09/2012

Kinetis L Peripheral Module Quick Reference

A Compilation of Demonstration Software for Kinetis L Series Modules

This collection of code examples, useful tips, and quick reference material has been created to help you speed the development of your applications. Most chapters in this document contain examples that can be modified to work with Kinetis MCU Family members. When you're developing your application, consult your device data sheet and reference manual for part-specific information, such as which features are supported on your device.

Sample code can be found at KL25_SC.exe, available from:
www.freescale.com/files/32bit/software/KL25_SC.exe

Information about the ARM core can be found in the help center at ARM.com

The most up-to-date revisions of our documents are on the Web. Your printed copy may be an earlier revision. To verify that you have the latest information available, refer to freescale.com

Freescale Semiconductor
Application Note

Document Number:AN4503
Rev. 1, 11/2012

Power Management for Kinetis and ColdFire+ MCUs

When and how to use low-power modes

by: Philip Drake

1 Introduction

Applications strive for high performance within constrained energy budgets, which continue to play a significant role in determining embedded designs. Increasing requirements do not allow for compromises on performance and continue to push for low energy budgets.

The Kinetis and ColdFire+ microcontroller families include internal power management features that can be used to control the microcontroller's power usage. This application note discusses how to use the power management systems, provides use case examples, and shows real-time current measurement results for these use cases.

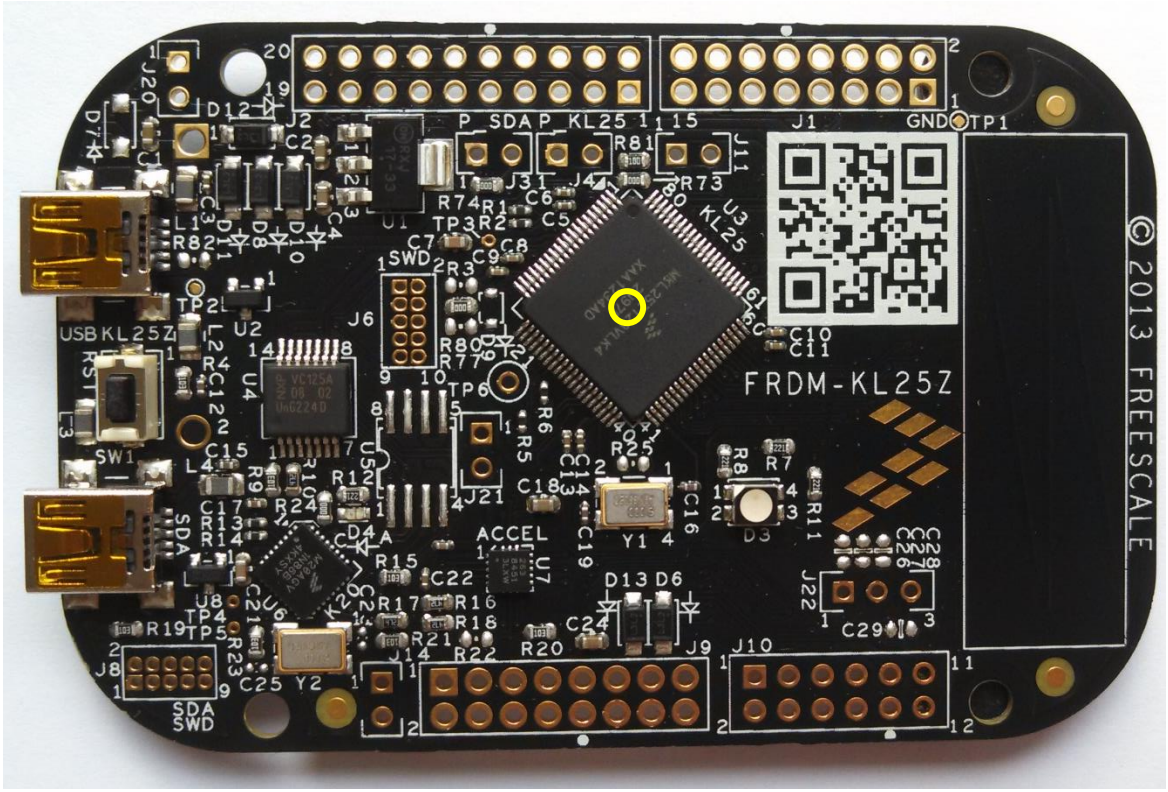
Also included is a discussion of the differences between power management features on the various microcontrollers, along with drivers demonstrating the low-power features. Tips are given for using each of the power modes.

Power management methods discussed here do not include

Contents

Overview

- Cortex-M0+ CPU Core
- KL25Z MCU and Peripherals
- FRDM-KL25Z Development Board

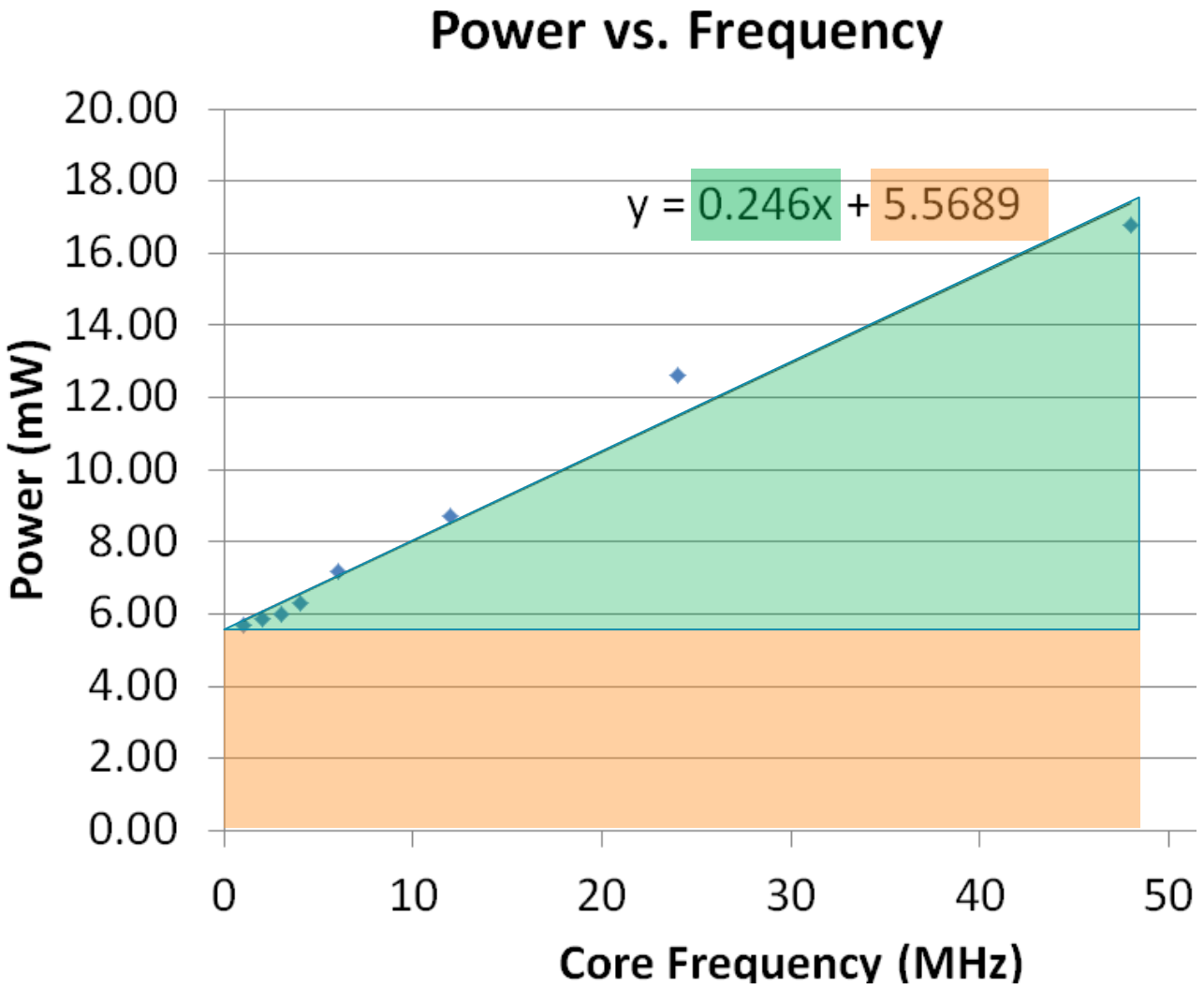


Setting the MCU Clock Frequency

KL25Z MCU Power at 3.0 V

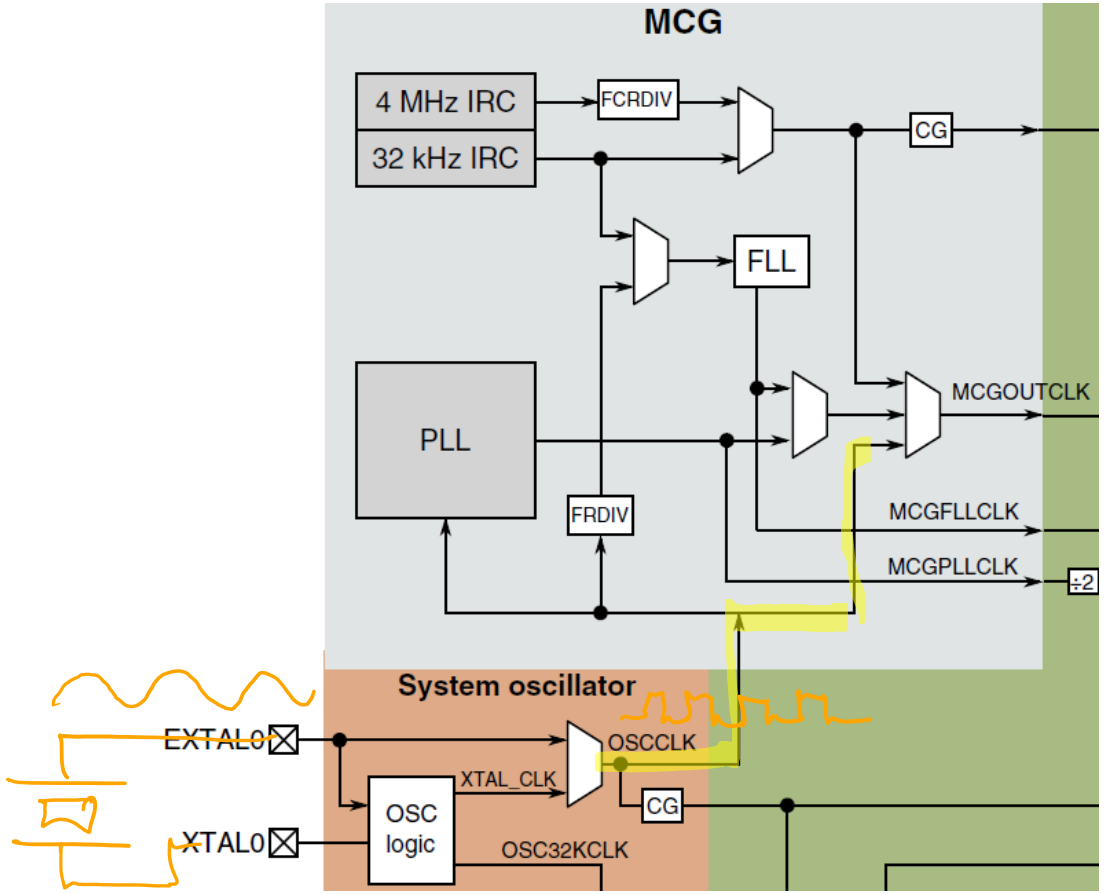
- Characteristics
  - Static power is 5.5689 mW
  - Dynamic power is roughly proportional to frequency:
    0.246 mW/MHz

- Can reduce power consumption by reducing CPU core frequency



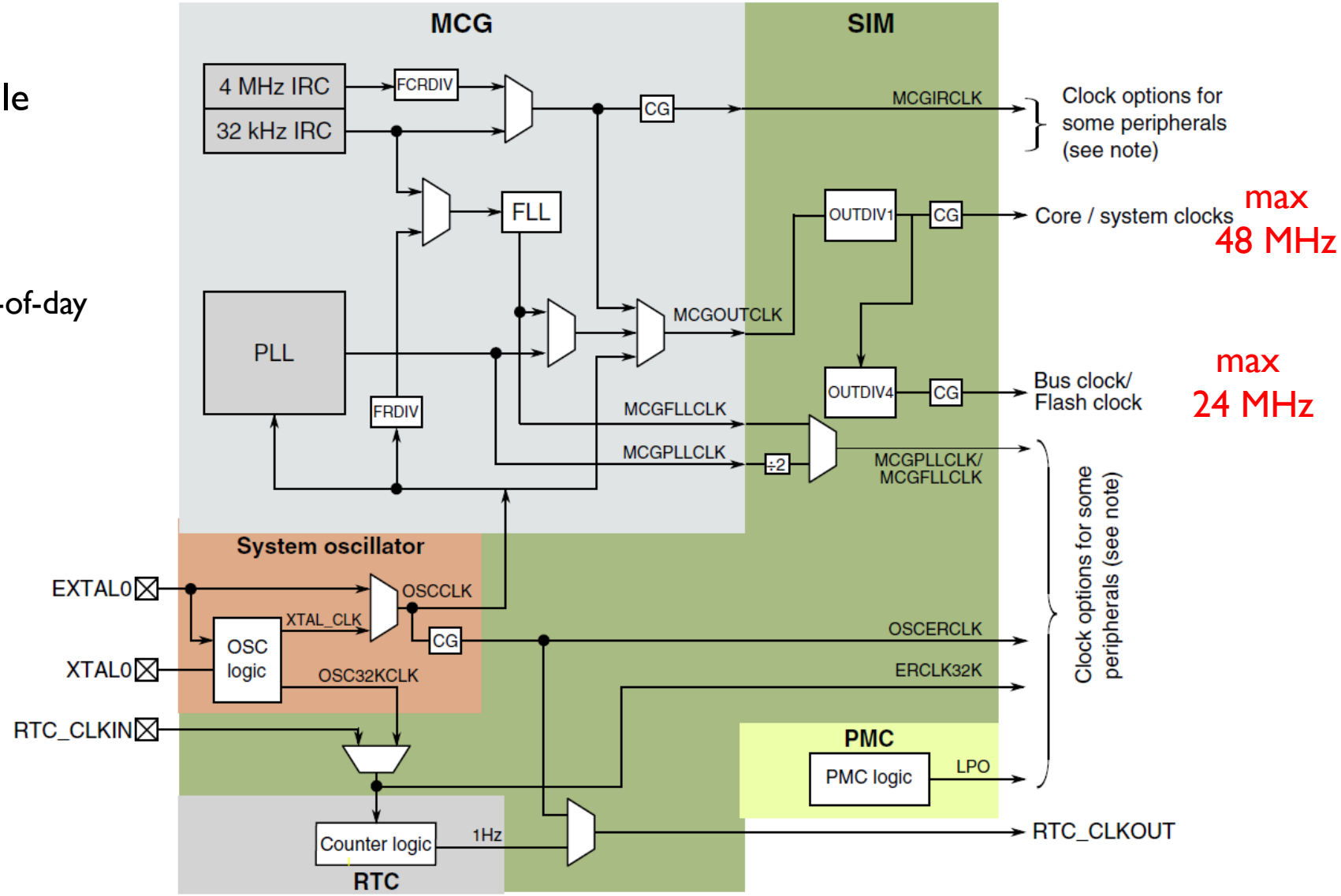
What Clock Sources are Available?

- Multipurpose Clock Generator (MCG)
  - Internal Reference Clocks
    - Fast IRC: 4 MHz
    - Slow IRC: 32 kHz
  - Frequency-Locked Loop
    - Used as Digitally-Controlled Oscillator (DCO)
    - Clocked by external or 32 kHz internal reference clock
    - Can multiply input frequency by an integer
  - Phase-Locked Loop
    - Has voltage-controlled oscillator (VCO)
    - Clocked by external reference clock
- System Oscillator
  - Uses external crystal or resonator to generate external reference clock



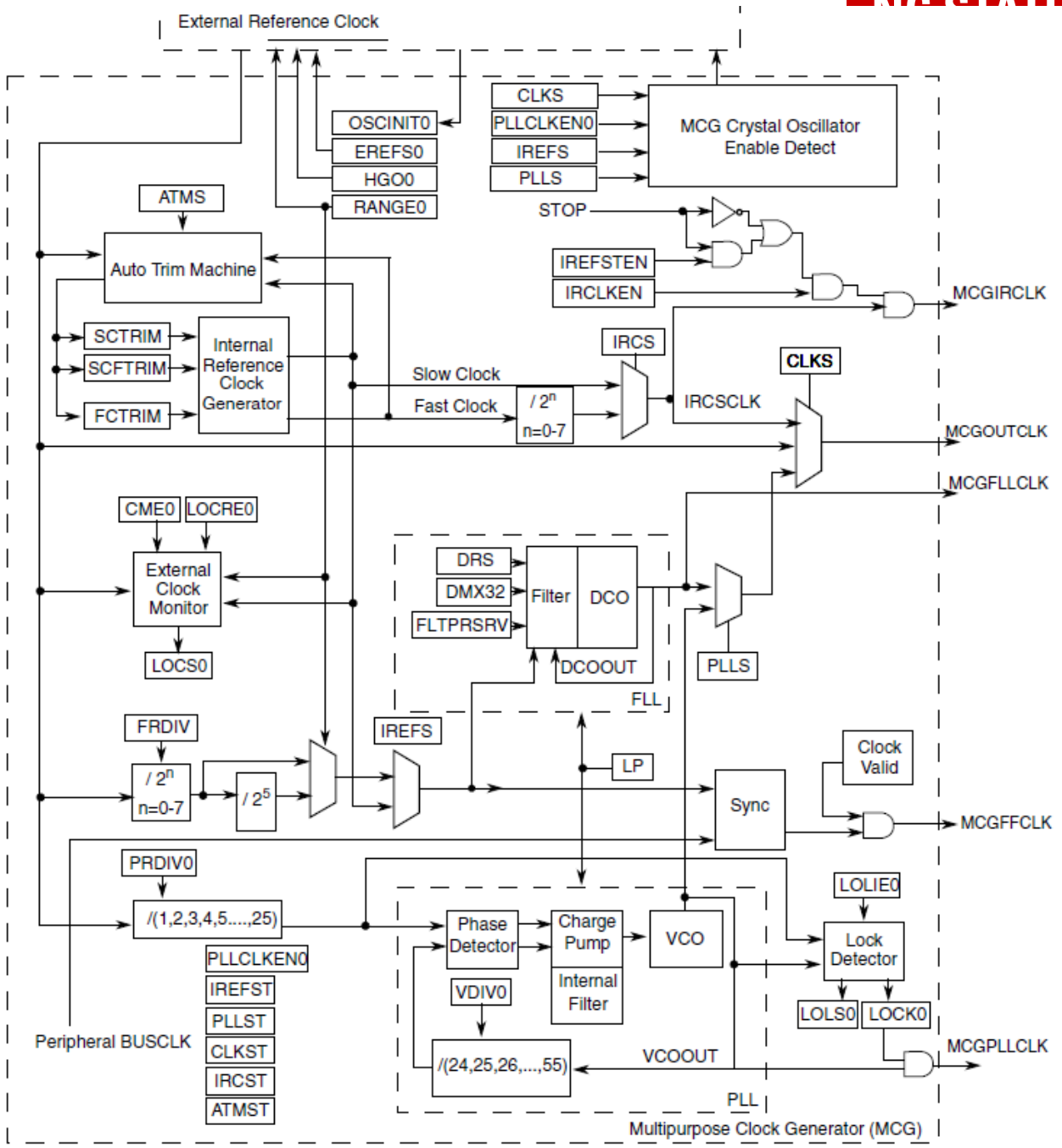
More on KL25 Clock System

- SIM – System Integration Module
  - Divides and routes clock signals
- RTC – Real-Time Clock
  - Hardware counters to track time-of-day and date.
- PMC – Power Management Controller
  - Provides LPO clock for some low-power peripherals



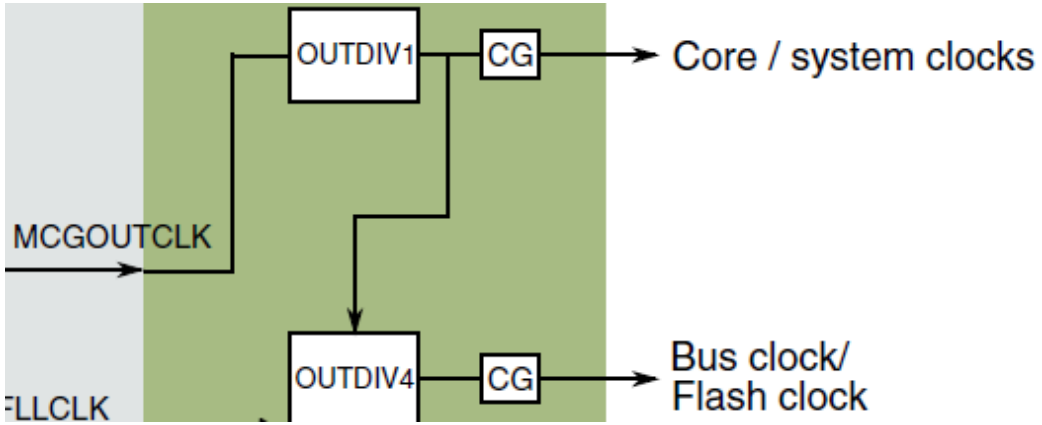
MCG Control Registers

- Extensive configuration options available!
- See Chapter 24 of reference manual for details



How Do We Divide the Clock Speed?

- Dividers are located in SIM (CLKDIV1)
- Core and System clock
  - Up to 48 MHz
  - Input is MCGOUTCLK
  - Divided by OUTDIV1+1 (four bits, 1 to 16)
- Bus and Flash clock
  - Up to 24 MHz
  - Input is core and system clock
  - Divided by OUTDIV4+1 (three bits, 1 to 8)

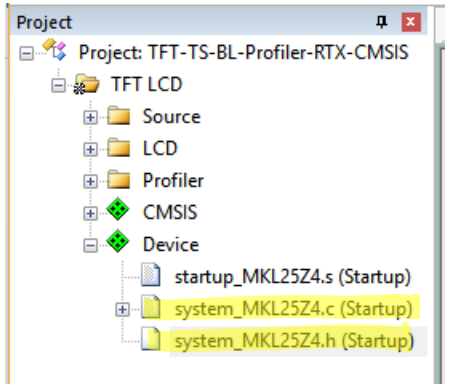


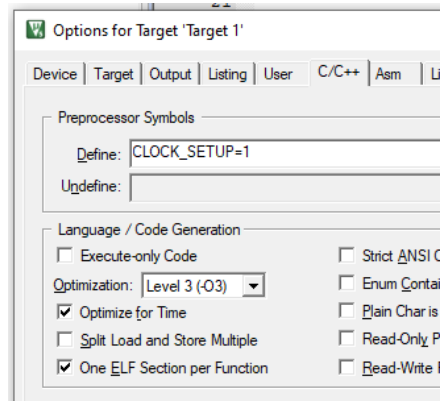
- SIM register CLKDIV1 controls both dividers

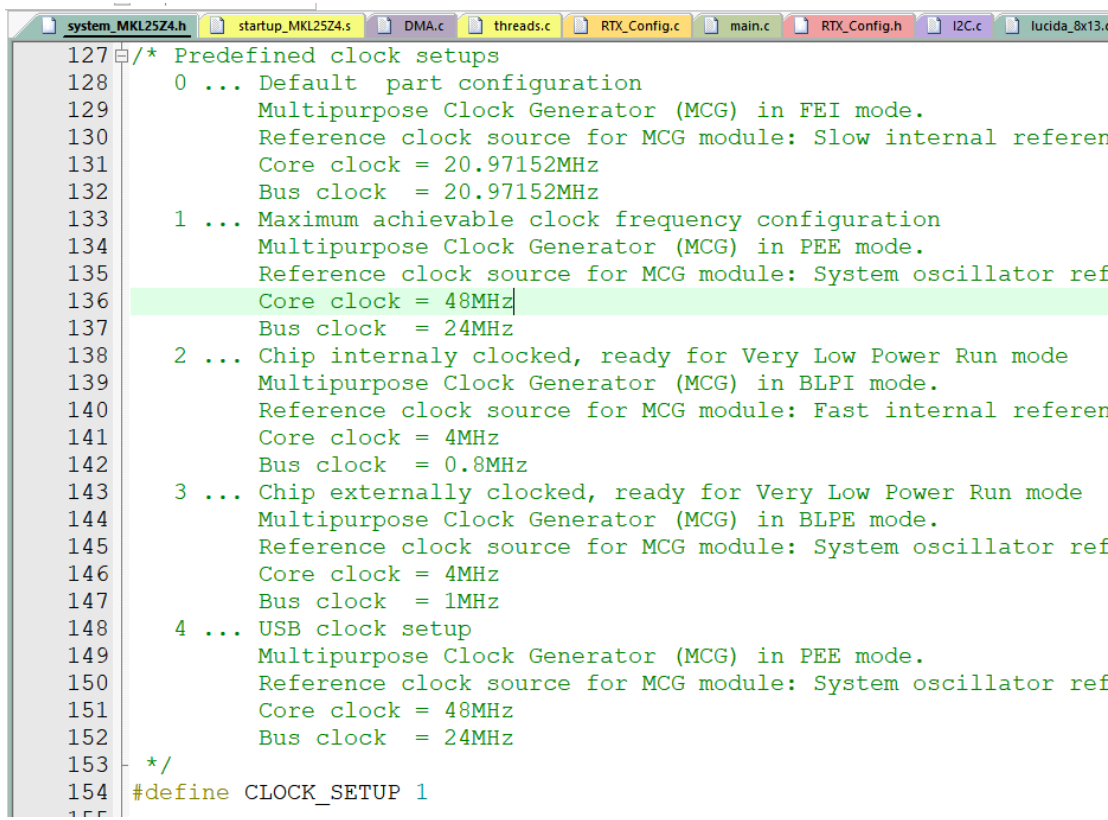
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUTDIV1				0									OUTDIV4			0															
W																																
Reset	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Configuring The System Clock

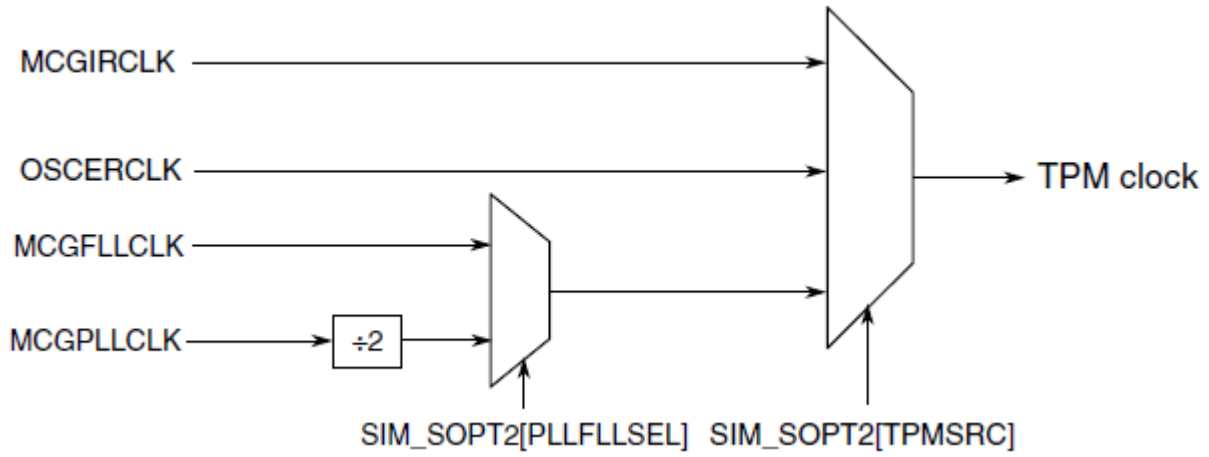
- Specify clock with CLOCK_SETUP symbol in program
- Two ways to define CLOCK_SETUP
  - Compiler command line options: -dCLOCK_SETUP=1
  - system_MKL25Z4.h: #define CLOCK_SETUP 1
- CLOCK_SETUP used by configuration code
  - Files system_MKL25Z4.[c|h] in Project's **Device** node
  - system_MKL25Z4.h
    - Five predefined clock setups
    - Settings for watchdog, allowed low power modes
  - system_MKL25Z4.c
    - "Provides a system configuration function and a global variable that contains the system frequency. It configures the device and initializes the oscillator"
    - void **SystemInit**(void)
      - Configures watchdog, low power modes allowed
      - Configures system clock (with safe transitions)
    - void **SystemCoreClockUpdate**(void)
      - Updates SystemCoreClock variable based on current hardware settings







Side Effects

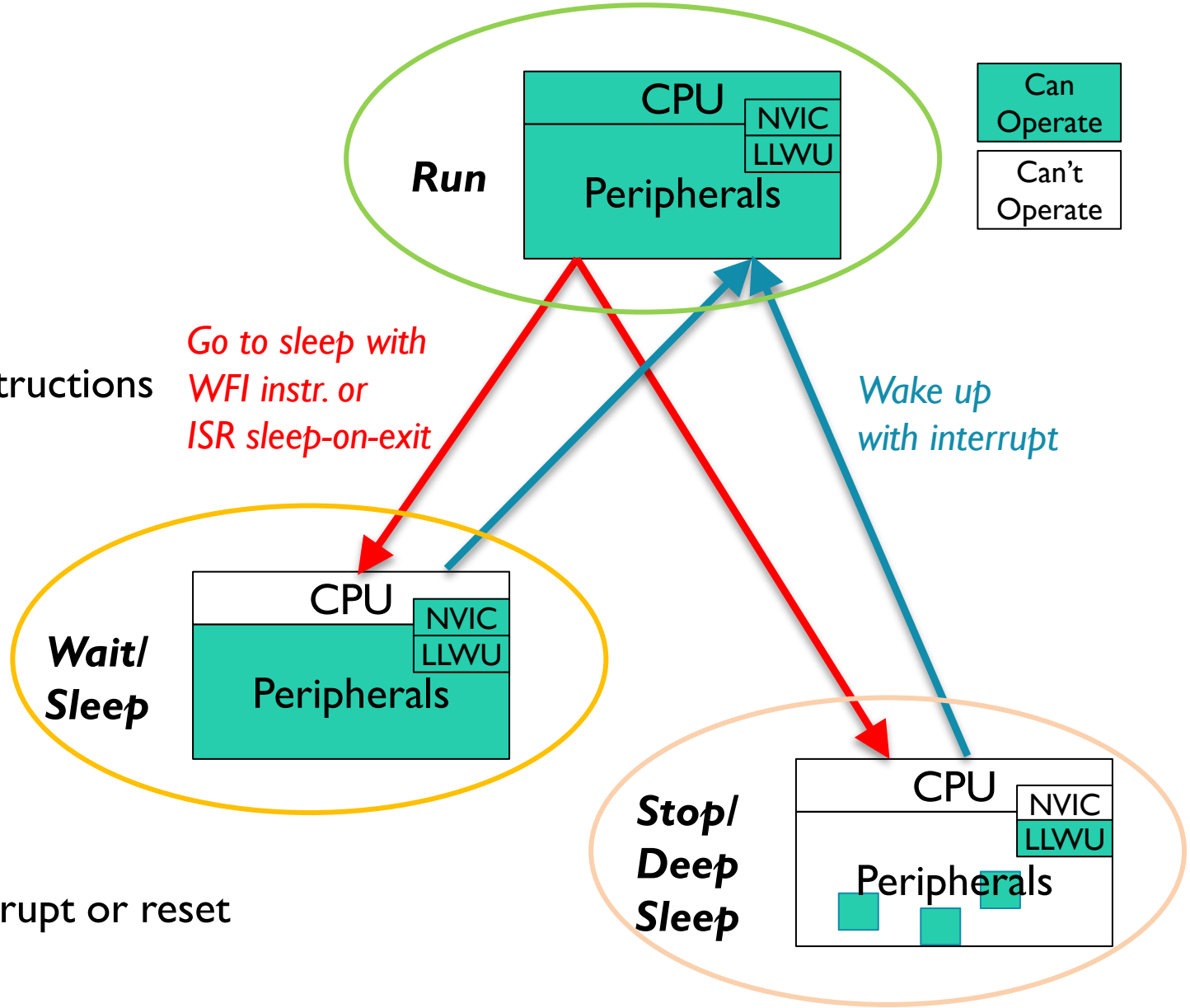


- PIT and other peripherals are clocked by bus clock
  - Changing bus frequency will change PIT period
  - May not be what you want
- TPM uses other clock inputs which don't change with bus clock
  - Can select FLL clock, PLL clock, internal reference clock, external reference clock, or oscillator clock
  - Use SIM SOPT2 register to select clock source

CPU and MCU Power Modes

Power Mode Overview

- See Table 7-1 in MCU SRM
- Run Modes – CPU executes instructions
  - Regular – CPU runs at up to 48 MHz
  - Very Low Power – runs at up to 4 MHz
- Stop/Wait Modes – CPU doesn't execute instructions
  - Wait (ARM "sleep")
    - All peripherals and NVIC **can** operate
  - Stop (ARM "deep sleep")
    - Most peripherals **cannot** operate
    - NVIC is not powered, must use LLWU
  - Specific type selected with control registers
    - SCR: SLEEPDEEP
    - SMC: RUNM, STOPM, PSTOPO, VLLSM
- Return to Run mode when triggered by interrupt or reset (depends on mode)



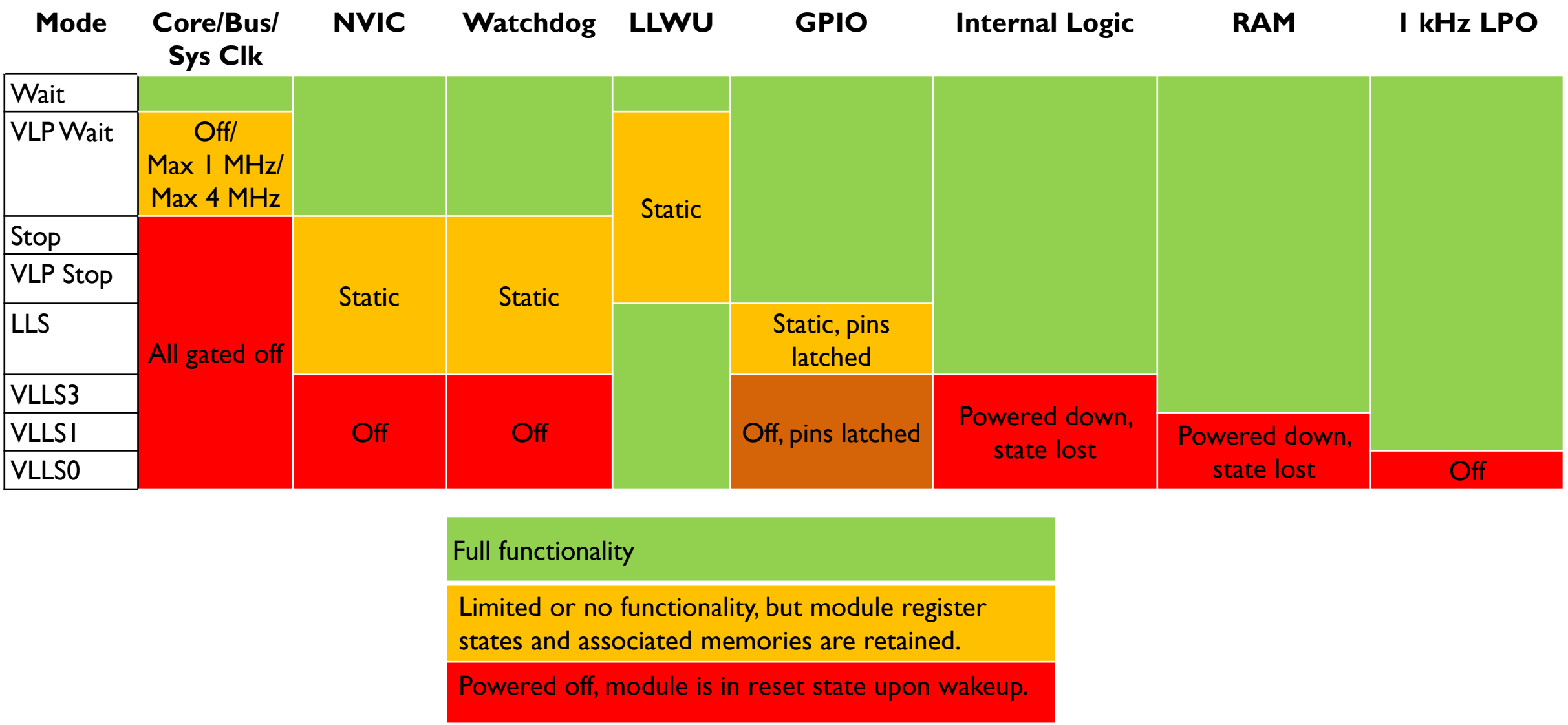
Example Power Mode Documentation Overview

- Full documentation in MCU Subfamily Reference Manual, Chapter 7

Table 7-1. Chip power modes

Chip mode	Description	Core mode	Normal recovery method
Normal run	Allows maximum performance of chip. Default mode out of reset; on-chip voltage regulator is on.	Run	—
Normal Wait - via WFI	Allows peripherals to function while the core is in sleep mode, reducing power. NVIC remains sensitive to interrupts; peripherals continue to be clocked.	Sleep	Interrupt
Normal Stop - via WFI	Places chip in static state. Lowest power mode that retains all registers while maintaining LVD protection. NVIC is disabled; AWIC is used to wake up from interrupt; peripheral clocks are stopped.	Sleep Deep	Interrupt
VLPR (Very Low Power Run)	On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. Reduced frequency Flash access mode (1 MHz); LVD off; in BLPI clock mode, the fast internal reference oscillator is available to provide a low power nominal 4MHz source for the core with the nominal bus and flash clock required to be <800kHz; alternatively, BLPE clock mode can be used with an external clock or the crystal oscillator providing the clock source.	Run	—
VLPW (Very Low Power Wait) -via WFI	Same as VLPR but with the core in sleep mode to further reduce power; NVIC remains sensitive to interrupts (FCLK = ON). On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency.	Sleep	Interrupt
VLPS (Very Low Power Stop)-via WFI	Places chip in static state with LVD operation off. Lowest power mode with ADC and pin interrupts functional. Peripheral clocks are stopped, but OSC, LPTMR, RTC, CMP, TSI can be used. TPM and UART can optionally be enabled if their clock source is enabled. NVIC is disabled	Sleep Deep	Interrupt

Mode Comparison



Details of Stop Modes

- STOP: The basic stop mode
  - Core and system clocks are gated off
- VLPS: Very-low-power stop
  - Core and system clocks are gated off
- LLS: Low-leakage stop
  - Core and system clocks are gated off.
  - Internal logic supply voltage is reduced but state is retained.
- VLLS3: Very-low-leakage stop 3
  - RAM and I/O states are retained.
  - Internal logic is powered down and lost, so CPU resumes with Reset handler.
- VLLS2: Not available on Cortex-M0+
- VLLS1: Very-low-leakage stop 1
  - I/O states are retained.
  - RAM and internal logic are powered down and lost, so CPU resumes with Reset handler.
- VLLS0: Very-low-leakage stop 0
  - I/O states are retained.
  - RAM and internal logic are powered down and lost, so CPU resumes with Reset handler.
  - 1 kHz LPO is disabled
  - Power on reset circuit can be enabled or disabled

Typical Currents in Different Modes (V_{DD} = 3 V)

Mode			Current		
MCU	CPU (ARM)	Normal	Very Low Power	Low Leakage	Very Low Leakage
Run		5000 µA			
Wait	Sleep	3700 µA			
Stop	Deep Sleep	345 µA			
Stop 3					
Stop 1					
Stop 0					

Typical Currents in Different Modes (V_{DD} = 3 V)

Mode			Current		
MCU	CPU (ARM)	Normal	Very Low Power	Low Leakage	Very Low Leakage
Run		5000 µA	250 µA		
Wait	Sleep	3700 µA	135 µA		
Stop	Deep Sleep	345 µA	4.4 µA		
Stop 3					
Stop 1					
Stop 0					

Typical Currents in Different Modes (V_{DD} = 3 V)

Mode			Current		
MCU	CPU (ARM)	Normal	Very Low Power	Low Leakage	Very Low Leakage
Run		5000 µA	250 µA		
Wait	Sleep	3700 µA	135 µA		
Stop	Deep Sleep	345 µA	4.4 µA	1.9 µA	
Stop 3					
Stop 1					
Stop 0					

Typical Currents in Different Modes (V_{DD} = 3 V)

Mode			Current		
MCU	CPU (ARM)	Normal	Very Low Power	Low Leakage	Very Low Leakage
Run		5000 µA	250 µA		
Wait	Sleep	3700 µA	135 µA		
Stop	Deep Sleep	345 µA	4.4 µA	1.9 µA	
Stop 3					1.4 µA
Stop 1					0.77 µA
Stop 0					0.38 µA

Recovery Methods: What Code Does CPU Run after Waking Up?

Chip Mode	Core Mode	Comments	Recovery Method	On Wake-Up, CPU runs
Wait	Sleep		Peripheral interrupt	Peripheral interrupt handler
Stop	Sleep Deep		Peripheral interrupt	Peripheral interrupt handler
LLS	Sleep Deep		Wakeup interrupt	LLWU interrupt handler
VLLS3	Sleep Deep		Wakeup reset	Reset handler, but LLWU flag is set
VLLS1	Sleep Deep	SRAM powered off and lost	Wakeup reset	Reset handler, but LLWU flag is set
VLLS0	Sleep Deep	SRAM powered off and lost	Wakeup reset	Reset handler, but LLWU flag is set

- Peripheral Interrupt: CPU runs interrupt handler
- Wakeup Interrupt: CPU runs LLWU interrupt handler
- Wakeup Reset: CPU runs reset handler (LLWU flag indicates cause).
  - See **armcc User Guide** section 4.82 (*How to prevent uninitialized data from being initialized to zero*)

4.82 How to prevent uninitialized data from being initialized to zero

The ANSI C specification states that static data that is not explicitly initialized, is to be initialized to zero.

Therefore, by default, the compiler puts both zero-initialized and uninitialized data into the same ZI data section, which is populated with zeroes at runtime by the C library initialization code.

You can prevent uninitialized data from being initialized to zero by placing that data in a different section. This can be achieved using `#pragma arm section`, or with the GNU compiler extension `__attribute__((section("*name*")))`.

The following example shows how to retain uninitialized data using `#pragma arm section`:

```
#pragma arm section zidata = "non_initialized"
int i, j; // uninitialized data in non_initialized section (without the pragma,
             would be in .bss section by default)
#pragma arm section zidata // back to default (.bss section)
int k = 0, l = 0; // zero-initialized data in .bss section
```

The `non_initialized` section is placed into its own `UNINIT` execution region, as follows:

```
LOAD_1 0x0
{
  EXEC_1 +0
  {
    * (+RO)
    * (+RW)
    * (+ZI) ; ZI data gets initialized to zero
  }
  EXEC_2 +0 UNINIT
  {
    * (non_init) ; ZI data does not get initialized to zero
  }
}
```

Related references

9.76 #pragma arm section [section_type_list] on page 9-589.

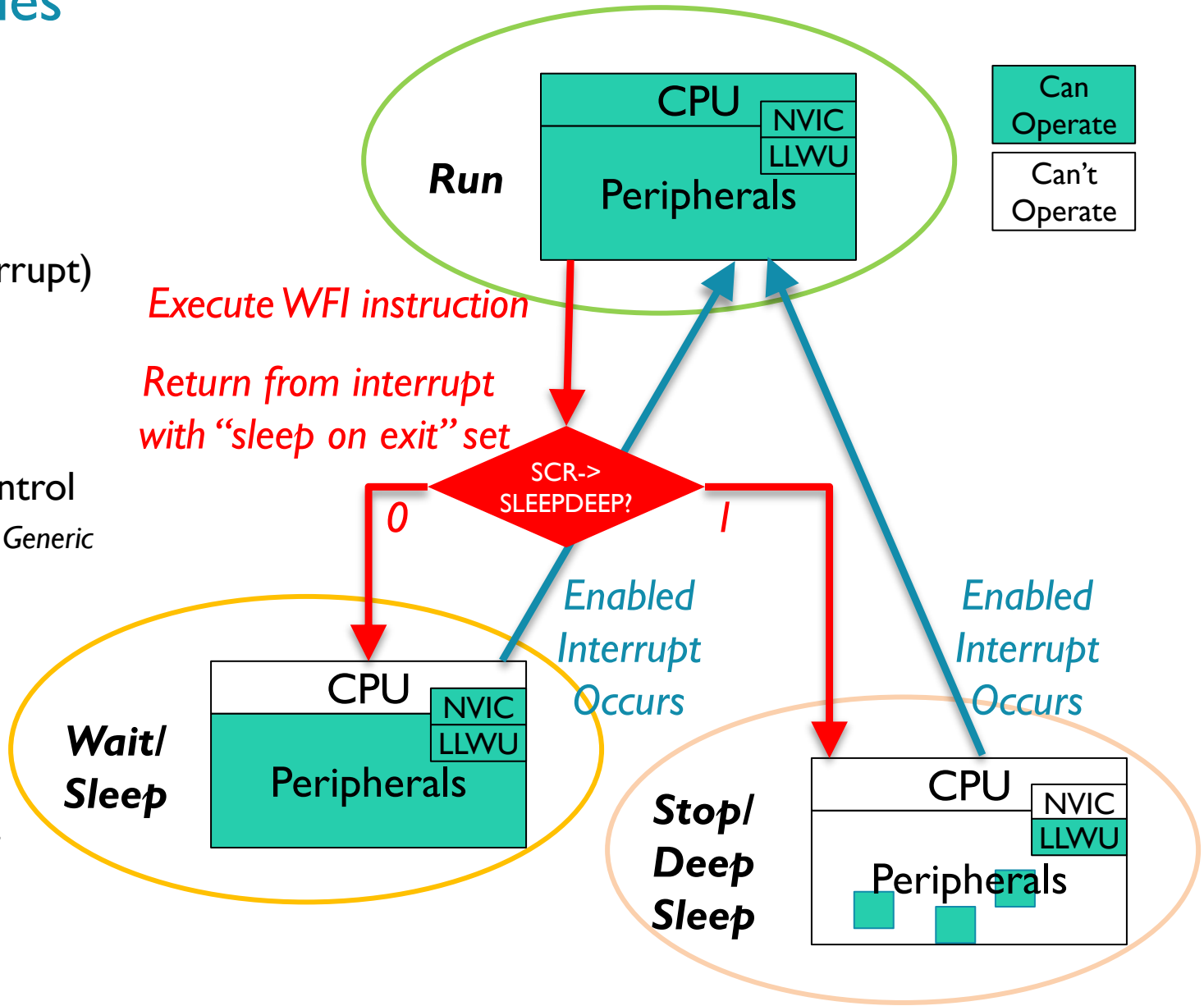
9.66 __attribute__((section("name"))) variable attribute on page 9-579.

Related information

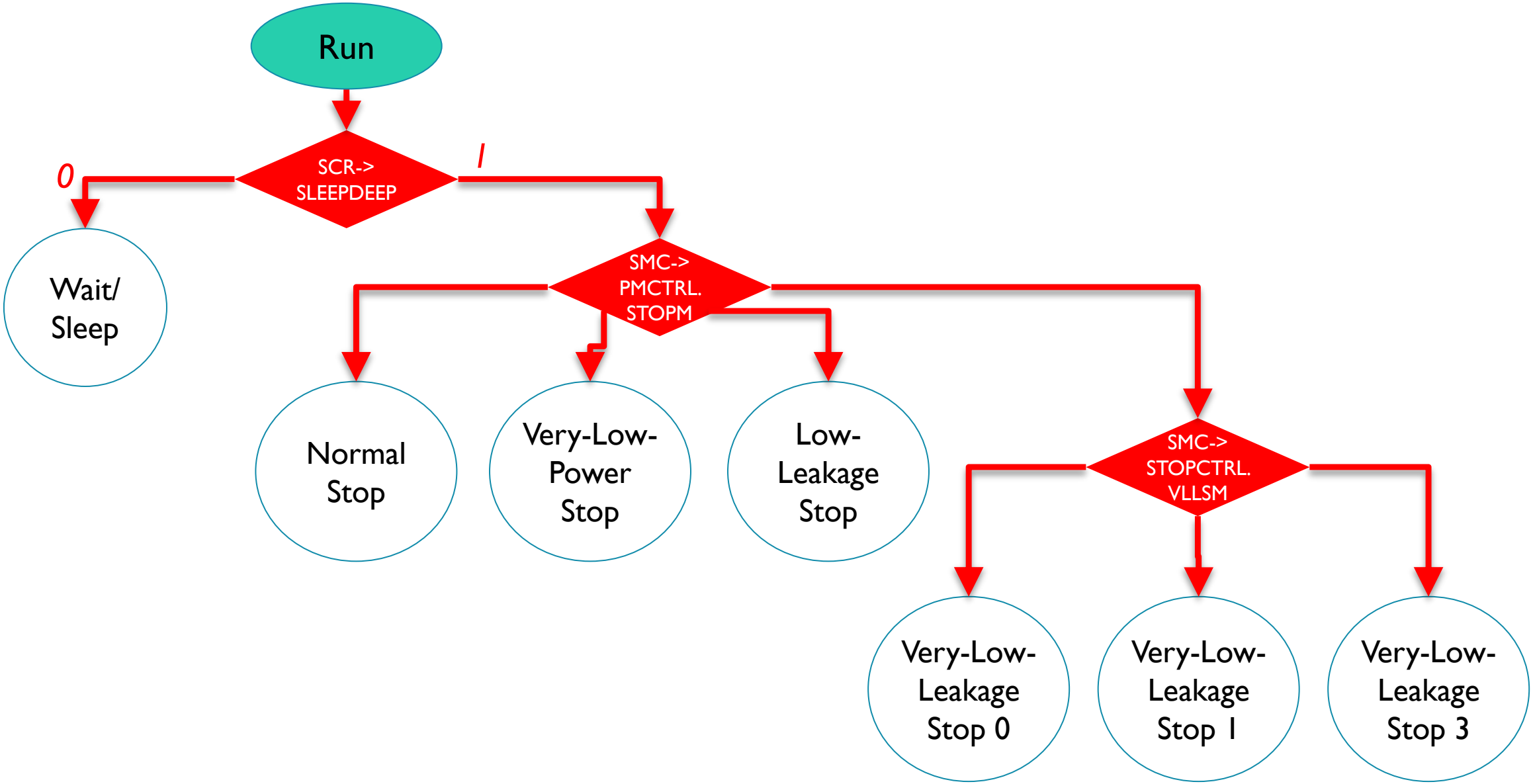
Execution region attributes.

Transitions Between Power Modes

- Enter low-power mode
  - How?
    - By executing WFI instruction (wait for interrupt)
    - By returning from exception handler (if SLEEPONEXIT in SCR is set)
  - Go to Wait/Sleep or Stop/Deep Sleep?
    - Selected by SLEEPDEEP field of System Control Register *Cortex-M0+ Peripheral, in Cortex-M0+ Device Generic User Guide, Section 4.3.6*
- Exit low-power mode
  - Exit when enabled interrupt occurs
- Sleep-on-Exit
  - Useful if MCU only executes code in ISRs
  - Set SLEEPONEXIT bit in SCR

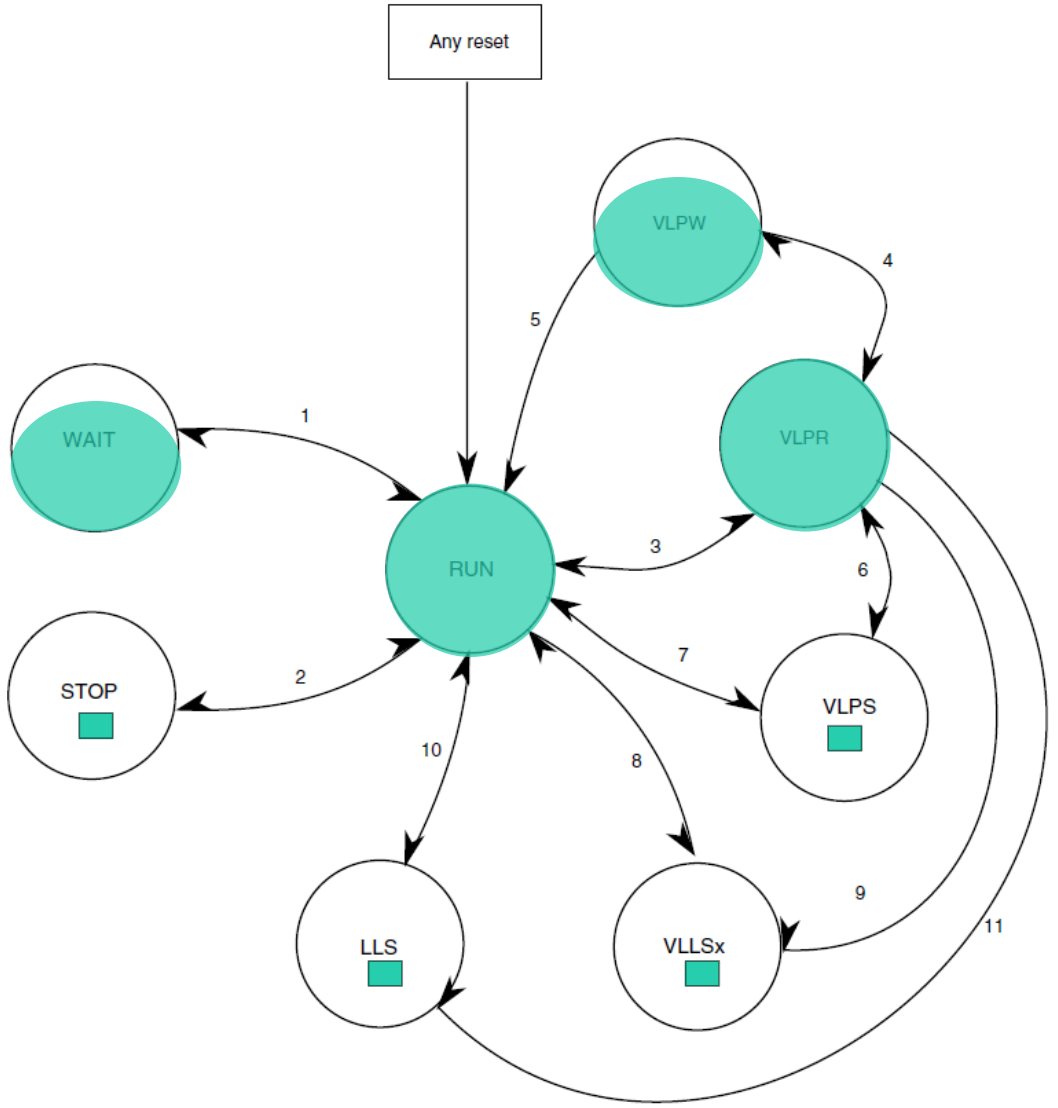


The Big Picture on Selecting Power-Saving Modes

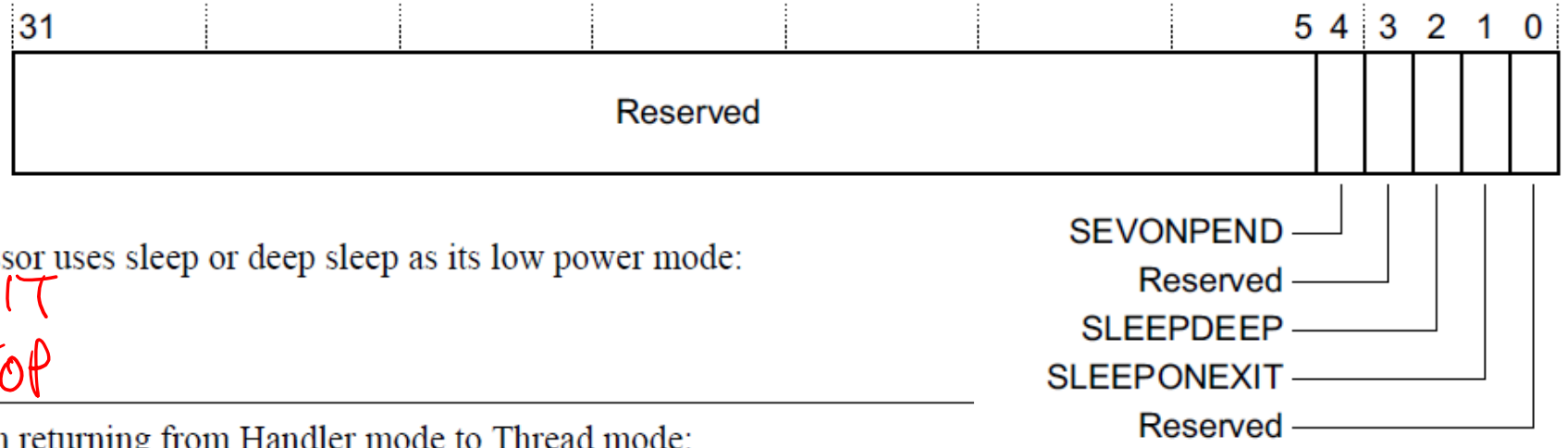


Valid Transitions Between Power Modes

- Reset puts MCU in RUN mode (CPU at up to 48 MHz)
- Going to Sleep
  - From RUN mode, can get into most other modes
    - Wait
    - Stop
    - LL Stop
    - VLL Stop
  - From VLPR (Very Low Power Run) mode (CPU at up to 4 MHz), can get into only some of other modes
    - VLP Wait
    - VLP Stop
    - LL Stop
- Waking Up
  - From LLS, VLPS, Wait, VLPW, CPU runs **interrupt** handler
  - From VLLSx, CPU runs **reset** handler



How Do We Select The Low Power Mode?



SLEEPDEEP	Controls whether the processor uses sleep or deep sleep as its low power mode: 0 = sleep. — WAIT 1 = deep sleep. — STOP
SLEEPONEXIT	Indicates sleep-on-exit when returning from Handler mode to Thread mode: 0 = do not sleep when returning to Thread mode. 1 = enter sleep, or deep sleep, on return from an ISR to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.

- First configure whether WFI will cause sleep (wait) or deep sleep (stop)
- Cortex-M0+ has System Configuration Block (standard peripheral) with System Control Register
  - SCB->SCR: SLEEPDEEP bit
  - See *Cortex-M0+ Device Generic User Guide, Section 4.3.6*

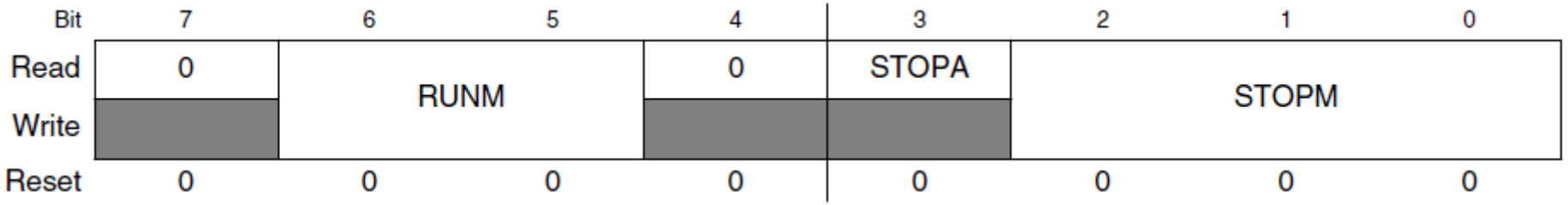
Allow Additional Modes

Address: 4007_E000h base + 0h offset = 4007_E000h

Bit	7	6	5	4	3	2	1	0
Read	0		AVLP	0	ALLS	0	AVLLS	0
Write								
Reset	0	0	0	0	0	0	0	0

- First need to allow the modes in Power Mode Protection register (SMC_PMPROT)
  - AVLP: allow very-low-power modes
  - ALLS: allow low-leakage stop mode
  - AVLLS: allow very-low-leakage stop mode
- Can only write PMPROT once after reset

Power Mode Control Register SMC_PMCTRL



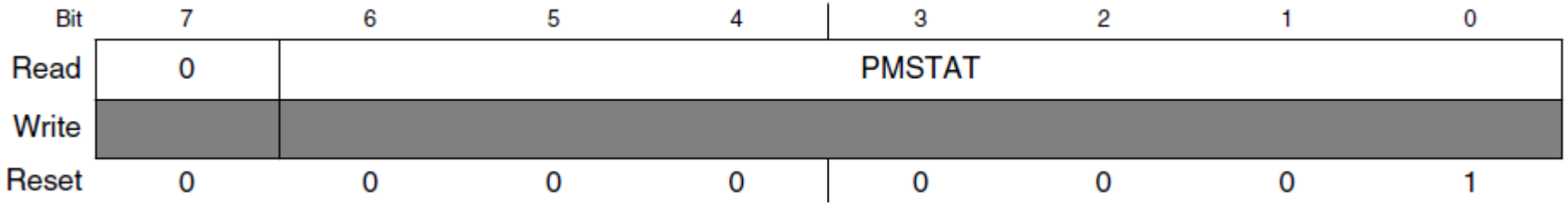
- RUNM: Run Mode control
  - 00 Normal Run
  - 10 Very Low-power run
  - Other values reserved
- STOPA: Stop aborted
  - Previous stop mode entry sequence did not complete
- STOPM: Stop Mode control
  - 000 Normal Stop
  - 010 Very-low-power stop
  - 011 Low-leakage stop
  - 100 Very-low-leakage stop
  - Other values reserved

Stop Mode Control Register – SMC_STOPCTRL

Bit	7	6	5	4	3	2	1	0
Read	PSTOPO		PORPO	0	0	VLLSM		
Write								
Reset	0	0	0	0	0	0	1	1

- PSTOPO – Allows stop to cause only a partial stop
  - 00: STOP: Normal stop mode
  - 01: PSTOP1: Partial stop with both system and bus clocks disabled
  - 10: PSTOP2: Partial stop with both system and bus clocks enabled
- PORPO – Controls power-on-reset circuit activity in VLLS0 mode
  - 0: POR enabled
  - 1: POR disabled
- VLLSM – Selects which VLLS Mode to enter if STOPM is VLLS
  - 000:VLLS0
  - 001:VLLS1
  - 010: n/a (VLLS2)
  - 011:VLLS3

Power Mode Status Register – SMC_PMSTAT

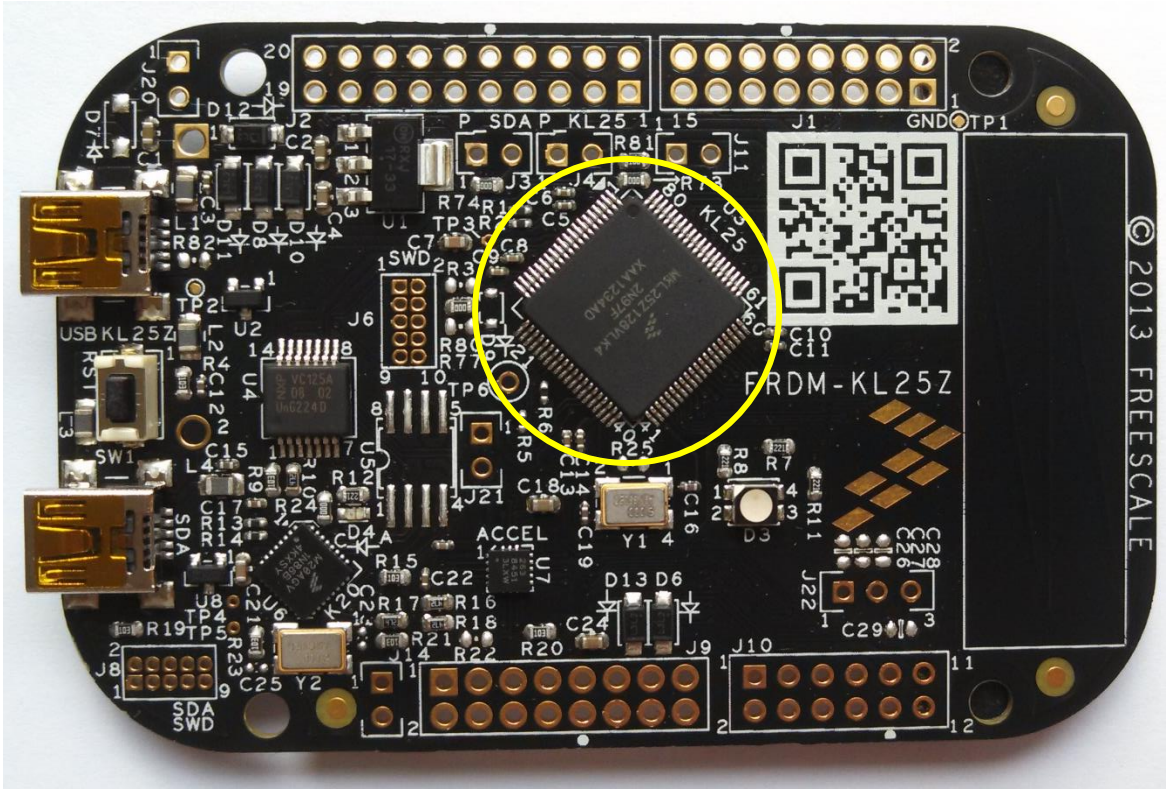


- Indicates the MCU's power mode at present time
- Read-only

000_0001	Current power mode is RUN
000_0010	Current power mode is STOP
000_0100	Current power mode is VLPR
000_1000	Current power mode is VLPW
001_0000	Current power mode is VLPS
010_0000	Current power mode is LLS
100_0000	Current power mode is VLLS

Overview

- Cortex-M0+ CPU Core
- KL25Z MCU and Peripherals
- FRDM-KL25Z Development Board

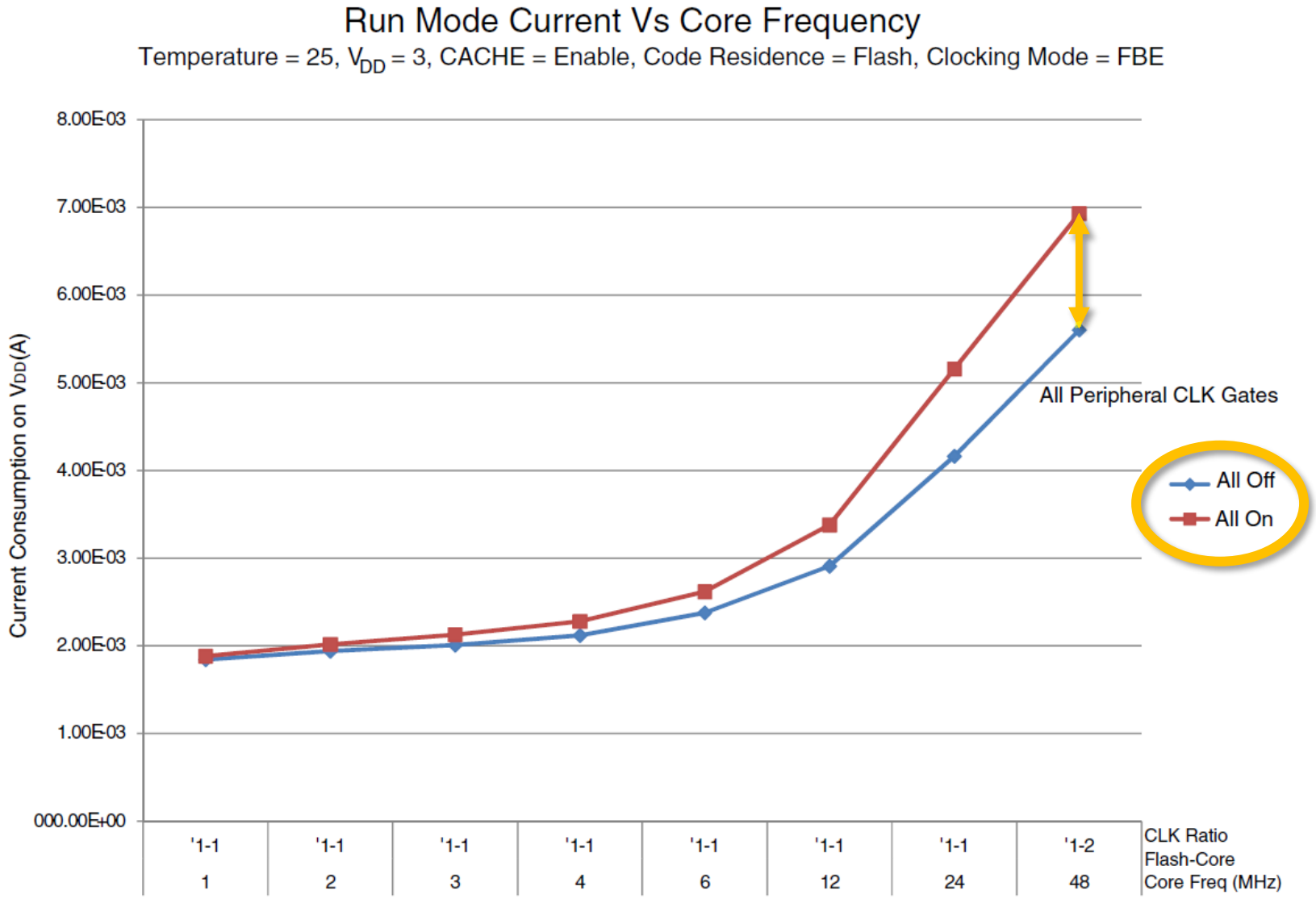


Overview

- Reduce power drawn by unused peripheral modules
  - Clock gating stops dynamic power
- Identify which peripheral modules can operate (and how much) in low power modes
- Use peripheral modules to wake up MCU

Clock Gating Cuts Dynamic Power

- Peripherals use power
  - Microamps, nanoamps – they may matter for your application!
- Save power by disabling clock to unused peripherals
- Use SIM System Clock Gating Control Registers (SIM_SCGC4-7) to disable those clock signals



Datasheet Provides Low-Power Mode Peripheral Adder Currents

Table 6. Low power mode peripheral adders — typical value (continued)

Symbol	Description	Temperature (°C)						Unit
		-40	25	50	70	85	105	
$I_{EREFSTEN32KHz}$	External 32 kHz crystal clock adder by means of the OSC0_CR[EREFSTEN and EREFSTEN] bits. Measured by entering all modes with the crystal enabled.							
	VLLS1	440	490	540	560	570	580	
	VLLS3	440	490	540	560	570	580	
	LLS	490	490	540	560	570	680	nA
	VLPS	510	560	560	560	610	680	
	STOP	510	560	560	560	610	680	
I_{CMP}	CMP peripheral adder measured by placing the device in VLLS1 mode with CMP enabled using the 6-bit DAC and a single external input for compare. Includes 6-bit DAC power consumption.	22	22	22	22	22	22	µA
I_{RTC}	RTC peripheral adder measured by placing the device in VLLS1 mode with external 32 kHz crystal enabled by means of the RTC_CR[OSCE] bit and the RTC ALARM set for 1 minute. Includes ERCLK32K (32 kHz external crystal) power consumption.	432	357	388	475	532	810	nA

- Table 6 shows current used by an enabled peripheral in a low-power mode

Currents (at 25°C) with Sorting

IADC		366	µA	IADC ADC peripheral adder combining the measured values at VDD and VDDA by placing the device in STOP or VLPS mode. ADC is configured for low power mode using the internal clock and continuous conversions.
ITPM				ITPM TPM peripheral adder measured by placing the device in STOP or VLPS mode with selected clock source configured for output compare generating 100Hz clock signal. No load is placed on the I/O generating the clock signal. Includes selected clock source and I/O switching currents.
	OSCERCLK (4MHz external crystal)	256	µA	
	MCGIRCLK (4MHz internal reference clock)	86	µA	
IUART				IUART UART peripheral adder measured by placing the device in STOP or VLPS mode with selected clock source waiting for RX data at 115200 baud rate. Includes selected clock source power consumption.
	OSCERCLK (4MHz external crystal)	237	µA	
	MCGIRCLK (4MHz internal reference clock)	66	µA	
IIREFSTEN4MHz		56	µA	IIREFSTEN4MHz 4 MHz internal reference clock (IRC) adder. Measured by entering STOP or VLPS mode with 4 MHz IRC enabled.
IIREFSTEN32KHz		52	µA	IIREFSTEN32KHz 32 kHz internal reference clock (IRC) adder. Measured by entering STOP mode with the 32 kHz IRC enabled.
IBG		45	µA	IBG Bandgap adder when BGEN bit is set and device is placed in VLPx, LLS, or VLLSx mode.
ICMP		22	µA	ICMP CMP peripheral adder measured by placing the device in VLLS1 mode with CMP enabled using the 6-bit DAC and a single external input for compare. Includes 6-bit DAC power consumption.
IEREFSTEN4MHz		22	µA	IEREFSTEN4MHz External 4MHz crystal clock adder. Measured by entering STOP or VLPS mode with the crystal enabled.
IEREFSTEN32KHz				IEREFSTEN32KHz External 32 kHz crystal clock adder by means of the OSC0_CR[EREFSTEN and EREFSTEN] bits. Measured by entering all modes with the crystal enabled.
	VLPS	560	nA	
	STOP	560	nA	
	VLLS1	490	nA	
	VLLS3	490	nA	
	LLS	490	nA	
IRTC		357	nA	IRTC RTC peripheral adder measured by placing the device in VLLS1 mode with external 32 kHz crystal enabled by means of the RTC_CR[OSCE] bit and the RTC ALARM set for 1 minute. Includes ERCLK32K (32 kHz external crystal) power consumption.

Low Power Modes vs. Module Operation

- What can work in low-power modes?
- Details in Section 7.5 of KL25 Sub-Family Reference Manual
- Key
  - FF: full functionality
  - static: non-functional, but state information in registers and memory is retained
  - powered, low power: non-functional, but memory is retained
  - OFF: not powered, is reset state on wake-up

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
Core modules						
NVIC	FF	FF	static	static	static	OFF
System modules						
Mode Controller	FF	FF	FF	FF	FF	FF
LLWU[1]	static	static	static	static	FF	FF[2]
Regulator	low power	low power	ON	low power	low power	low power in VLLS3, OFF in VLLS0/1
LVD	disabled	disabled	ON	disabled	disabled	disabled
Brown-out Detection	ON	ON	ON	ON	ON	ON in VLLS1/3, optionally disabled in VLLS0[3]
DMA	FF	FF	Async operation	Async operation	static	OFF
Watchdog	FF	FF	static	static	static	OFF

Low Power Modes vs. Module Operation

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
Memory and memory interfaces						
Flash	1 MHz max access - no program	low power	low power	low power	OFF	OFF
SRAM_U and SRAM_L	low power	low power	low power	low power	low power	low power in VLLS3, OFF in VLLS0/1
Clocks						
1kHz LPO	ON	ON	ON	ON	ON	ON in VLLS1/3, OFF in VLLS0
System oscillator (OSC)	OSCERCLK max of 16MHz crystal	OSCERCLK max of 16MHz crystal	OSCERCLK optional	OSCERCLK max of 16MHz crystal	limited to low range/low power	limited to low range/low power in VLLS1/3, OFF in VLLS0
MCG	4 MHz IRC	4 MHz IRC	static - MCGIRCLK optional; PLL optionally on but gated	static - MCGIRCLK optional	static - no clock output	OFF
Core clock	4 MHz max	OFF	OFF	OFF	OFF	OFF
System clock	4 MHz max	4 MHz max	OFF	OFF	OFF	OFF
Bus clock	1 MHz max	1 MHz max	OFF	OFF	OFF	OFF

Low Power Modes vs. Peripheral Module Operation

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
Timers						
TPM	FF	FF	FF with clocks	FF with clocks	static	OFF
PIT	FF	FF	static	static	static	OFF
LPTMR	FF	FF	FF	FF	FF	FF[4]
RTC	FF	FF	FF	FF	FF	FF[5]
Analog						
16-bit ADC	FF	FF	ADC internal clock only	ADC internal clock only	static	OFF
CMP[6]	FF	FF	HS or LS compare	HS or LS compare	LS compare	LS compare in VLLS1/3, OFF in VLLS0
6-bit DAC	FF	FF	static	static	static	static, OFF in VLLS0
12-bit DAC	FF	FF	static	static	static	static
Human-machine interfaces						
GPIO	FF	FF	wakeup	wakeup	static, pins latched	OFF, pins latched
TSI	FF	wakeup[7]	wakeup[7]	wakeup[7]	wakeup[7]	wakeup[7]

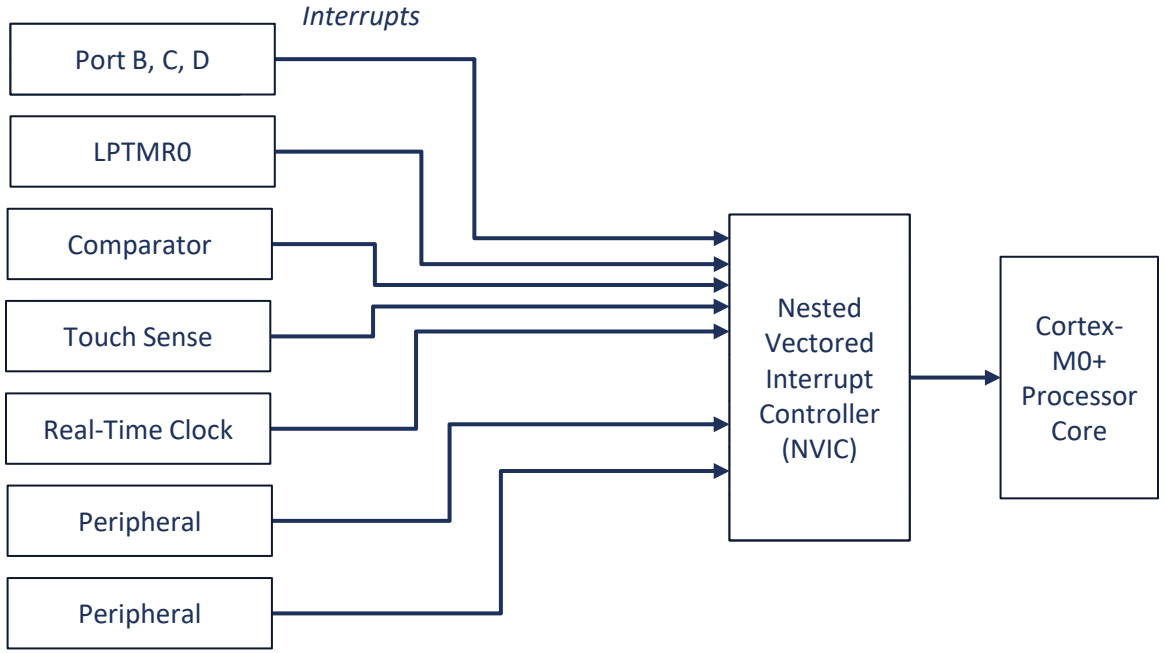
Low Power Modes vs. Peripheral Module Operation

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
Communication Interfaces						
USB FS/LS	static	static	static	static	static	OFF
USB Voltage Regulator	optional	optional	optional	optional	optional	optional
UART0	1 Mbps	1 Mbps	FF with clocks	FF with clocks	static	OFF
UART1 , UART2	62.5 kbps	62.5 kbps	static, wakeup on edge	static, wakeup on edge	static	OFF
SPI0	master mode 500 kbps, slave mode 250 kbps	master mode 500 kbps, slave mode 250 kbps	static, slave mode receive	static, slave mode receive	static	OFF
SPI1	master mode 2 Mbps, slave mode 1 Mbps	master mode 2 Mbps, slave mode 1 Mbps	static, slave mode receive	static, slave mode receive	static	OFF
I^2C0	50 kbps	50 kbps	static, address match wakeup	static, address match wakeup	static	OFF
I^2C1	50 kbps	50 kbps	static, address match wakeup	static, address match wakeup	static	OFF

Using Peripherals to Wake Up

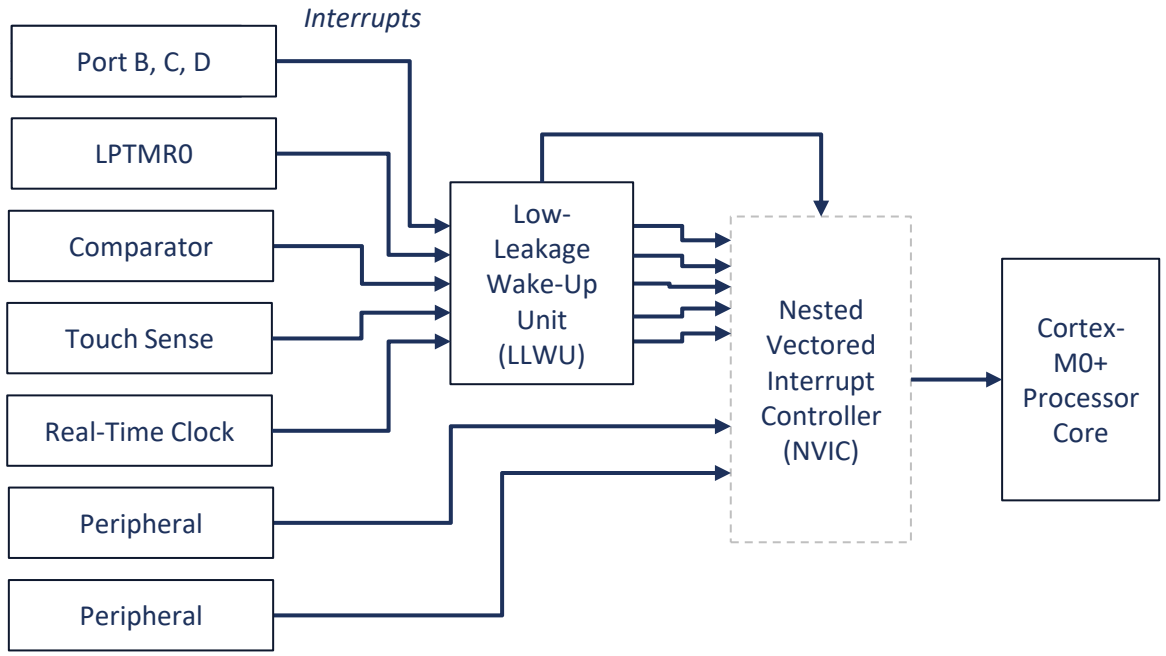
How Do We Wake Up?

Run and Wait Modes



- Run and wait modes
  - NVIC is awake

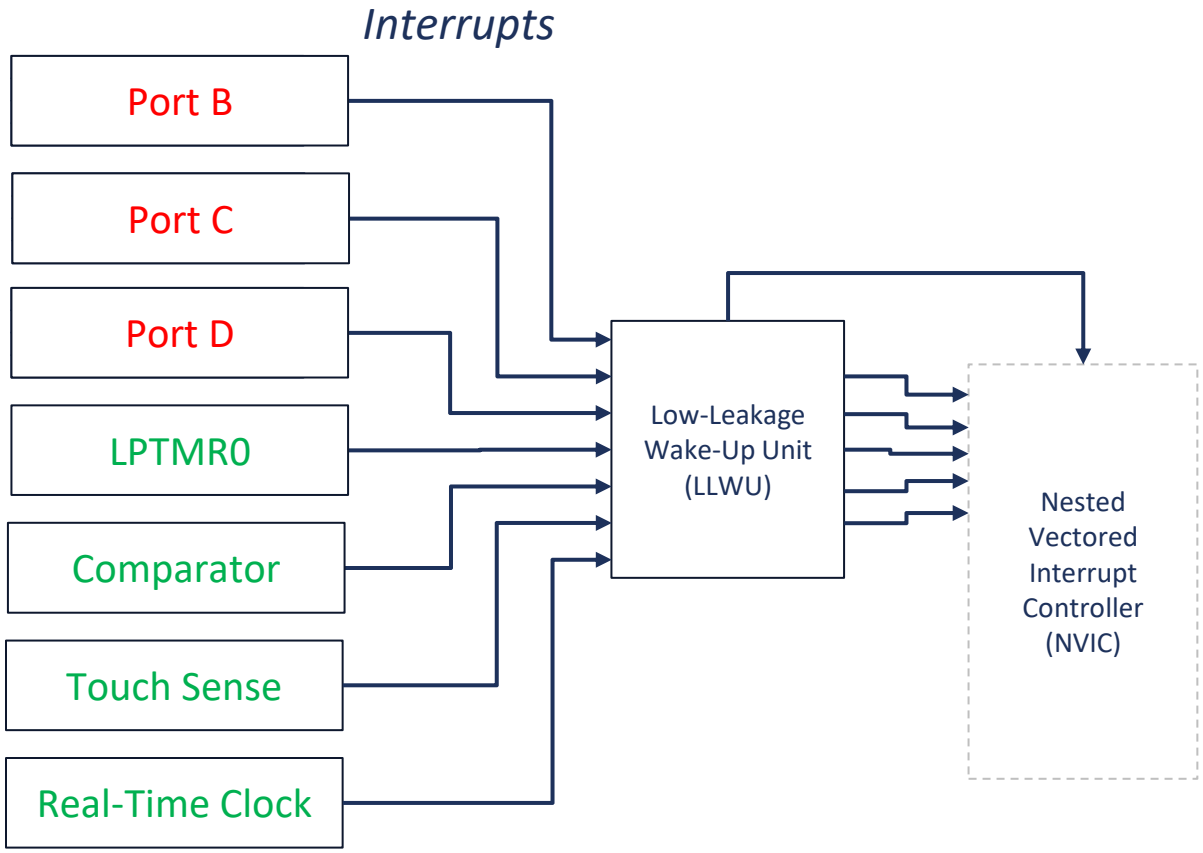
Stop Modes



- Stop modes
  - NVIC is not awake, so use low-leakage wakeup unit (LLWU)
    - LLWU has own interrupt vector
    - LLWU will wake up NVIC

Low-Leakage Wakeup Unit

- LLWU Features
  - Is activated only in low-leakage (LL) power modes
  - Uses very little power
  - Can wake up MCU or NVIC
- Wakeup Sources
  - Interrupts from edges on external pin inputs
  - Interrupt flags from other peripheral modules



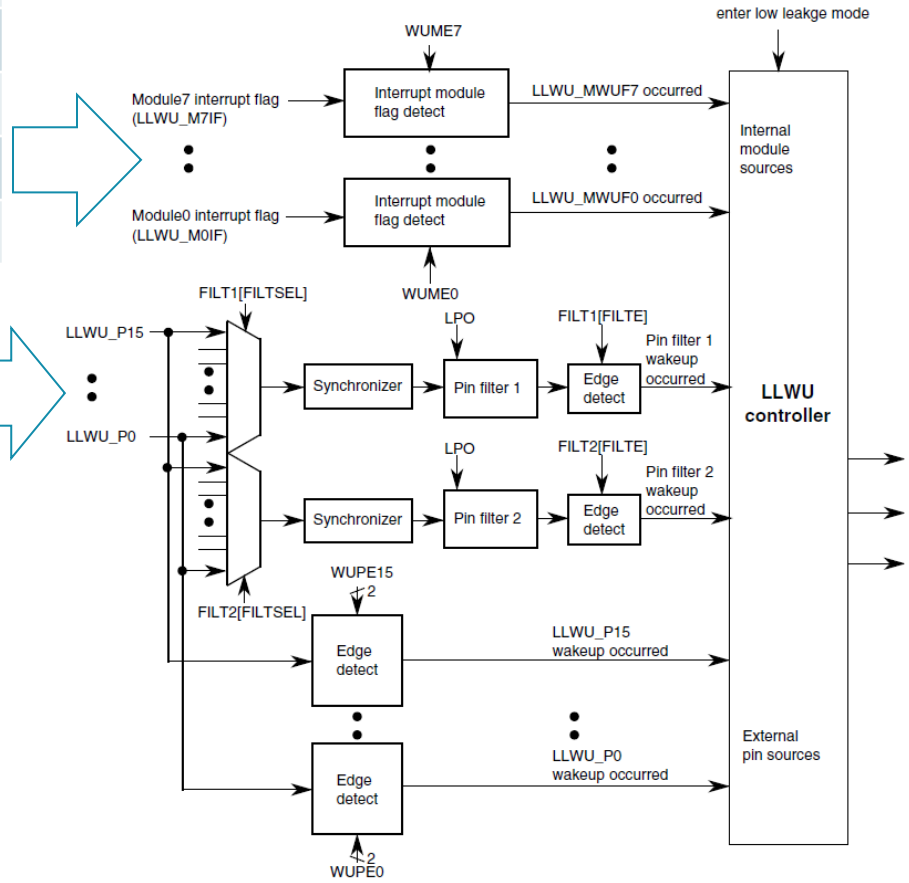
LLWU Wakeup Source Details

- Internal peripheral modules
  - LLWU_M0IF-M7IF
  - Interrupt request flags for LPTMR0, CMP0, etc. named LLWU_P0, LLWU_P1, etc.
  - Enabled by fields WUME0-WUME7 in register LLWU_ME
  - Wake up source indicated by MWUF flags in LLWU_F3

- Edges on external pin inputs
  - Pins PTB0, PTC1, etc. named LLWU_P0, LLWU_P1, etc.
  - Enabled by fields WUPE0-WUPE15 in registers LLWU_PE1-LLWU_PE4
  - Can filter out noise to prevent false wakeup
  - Wake up source indicated by WUF flags in LLWU_F1 and LLWU_F2

Input	Module
LLWU_M0IF	LPTMR0
LLWU_M1IF	CMP0
LLWU_M2IF	Reserved
LLWU_M3IF	Reserved
LLWU_M4IF	TSI0
LLWU_M5IF	RTC Alarm
LLWU_M6IF	Reserved
LLWU_M7IF	RTC Seconds

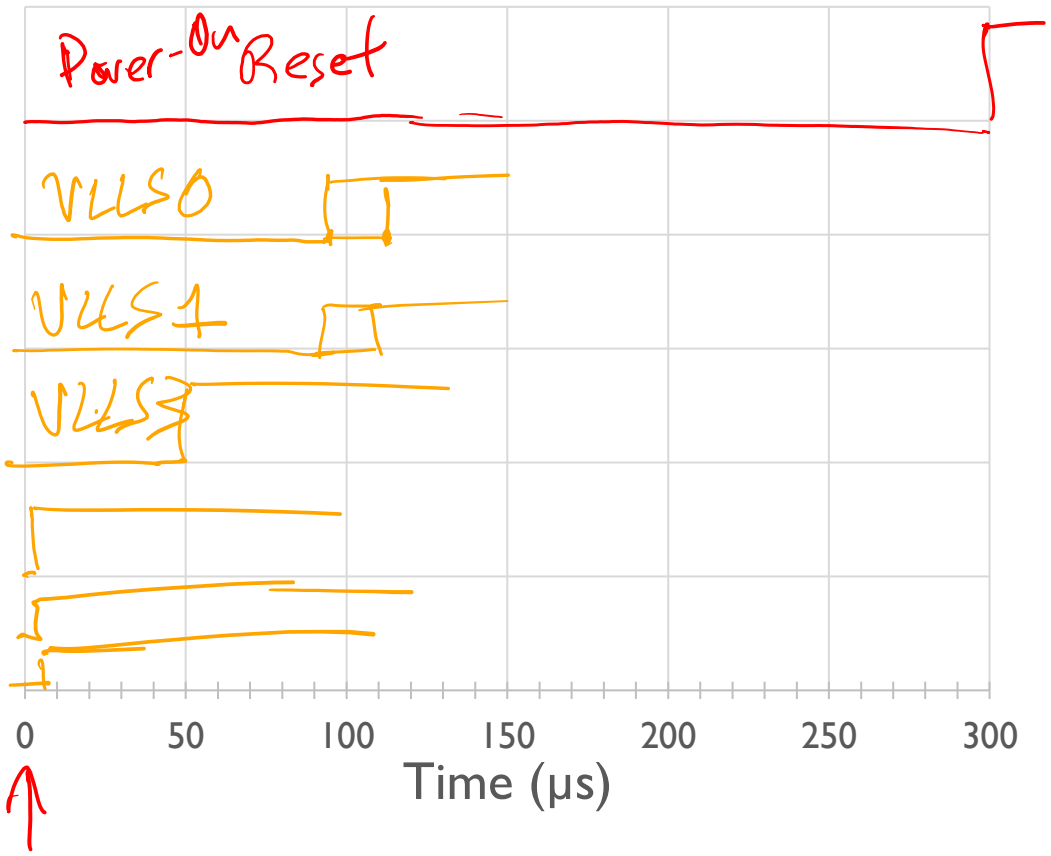
Input	Input Pin
LLWU_P5	PTB0
LLWU_P6	PTC1
LLWU_P7	PTC3
LLWU_P8	PTC4
LLWU_P9	PTC5
LLWU_P10	PTC6
LLWU_P14	PTD4
LLWU_P15	PTD6



Transition Delays - Visualized

Table 4. Power mode transition operating behaviors

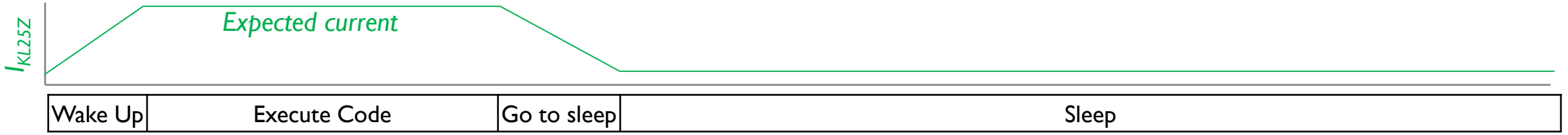
Symbol	Description	Min.	Typ.	Max.	Unit
t_{POR}	After a POR event, amount of time from the point V_{DD} reaches 1.8 V to execution of the first instruction across the operating temperature range of the chip.	—	—	300	µs
	• VLLS0 → RUN	—	95	115	µs
	• VLLS1 → RUN	—	93	115	µs
	• VLLS3 → RUN	—	42	53	µs
	• LLS → RUN	—	4	4.6	µs
	• VLPS → RUN	—	4	4.4	µs
	• STOP → RUN	—	4	4.4	µs



- It takes time to start running again, depending on situation
- Using reset:
  - Up to 300 microseconds
- Using an interrupt
  - ~ 100 microseconds to come out of VLLS0 or VLLS1
  - ~ 50 microseconds to come out of VLLS3
  - ~4 microseconds to come out of LLS, VLPS or STOP

Putting Together the Wake-Up Pieces

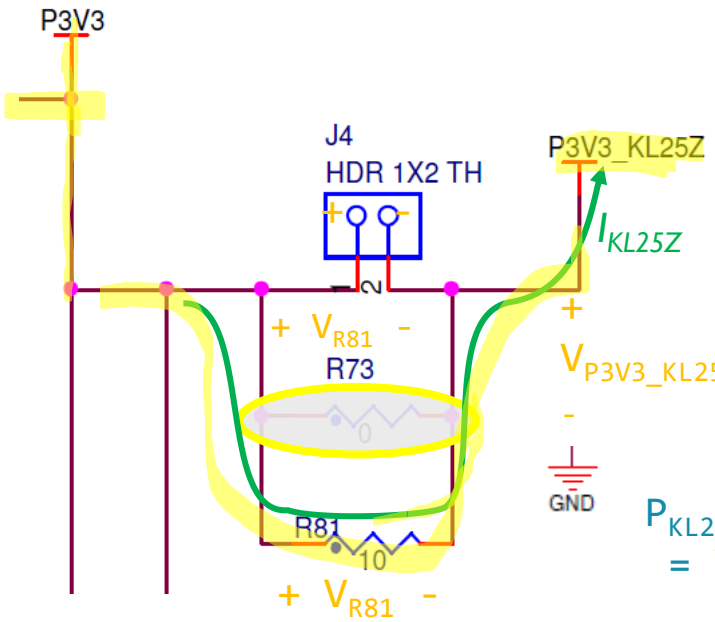
Reality Check on System Using Sleep and Wake-Up



- Reality check on a simple program
  - Wake up, do a little work, go back to sleep
  - Does our mental model match reality?
    - Did we under/overestimate something important?
    - Did we forget something important?
    - Did we not know something important?

Activity	Time Taken t	Power Use P	Energy = $\int Pdt$
Wake Up			
Execute Code			
Go to sleep			
Sleep			

- Let's examine time taken and power drawn for each step with scope.
  - Measure voltage V_{R81} across J4 (P_KL25 on PCB). Not ground-referenced!
  - MCU current $I_{KL25Z} = V_{R81}$ / R81
  - Power = $V_{P3V3\_KL25Z} * I_{KL25Z}$

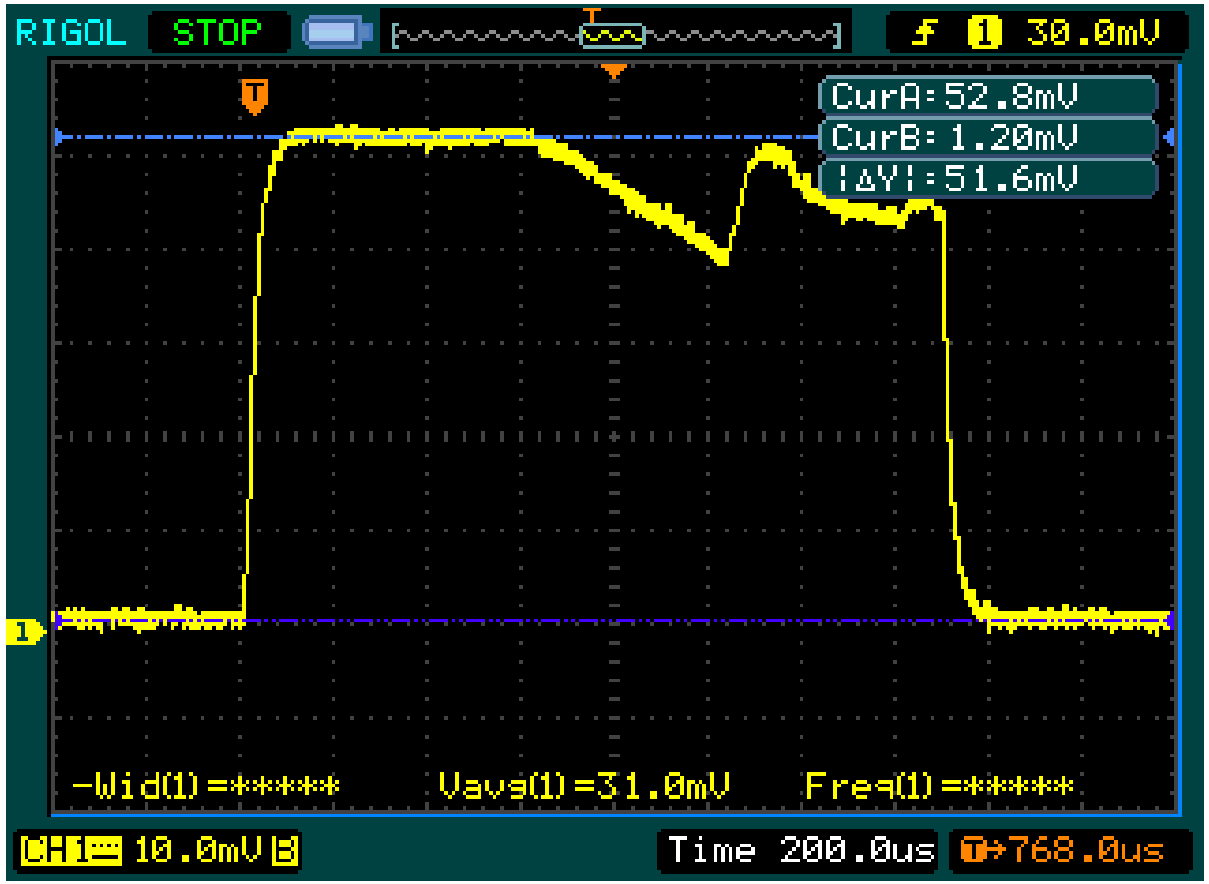


$I_{KL25Z} = V_{R81}/10\Omega$

$P_{KL25Z} = V_{P3V3\_KL25Z} * I_{KL25Z}$
$= V_{P3V3\_KL25Z} * V_{R81}/10\Omega$

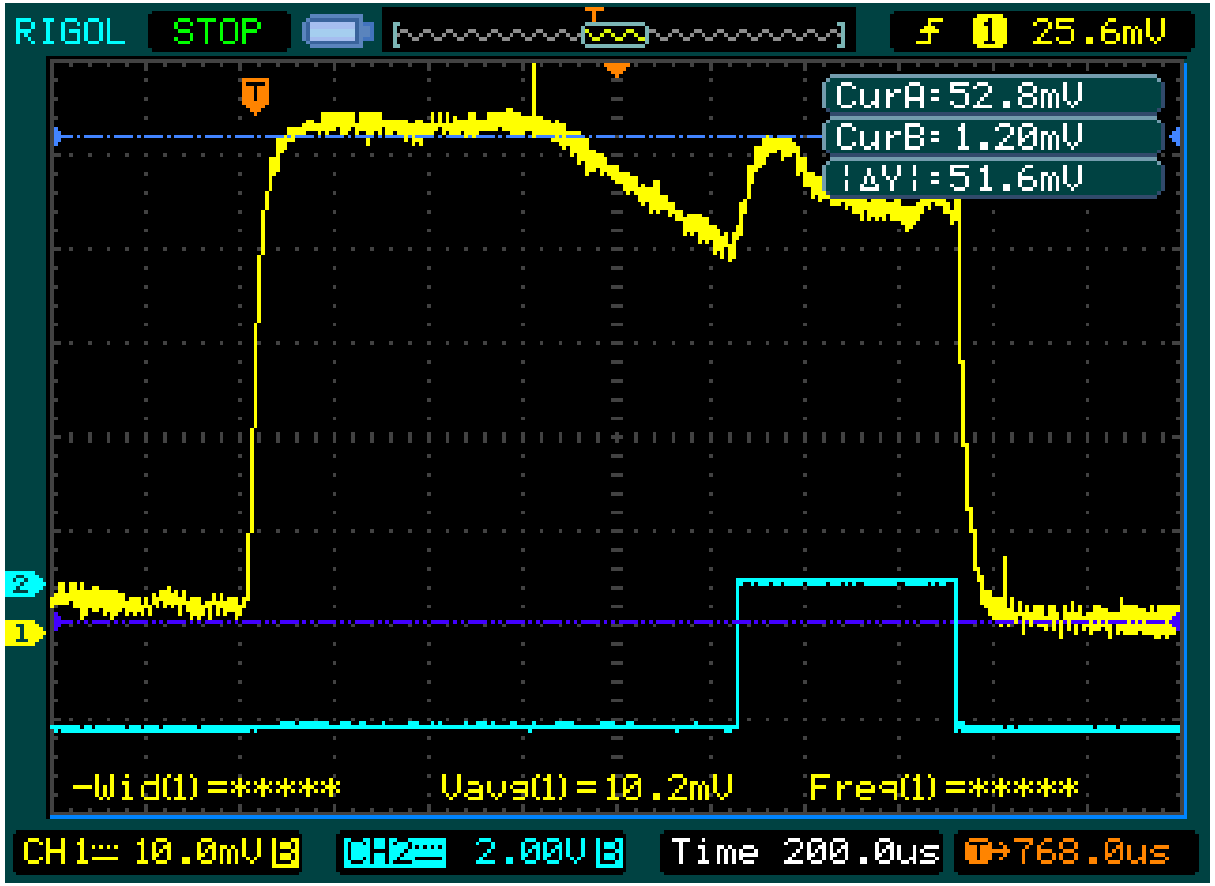
Profiling Power and Energy Consumption

- Current consumption
  - Min: 1.2 mV/10 Ω = 0.12 mA
  - Max: 52.8 mV/10 Ω = 5.28 mA
- Power consumption (at 3.07 V)
  - Min: 0.12 mA * 3.07 V = 0.37 mW
  - Max: 5.28 mA * 3.07 V = 16.2 mW
- Strange trace shape. What is happening here?
- Think: Duration is about 1.5 ms
  - f_{CPU} = 48 MHz, so about 72,000 clock cycles. More than a "little" work!
- Do: Let's find out when the CPU is awake by using an output bit. Awake: 1, asleep: 0
  - Set to 1 on reset, entry to ISR, immediately after WFI instruction
  - Clear to 0 immediately before WFI



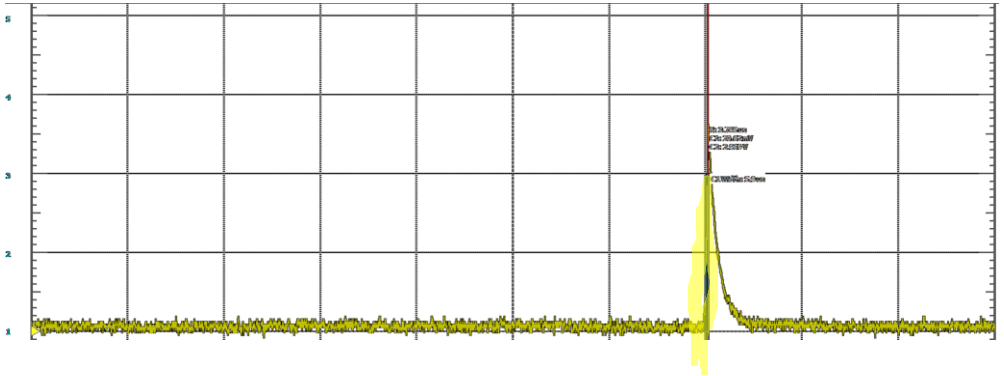
The Plot Thickens

- Current rises long before ISR starts running (by about 1 ms)
- 1 ms is long compared to the ISR duration (0.5 ms), leading to a lot of extra power and energy consumption
- What is happening before the ISR?

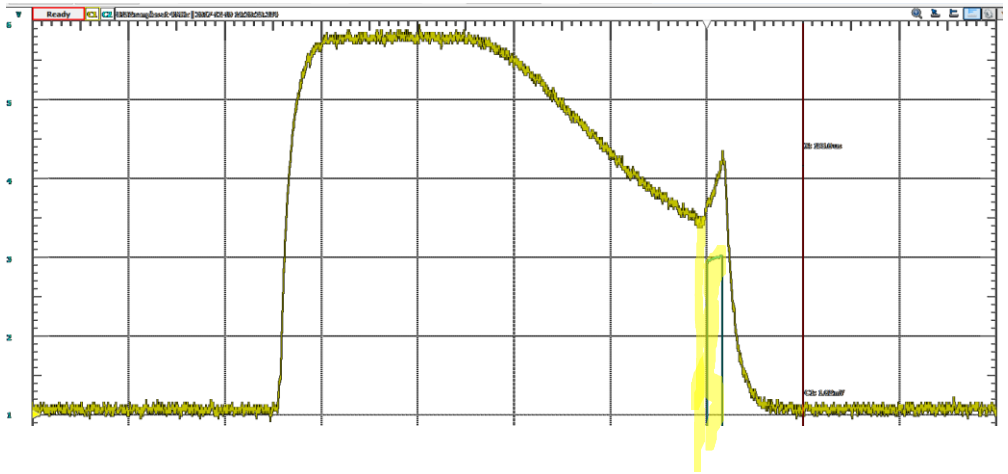


Try Different Clock Setup Modes

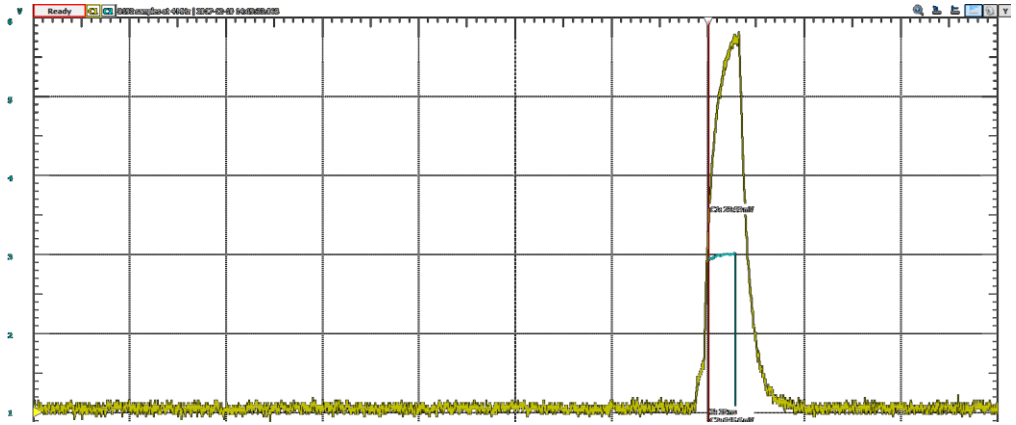
- Same program, different modes (CLOCK_SETUP=0-3)
- Settings: H: 200 us/div, V: 1 mA/div
- Mode 0: Internal oscillator (no crystal)



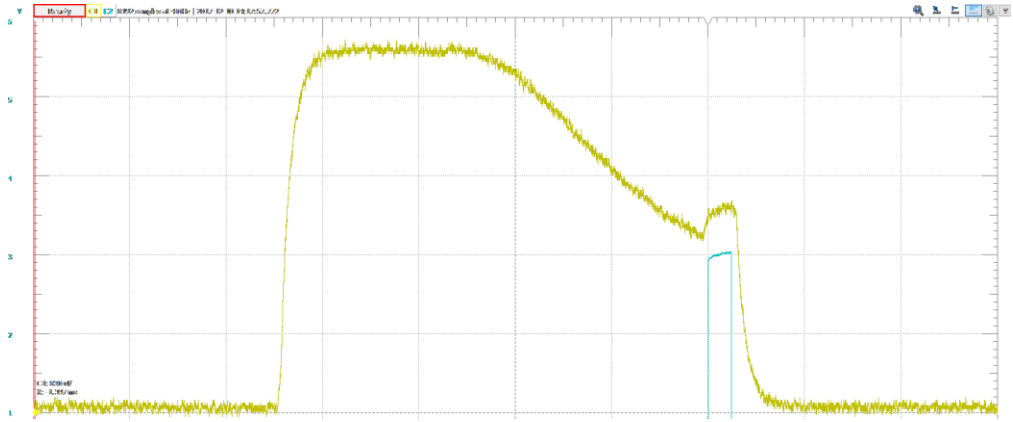
- Mode 1: External crystal + system oscillator



- Mode 2: Internal oscillator (no crystal)

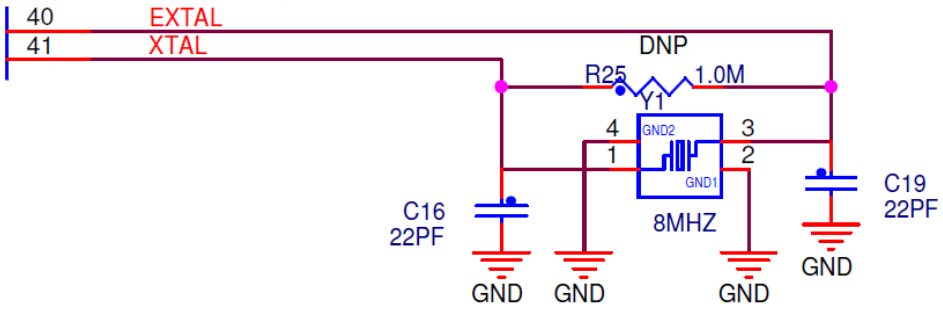


- Mode 3: External crystal + system oscillator



- Let's superimpose the plots to make it easier to evaluate differences

Oscillator Start-Up Delay



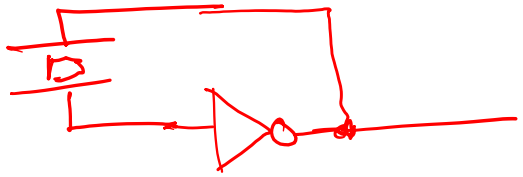


Table 14. Oscillator frequency specifications

Symbol	Description	Typ.	Unit
t_{cst}	Crystal startup time — 32 kHz low-frequency, low-power mode (HGO=0)	750	ms
	Crystal startup time — 32 kHz low-frequency, high-gain mode (HGO=1)	250	ms
	Crystal startup time — 8 MHz high-frequency (MCG_C2[RANGE]=01), low-power mode (HGO=0)	0.6	ms
	Crystal startup time — 8 MHz high-frequency (MCG_C2[RANGE]=01), high-gain mode (HGO=1)	1	ms

Osc. Period

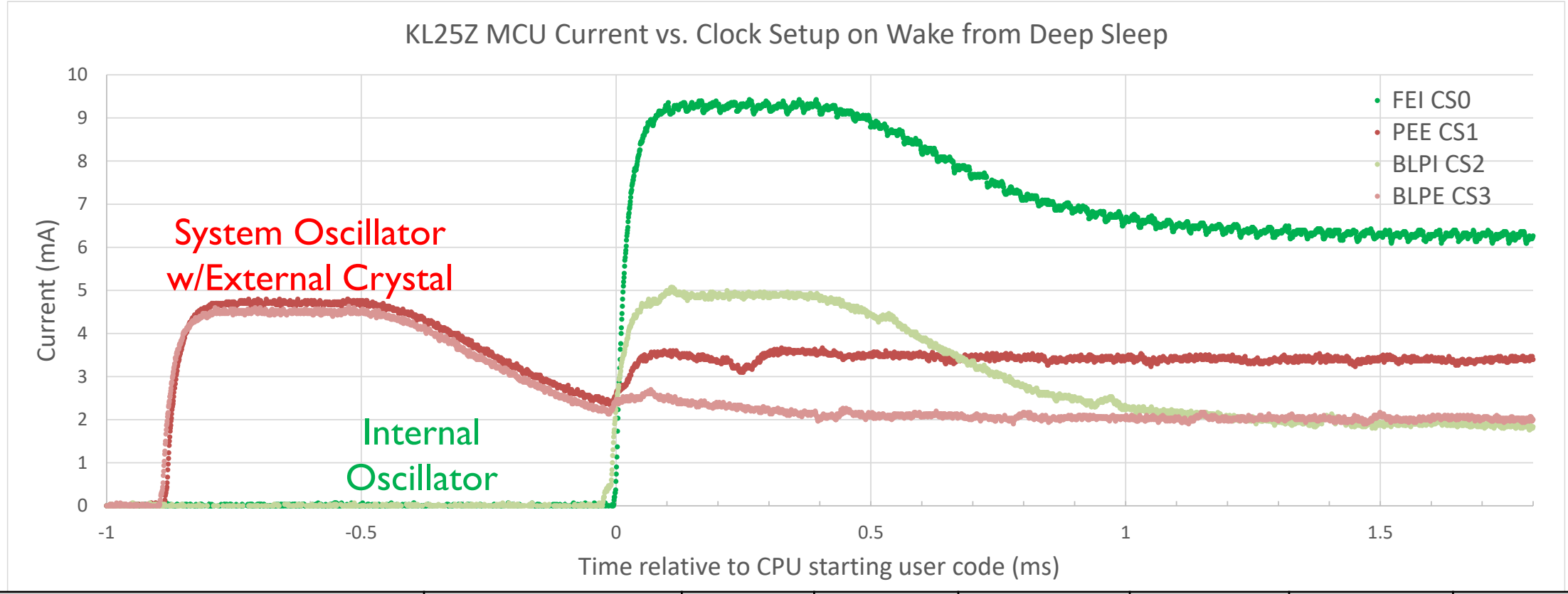
1/(32 kHz) = 30.5 µs

1/(8 MHz) = 0.125 us

Table 13. Oscillator DC electrical specifica

Symbol	Description	Typ.	Unit
V_{DD}	Supply voltage	—	V
I_{DDOSC}	Supply current — low-power mode (HGO=0)		
	• 32 kHz	500	nA
	• 4 MHz	200	µA
	• 8 MHz (RANGE=01)	300	µA
	• 16 MHz	950	µA
	• 24 MHz	1.2	mA
	• 32 MHz	1.5	mA
I_{DDOSC}	Supply current — high gain mode (HGO=1)		
	• 32 kHz	25	µA
	• 4 MHz	400	µA
	• 8 MHz (RANGE=01)	500	µA
	• 16 MHz	2.5	mA
	• 24 MHz	3	mA
	• 32 MHz	4	mA

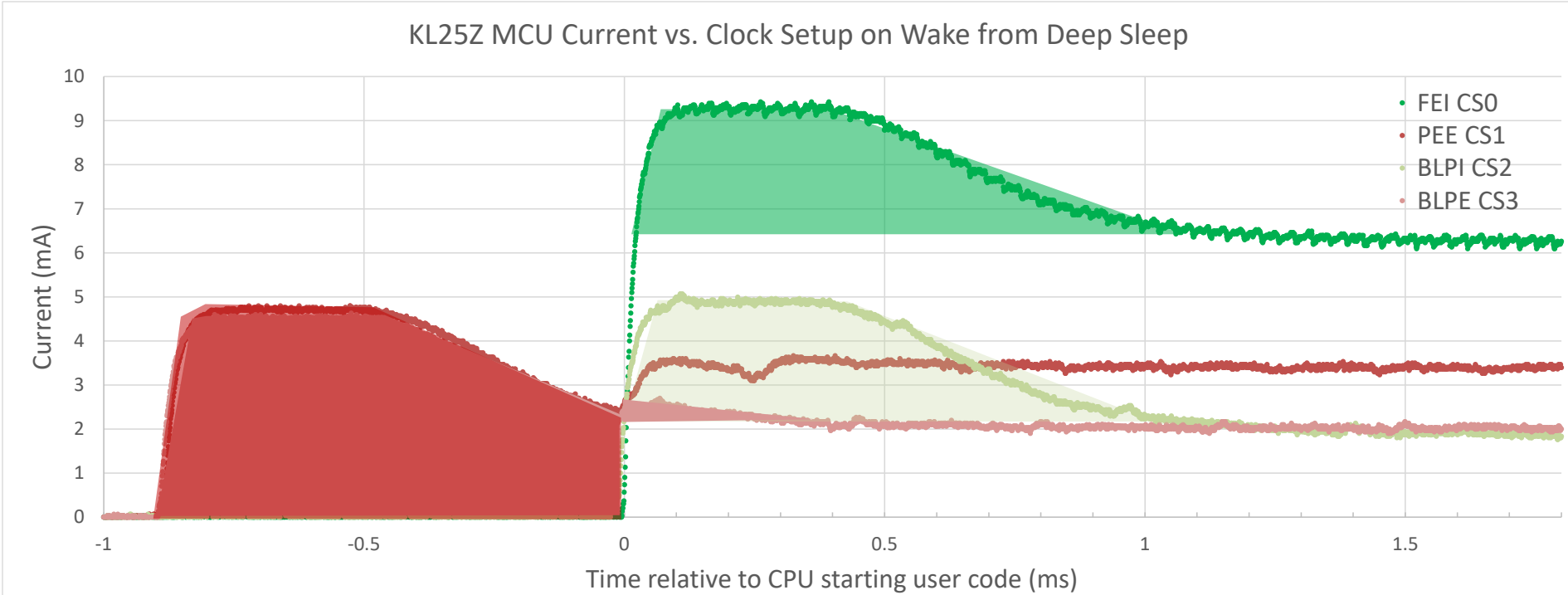
MCU Current vs. Clock Setup (V_{CC} = 3.07V)



CLOCK _SETUP	MCG Mode Name	Clock Source	FLL	PLL	CPU Freq.	Bus Freq.	Final MCU Current	Final Avg. MCU Power
0	FEI: FLL Engaged Internal	Slow internal ref.	Engaged	Disabled*	20.9 MHz	20.9 MHz	6.3 mA	19.34 mW
1	PEE: PLL Engaged External	System oscillator ref.	Disabled	Engaged	48 MHz	24 MHz	3.4 mA	10.44 mW
2	BLPI: Bypassed Low-Power Internal	Fast internal ref.	Disabled	Disabled	4 MHz	0.8 MHz	1.9 mA	5.83 mW
3	BLPE: Bypassed Low-Power External	System oscillator ref.	Disabled	Disabled	4 MHz	1 MHz	2.1 mA	6.45 mW

Quantifying Wake-Up Energy Costs

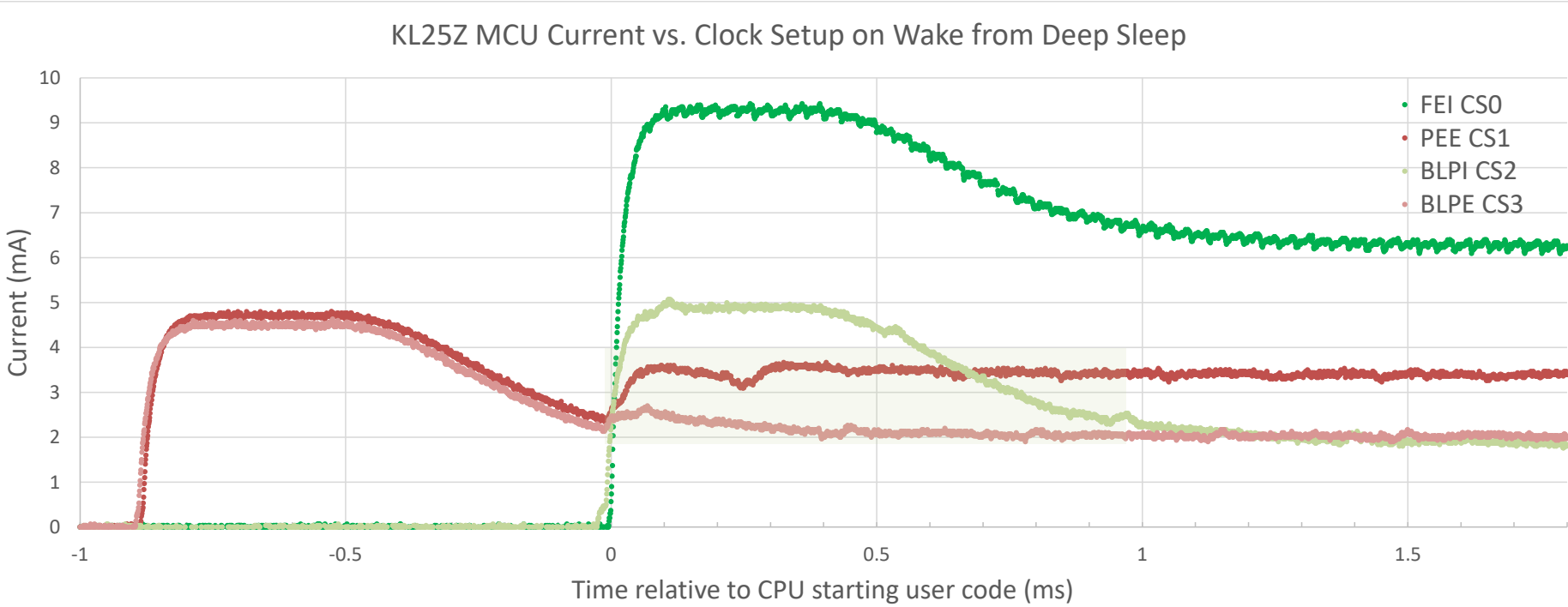
- Examine burst of current for wake-up
- Calculate energy based on area in shaded regions
  - Energy = time * current * supply voltage



CLOCK _MODE	MCG Mode	Supply Voltage	Region 1 Avg. Current (mA)	Region 1 Time (us)	Region 2 Avg. Current (mA)	Region 2 Time (us)	Region 3 Avg. Current (mA)	Region 3 Time (us)	Total Wake-up Energy (uJ)
0	FEI	3.07	3	400	1.5	700	0	0	6.9
1	PEE	3.07	4.8	400	3.5	420	0	0	10.4
2	BLPI	3.07	2.9	380	1.45	600	0	0	6.1
3	BLPE	3.07	4.5	400	3.3	420	0.25	300	10.0

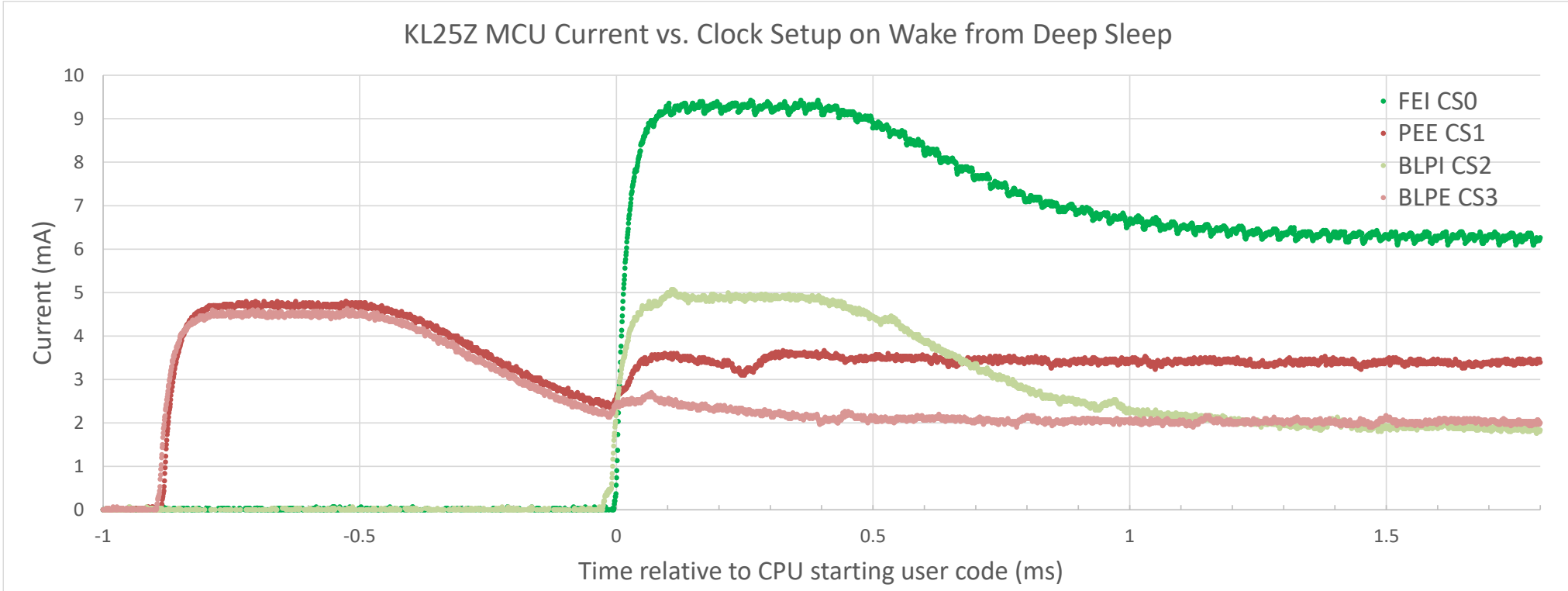
Modeling Wake-Up Power Use

- What is average power use for periodic wake-up at 10 Hz?
  - 6.9 uJ*10Hz = 6.9 uWs*10/s = 69 uW
- Raise it to 100 Hz?
  - 6.9 uJ*100Hz = 6.9 uWs*100/s = 690 uW
- Can find average power used during each wake-up
  - Divide energy by wake-up duration
  - E.g. 6.9 uJ/1100 us = 0.00627 W = 6.27 mW



CLOCK _MODE	MCG Mode	During Wake-Up	
		Duration (us)	Avg. Power(mW)
0	FEI	1100	6.28
1	PEE	820	12.69
2	BLPI	980	6.18
3	BLPE	820	12.21

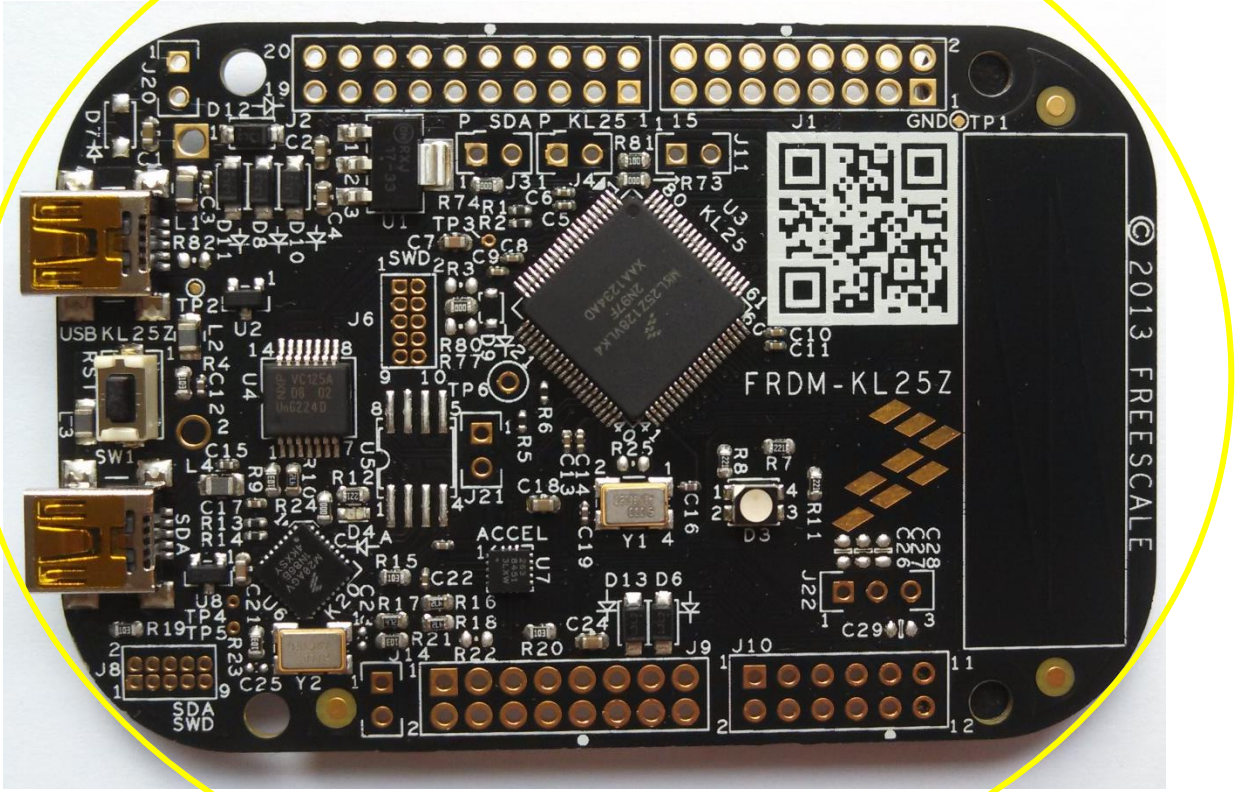
Summary of Power and Wake-Up Energy Use



CLOCK	MCG Mode	Clock Source	CPU Freq.	Bus Freq.	MCU Current	Energy per Wake-Up	Wake-Up Latency
0	FEI	Slow internal ref.	20.9 MHz	20.9 MHz	6.3 mA	6.91 µJ	~5 µs
1	PEE	System oscillator ref.	48 MHz	24 MHz	3.4 mA	12.69 µJ	820 µs
2	BLPI	Fast internal ref.	4 MHz	0.8 MHz	1.9 mA	6.18 µJ	~20 µs
3	BLPE	System oscillator ref.	4 MHz	1 MHz	2.1 mA	12.21 µJ	820 µs

Overview

- Cortex-M0+ CPU Core
- KL25Z MCU and Peripherals
- FRDM-KL25Z Development Board

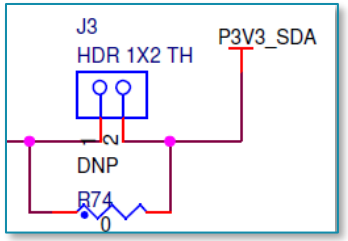


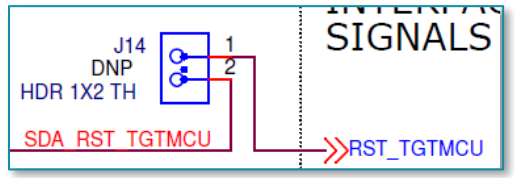
FRDM-KL25Z Features for Low-Power or Low-Energy

- SDA debug MCU can be disabled
- Linear Voltage Regulator can be bypassed
- High-Efficiency LEDs
- Accelerometer

FRDM-KL25Z Features for Low-Power or Low-Energy

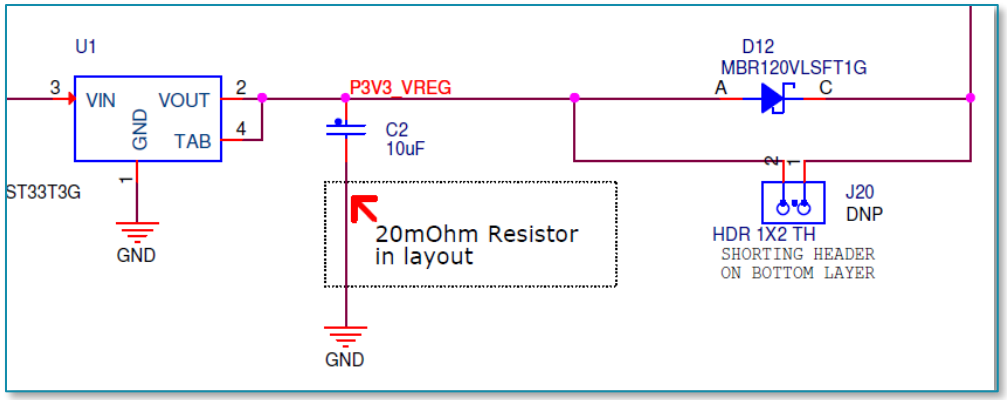
- SDA debug MCU can be disabled
  - Disconnect power (J3: P_SDA) and target MCU reset signal (J14)
  - If you remove J3 but keep J14, target MCU is held in reset





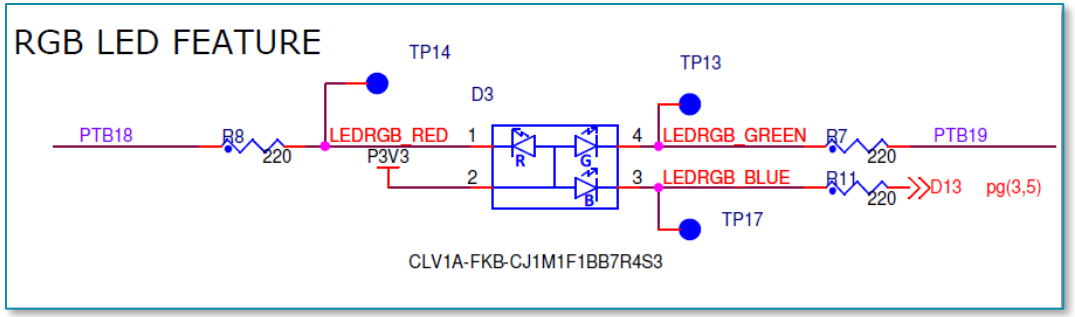
FRDM-KL25Z Features for Low-Power or Low-Energy

- Linear Voltage Regulator – can be bypassed
  - Supply power directly to P3V3 rail (e.g. with ultracapacitor)
  - Prevent P3V3 from back-driving regulator, wasting power: Cut trace across J20 which shorts diode D12 on regulator output.



FRDM-KL25Z Features for Low-Power or Low-Energy

- High-Efficiency LEDs
  - Don't need much current (~2 mA) to be very bright
  - Can pulse-width modulate them for dimming
  - Can change series resistors



- Accelerometer
  - Sleep and auto-sleep modes available
  - Interrupts to wake up MCU
  - Adjustable sampling rate