Microcontroller: Clocks, Power Management, Interrupts

Amarjeet Singh

January 23, 2013

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Logistics

- Assignment-1
  - Did everyone finish it?
  - Do you now have your laptop setup for embedded programming, you know the programming process and what all you need to do to create a simple program – improved confidence about hardware programming?
  - Will let you know the date for demo session – one of the days during lunch time

- Assignment-2
  - Start working now: Understanding control panel program is non-trivial
    - You can leave control panel program completely and just enhance your first assignment to now read the data and display it rather than displaying a static string
  - Get the sensor from the lab: Analog Temperature Sensor (LM35)
  - Expected to know the broad registers being used – ADC functionality is complex and the details will be eventually discussed in the class

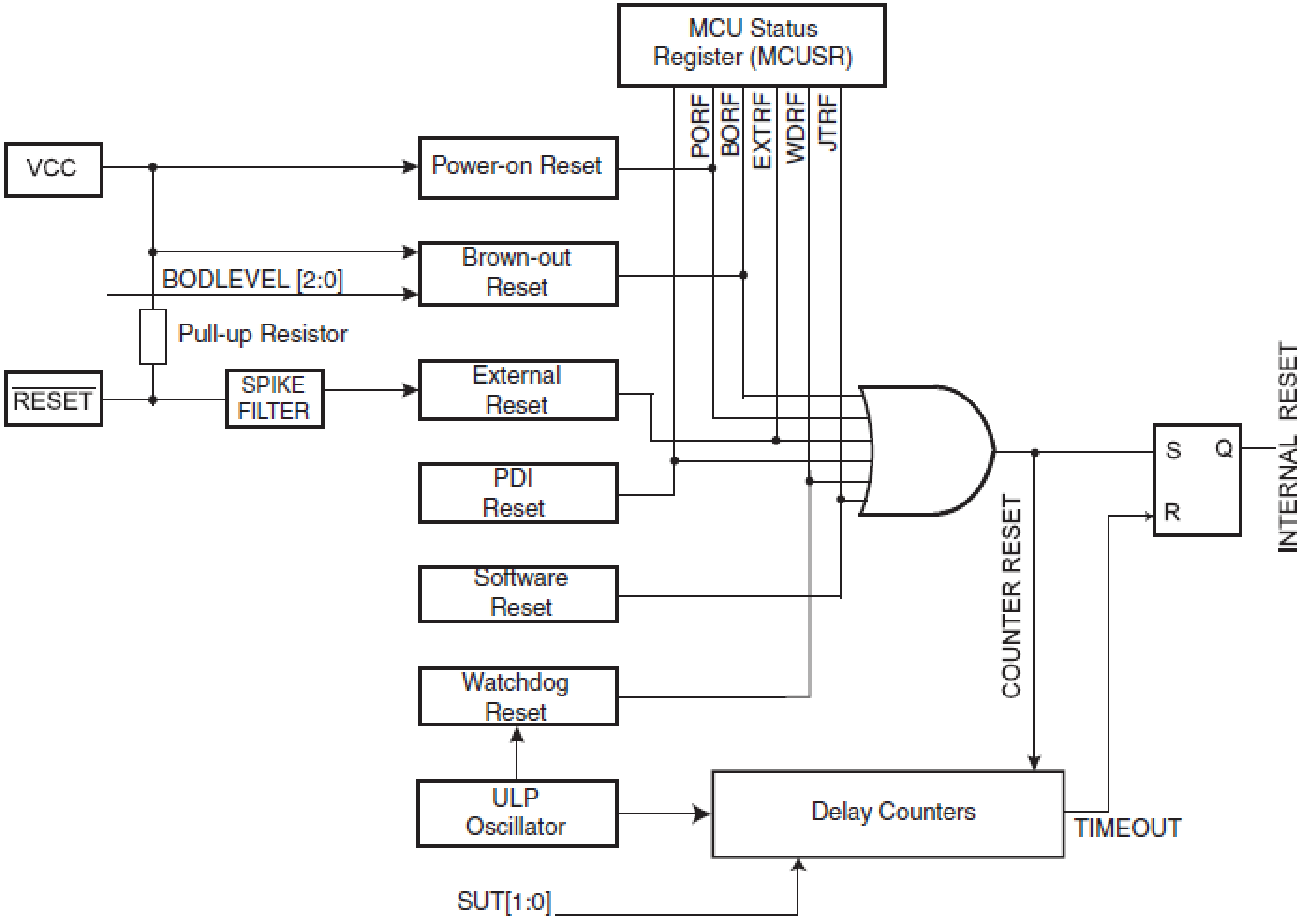
Logistics

- Website updated with slides from last lecture
- Friday is a holiday
- 3 classes next week (Tuesday, Friday, Saturday – Friday Timetable)
  - Saturday class to be taken by TA (traveling out of station for a workshop)
- Code committing in SVN

Revision from last class

- What different registers will be associated with I/O pin configuration?
- What is input sensing? Is it only clock based?
- What are intuitively the common sources of system reset?
- What additional is done in internal reset after all the external reset sources are removed?

Xmega-A1: Reset Logic



Xmega-A1: Registers for Reset Control

Bit	7	6	5	4	3	2	1	0	
+0x00	–	–	SRF	PDIRF	WDRF	BORF	EXTRF	PORF	STATUS
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

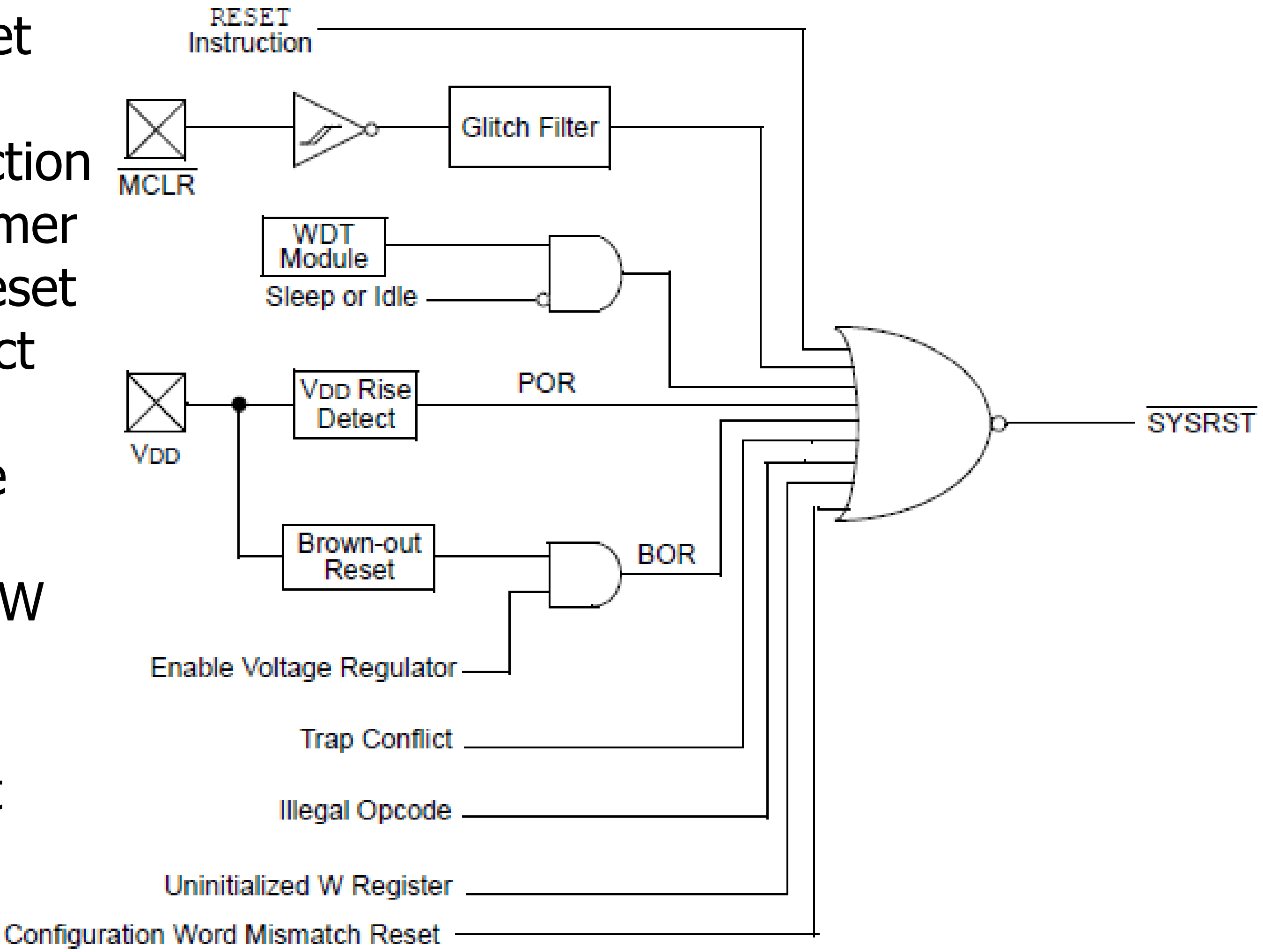
- Reset Status Register: Flag is set if the corresponding reset occurs and cleared by power-on reset or by writing a one to the bit location
    - PORF cleared by explicitly writing 1 to that location

Bit	7	6	5	4	3	2	1	0	
+0x01	–	–	–	–	–	–	–	SWRST	CTRL
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Reset Control Register: To set the bit for executing software reset

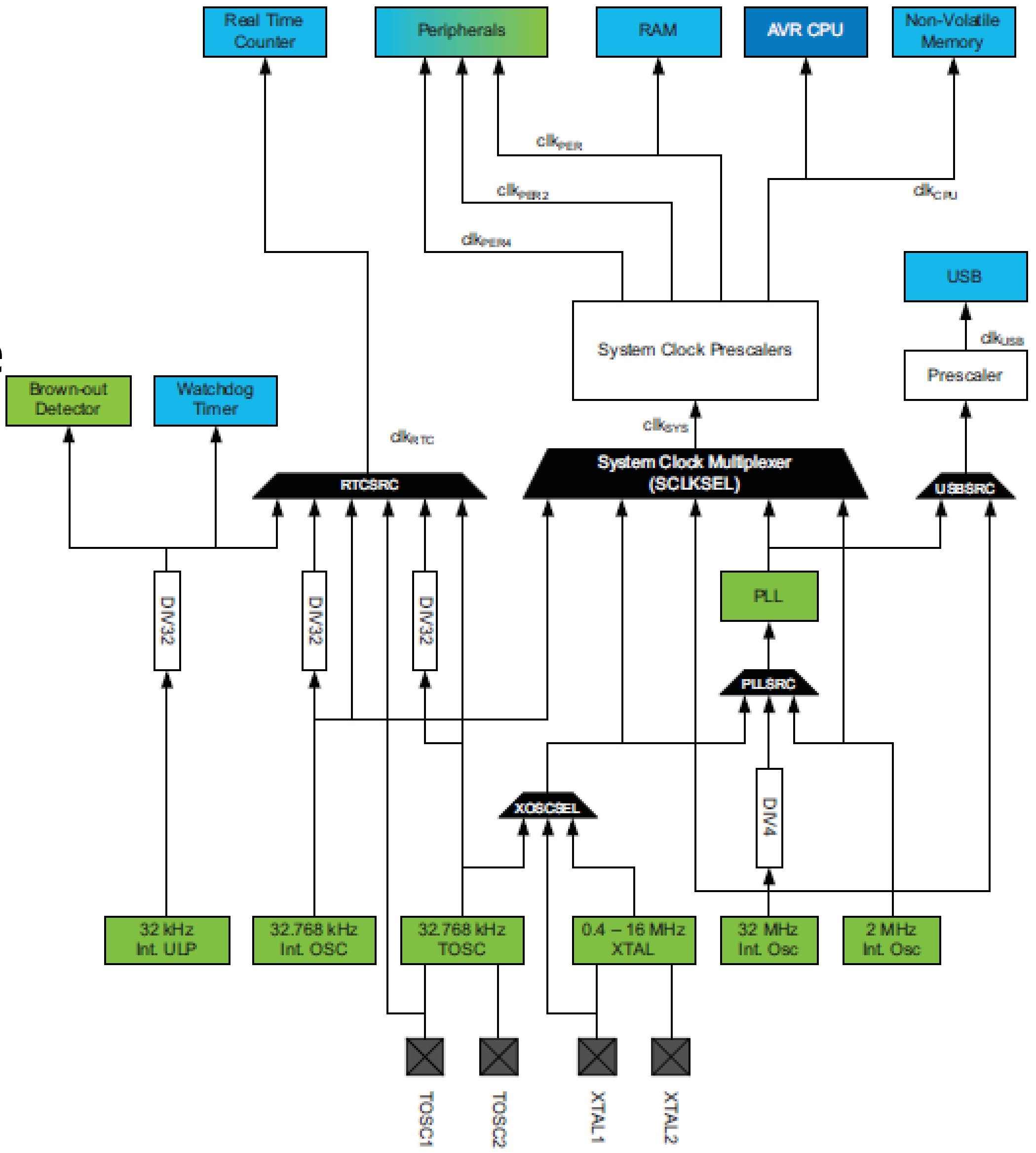
PIC24F: Reset Logic

- POR: Power on Reset
- MCLR: Pin Reset
- SWR: Reset Instruction
- WDT: Watchdog Timer
- BOR: Brown-out Reset
- TRAPR: Trap Conflict Reset
- IOP: Illegal Opcode Reset
- UWR: Uninitialized W Register
- CM: Configuration Word Mismatch Reset



xMega-A1: Clock Sources

- Supports both internal and external clock sources
- Phase Lock Loop (PLL) and clock pre-scalers can be used to generate wide range of frequencies
- At reset, all clock sources except 32 KHz ULP (Ultra Low Power) oscillator are disabled
- After reset device always starts up running from internal 2 MHz oscillator
  - Clock sources can be changed any time during normal operation

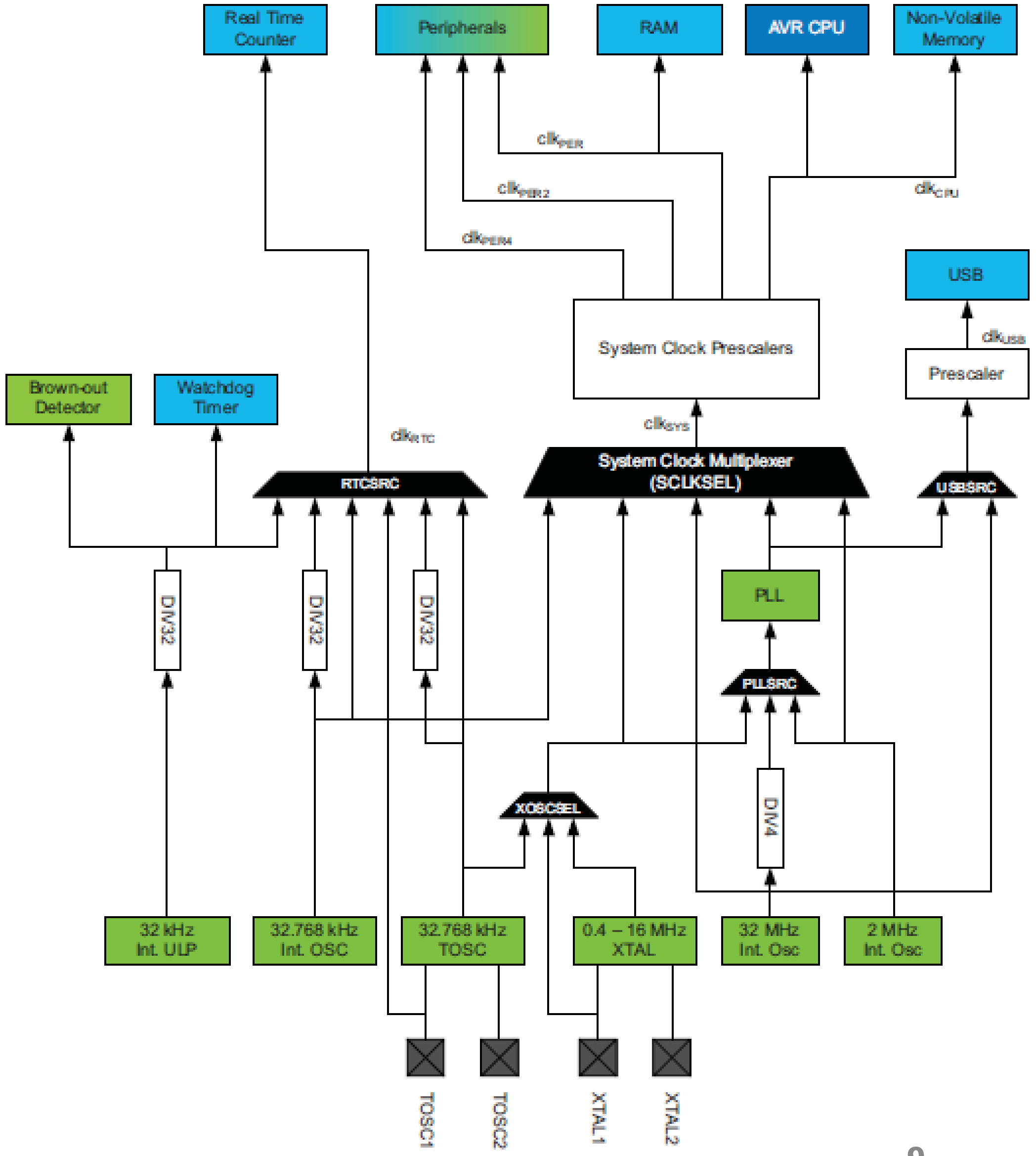


xMega-A1: Clock Distribution

- Not all the clocks need to be active at a given time
  - Clock for CPU and peripherals can be stopped during sleep mode

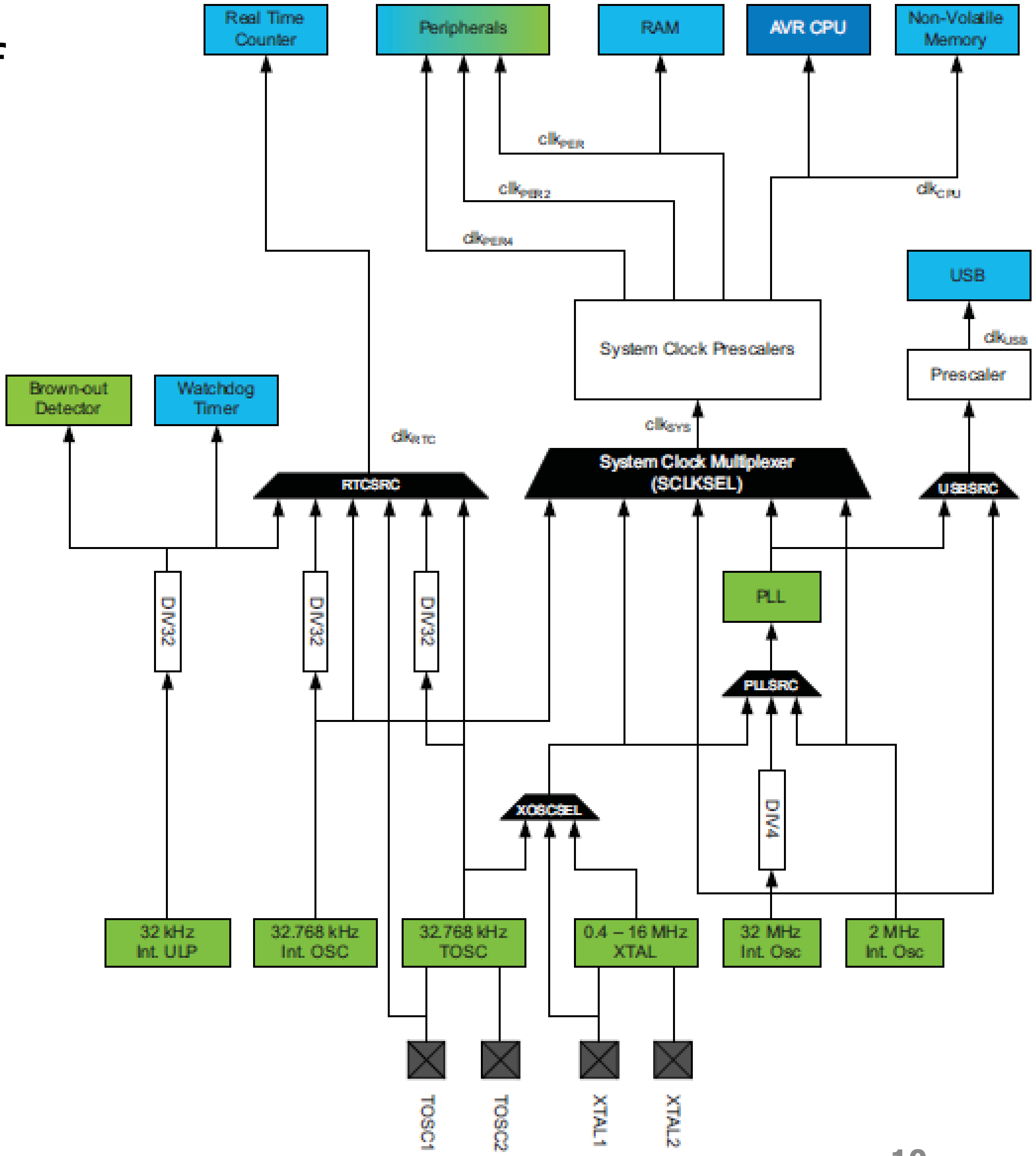
- clk_{SYS}: Output of main system clock selection

- clk_{CPU}: Routed to CPU and non-volatile memory (Flash)
  - Halting the clock inhibits CPU from executing instructions



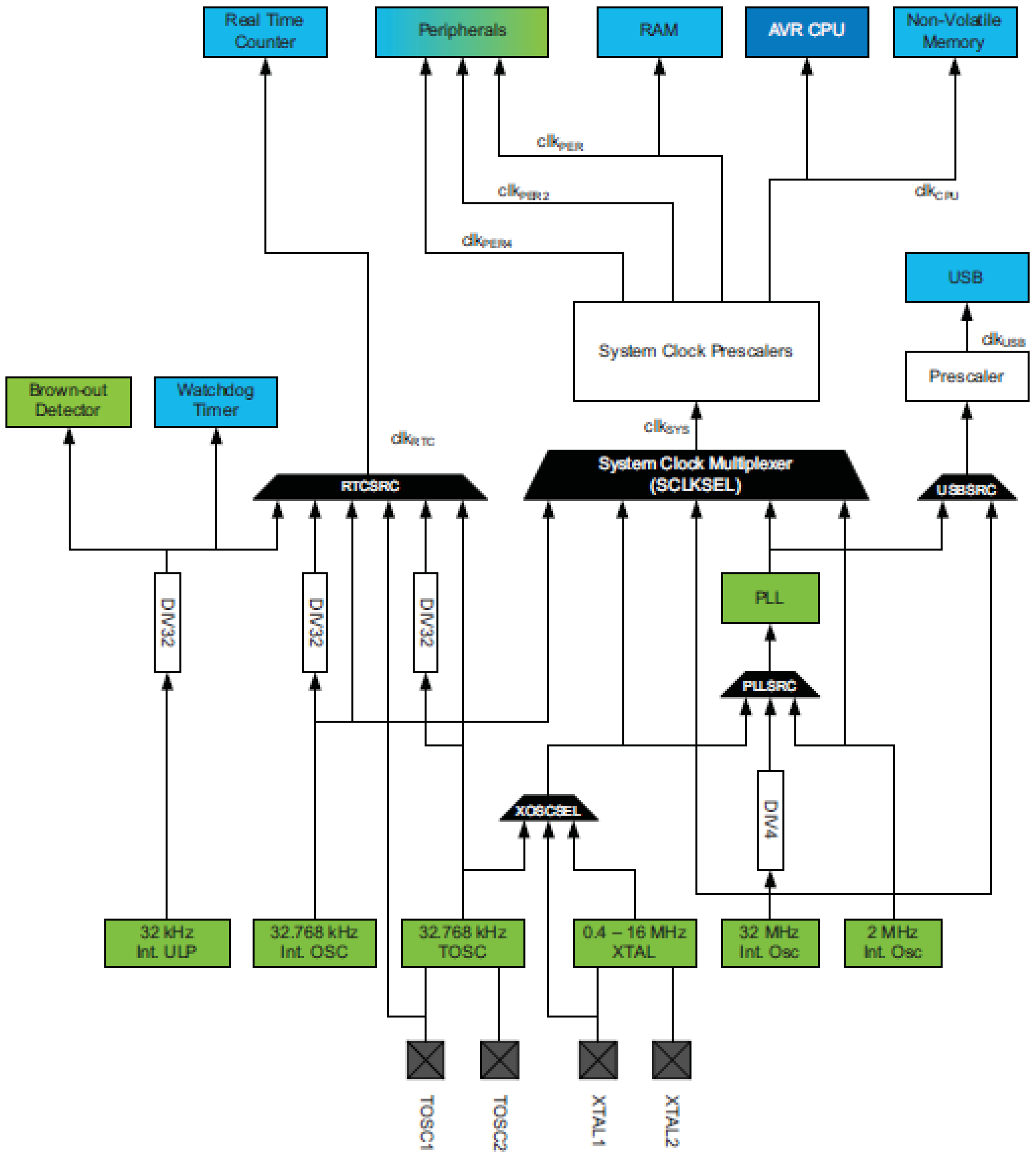
xMega-A1: Clock Distribution

- clk_{PER}: Used by majority of peripherals and system modules (DMAC, RAM, interrupt controller..)
  - Synchronous to clk_{CPU} but may run even when clk_{CPU} is turned off
- $clk_{PER\ 2x/4x}$: Modules that can run 2x/4x CPU clock can use these clocks
- clk_{USB}: Clock for USB module (12/48 MHz)



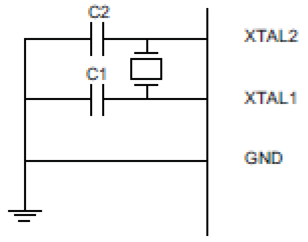
xMega-A1: Clock Distribution

- clk_{RTC}: Clock for real time counter (RTC), clocked asynchronously directly from external/internal 32.768 KHz or ULP Oscillator
  - Dedicated clock domain allows the operation of this peripheral even when the device is in sleep mode and rest of the clocks are stopped

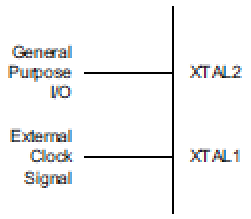


xMega-A1: Connecting External Clocks

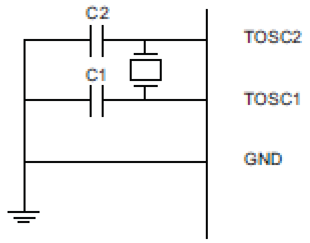
- 0.4 MHz – 16 MHz crystal Oscillator



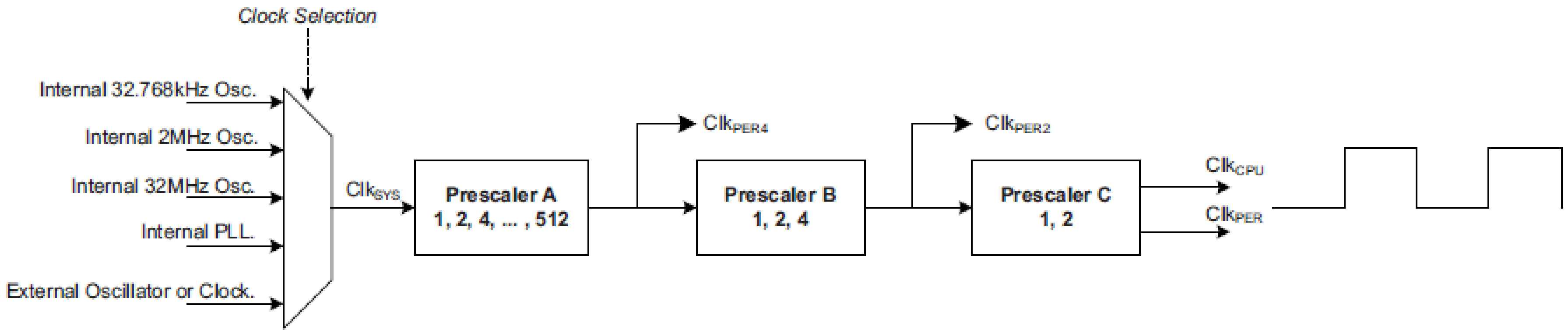
- External Clock Input



- 32.768 KHz Crystal Oscillator



xMega-A1: Clock Selection/ Prescalers



- System clock source selectable from software
- System clock fed into prescaler that can divide the clock by a factor from 1 to 2048 before it is routed to CPU and peripherals
  - First stage prescaler (A) can divide by a factor between 1 and 512
  - Prescaler B and C allow further division by a factor from 1 to 4 – If they are not used all the clocks will run at the same frequency as output of Prescaler A

xMega-A1: PLL

- Built in Phase Lock Loop (PLL) can be used to generate a higher frequency system clock
  - Multiplication factor from 1 to 31

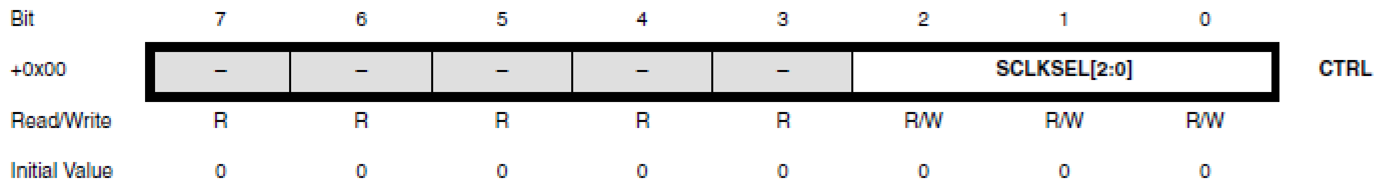
$$f_{OUT} = f_{IN} \cdot \text{PLL\_FAC}$$

- Input sources to PLL
  - 2 MHz internal oscillator
  - 32 MHz internal oscillator divide by 4
  - 0.4 MHz – 16 MHz crystal oscillator
  - External clock

- To enable PLL
  - Enable reference clock source
  - Set the multiplication factor and clock reference for PLL
  - Wait until clock reference source is stable
  - Enable PLL

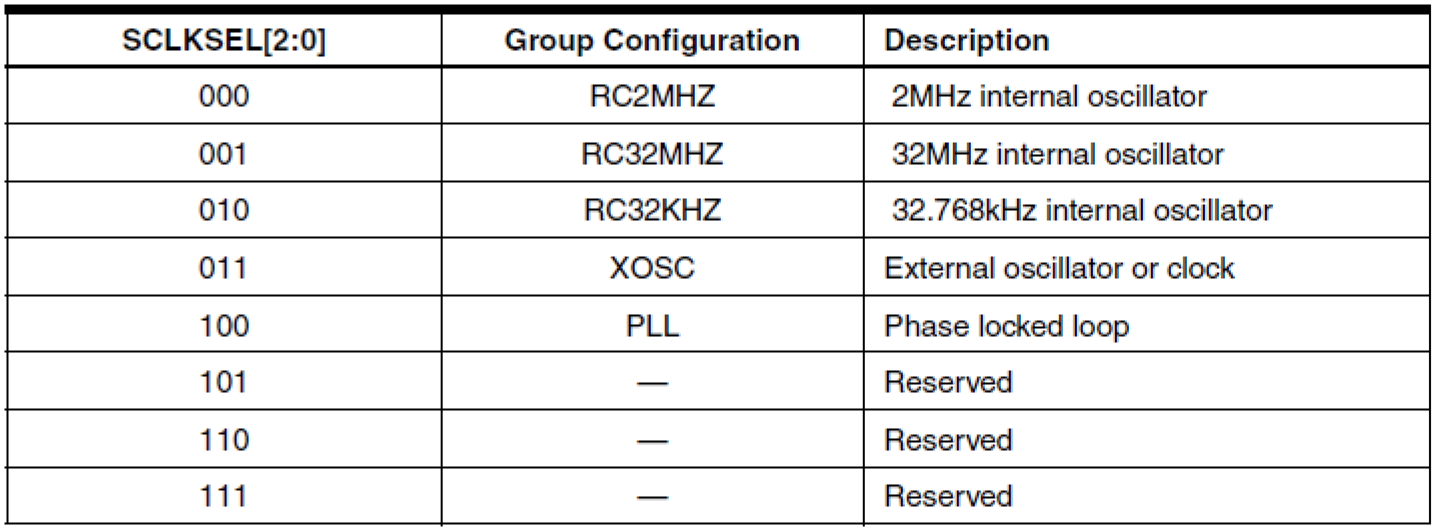
xMega-A1 Clocks: Associated Registers

- System Clock Control Register



Bit	7	6	5	4	3	2	1	0	
+0x00	–	–	–	–	–	SCLKSEL[2:0]			CTRL
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 7-1. System clock selection.



SCLKSEL[2:0]	Group Configuration	Description
000	RC2MHZ	2MHz internal oscillator
001	RC32MHZ	32MHz internal oscillator
010	RC32KHZ	32.768kHz internal oscillator
011	XOSC	External oscillator or clock
100	PLL	Phase locked loop
101	—	Reserved
110	—	Reserved
111	—	Reserved

xMega-A1 Clocks: Associated Registers

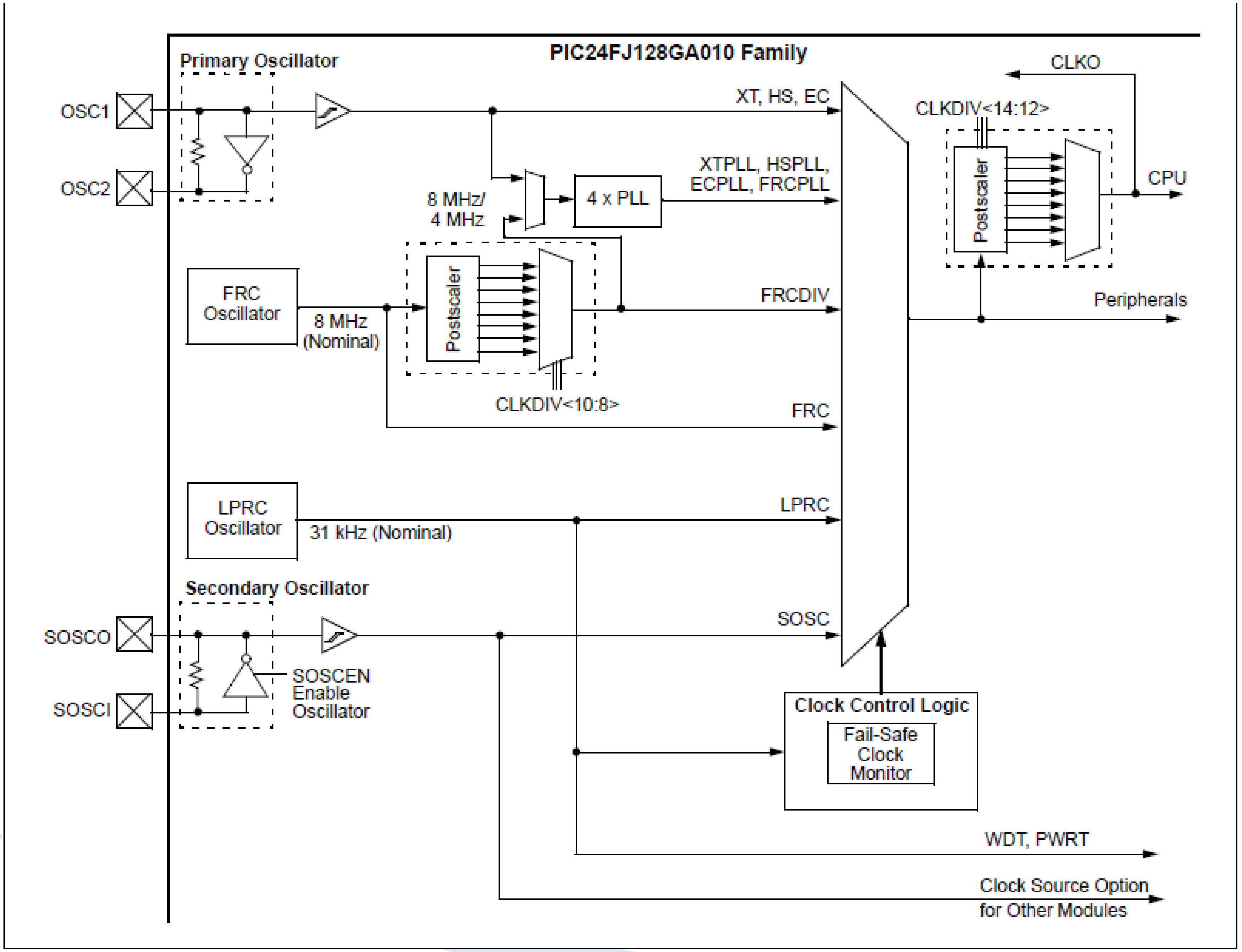
- System Clock Prescaler Register

Bit	7	6	5	4	3	2	1	0	
+0x01	–	PSADIV[4:0]					PSBCDIV		PSCTRL
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 7-2. Prescaler A division factor.

PSADIV[4:0]	Group Configuration	Description
00000	1	No division
00001	2	Divide by 2
00011	4	Divide by 4
00101	8	Divide by 8
00111	16	Divide by 16
01001	32	Divide by 32
01011	64	Divide by 64
01101	128	Divide by 128
01111	256	Divide by 256
10001	512	Divide by 512

PIC24F: Clock Selection



AVR: Fuses

- Similar to a config file, fuses are used to configure important system parameters at the time of programming
  - E.g. specifying the clock source/speed, reset sources
- Fuses are not erased when AVR memory is erased
  - It is not required to reprogram the fuses every time AVR is programmed
  - It can cause problems if incorrect settings are selected
- If you want to change some of the fuse bits
  - Read the current fuse configuration to get the configuration of the remaining bits
  - Combine it with new settings to get the byte information for the fuse bits
  - Configure the byte information in fuse bytes while programming the chip
- Do not change fuses when operating from boot loader
- Fuse values are latched when the device enters the programming mode and changes will have no effect until the part leaves the programming mode

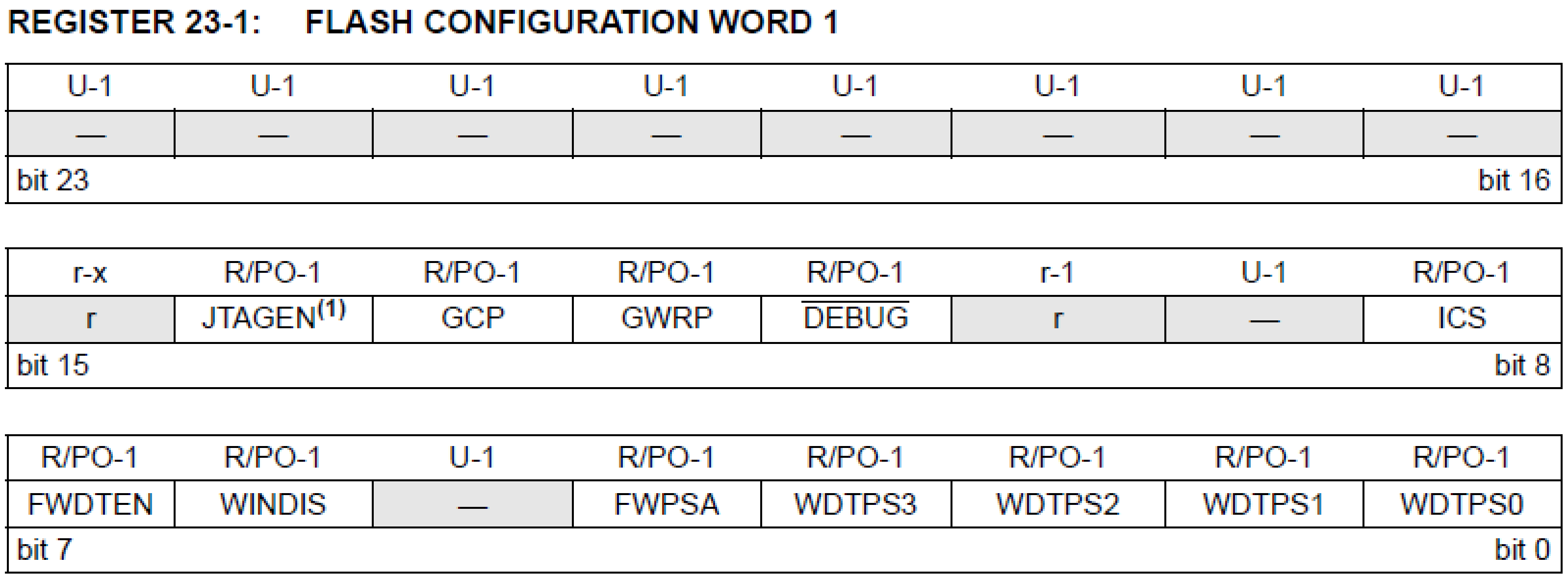
AT xMega-A1 Fuse Registers Examples

Bit	7	6	5	4	3	2	1	0	
+0x02	–	BOOTRST	TOSCSEL	–	–	–	BODPD[1:0]		FUSEBYTE2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

Bit	7	6	5	4	3	2	1	0	
+0x04	–	–	–	RSTDISBL	STARTUPTIME[1:0]		WDLOCK	JTAGEN	FUSEBYTE4
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

PIC24F: Configuration Memory

- Similar to Fuse Bytes in AVR



REGISTER 23-1: FLASH CONFIGURATION WORD 1

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	R/PO-1	R/PO-1	R/PO-1	R/PO-1	r-1	U-1	R/PO-1
r	JTAGEN[1]	GCP	GWRP	$\overline{\text{DEBUG}}$	r	—	ICS
bit 15							bit 8

R/PO-1	R/PO-1	U-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1
FWDTEN	WINDIS	—	FWPSA	WDTPS3	WDTPS2	WDTPS1	WDTPS0
bit 7							bit 0

xMega: Power Management

- Sleep modes and clock gating enable application to shut down unused modules in MCU, thereby saving power

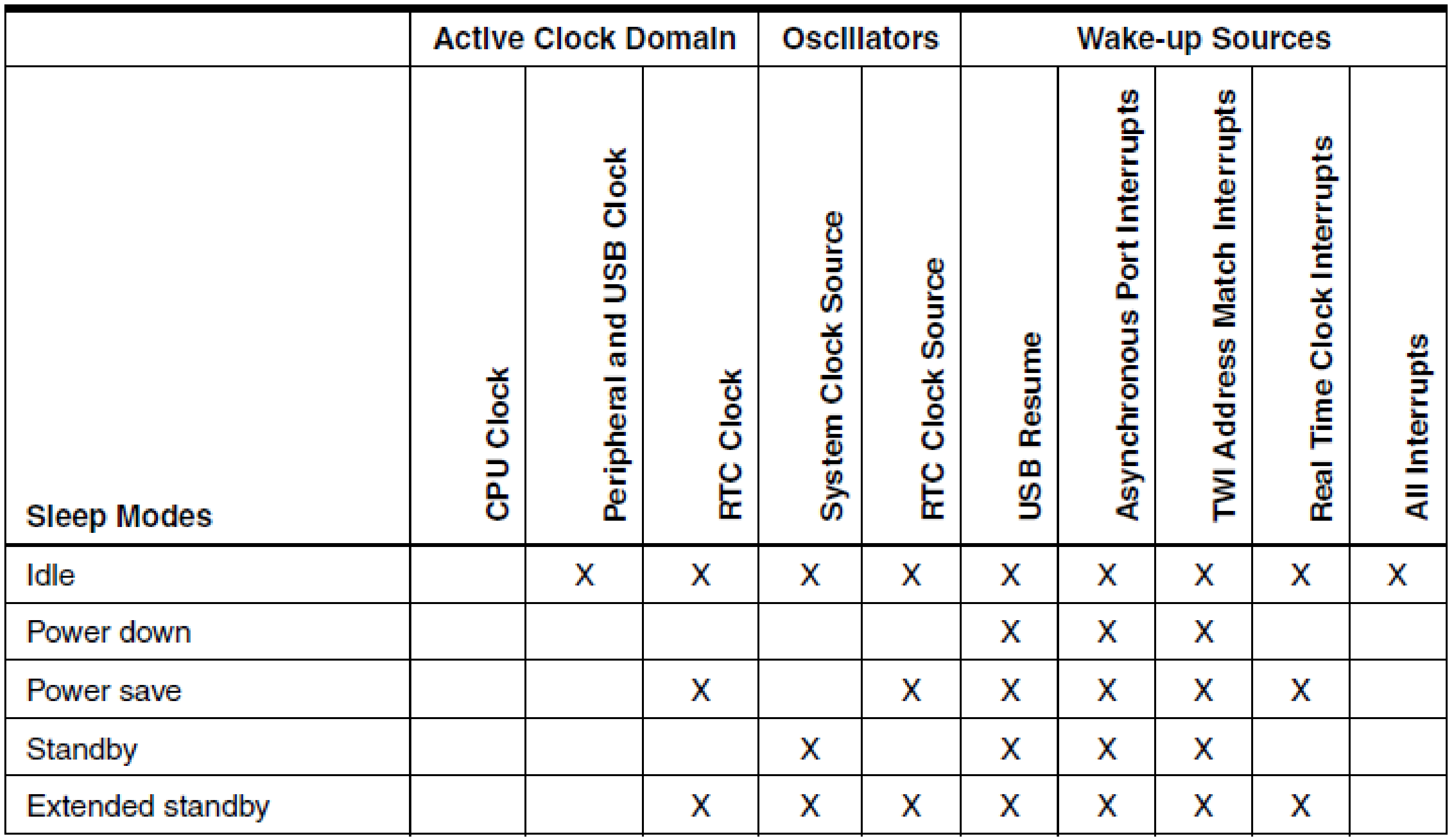
- All sleep modes can be entered from active mode

- Once the device is in sleep, program execution is stopped
  - Interrupts or reset can be used to wake the device again

- In addition to sleep mode, power reduction register provides a way to stop the clock to individual peripherals from software
  - Allow much more fine-tuned power management than is possible with sleep modes only

xMega: Sleep Modes

- Five different sleep modes - Idle, Power down, Power save, Standby, Extended Standby
  - Dedicated SLEEP instruction to enter the sleep mode
  - Different interrupts in each mode to wake up the device: After wake up, device executes ISR before continuing normal operation



	Active Clock Domain			Oscillators		Wake-up Sources				
Sleep Modes	CPU Clock	Peripheral and USB Clock	RTC Clock	System Clock Source	RTC Clock Source	USB Resume	Asynchronous Port Interrupts	TWI Address Match Interrupts	Real Time Clock Interrupts	All Interrupts
Idle		X	X	X	X	X	X	X	X	X
Power down						X	X	X		
Power save			X		X	X	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

xMega: Sleep Mode Registers

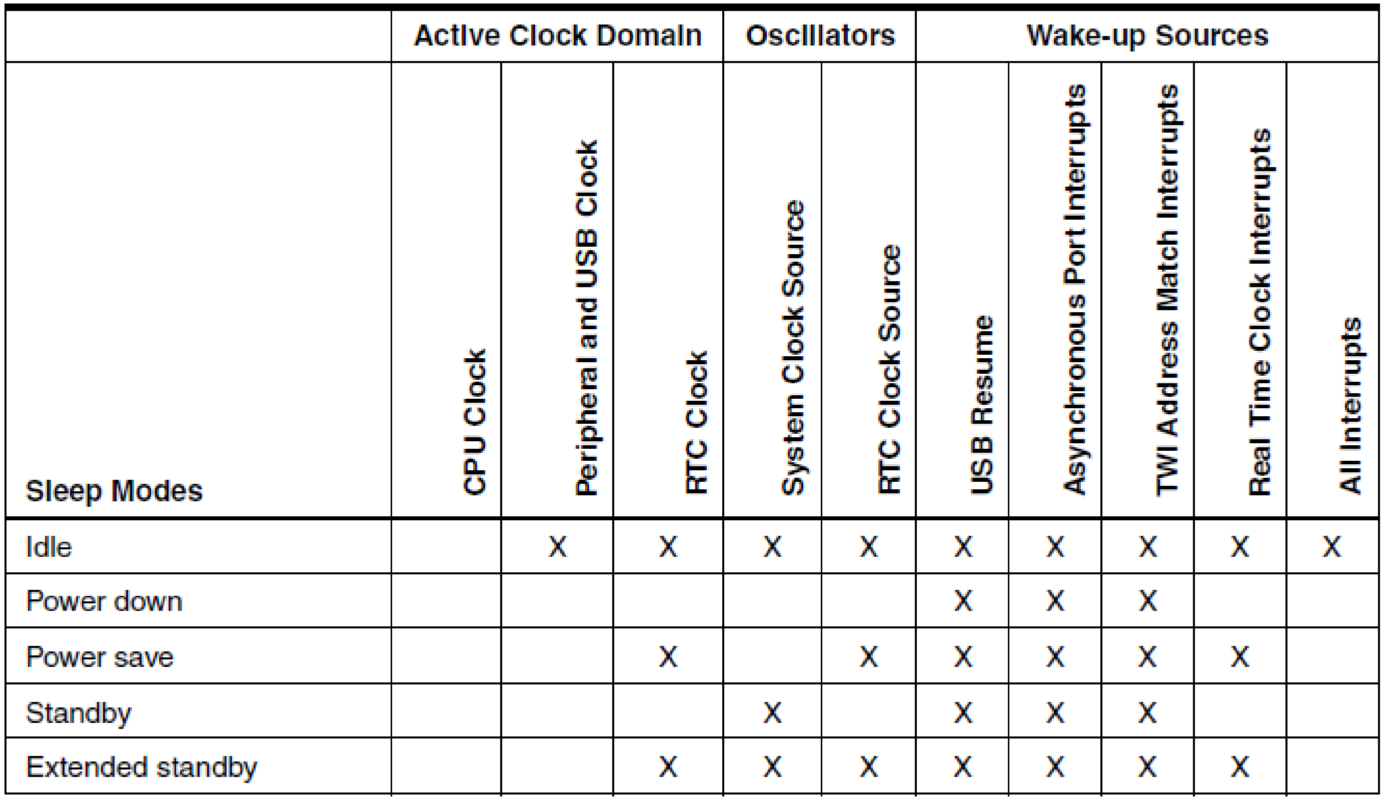
- Sleep Control Register (CTRL) contains control bits for power management

Bit	7	6	5	4	3	2	1	0	
+0x00	–	–	–	–	SMODE[2:0]			SEN	CTRL
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- To enter any of the mode, SEN bit in CTRL must be written to logic 1 and SLEEP instruction must be executed

SMODE[2:0]	Group Configuration	Description
000	IDLE	Idle mode
001	—	Reserved
010	PDOWN	Power-down mode
011	PSAVE	Power-save mode
100	—	Reserved
101	—	Reserved
110	STDBY	Standby mode
111	ESTDBY	Extended standby mode

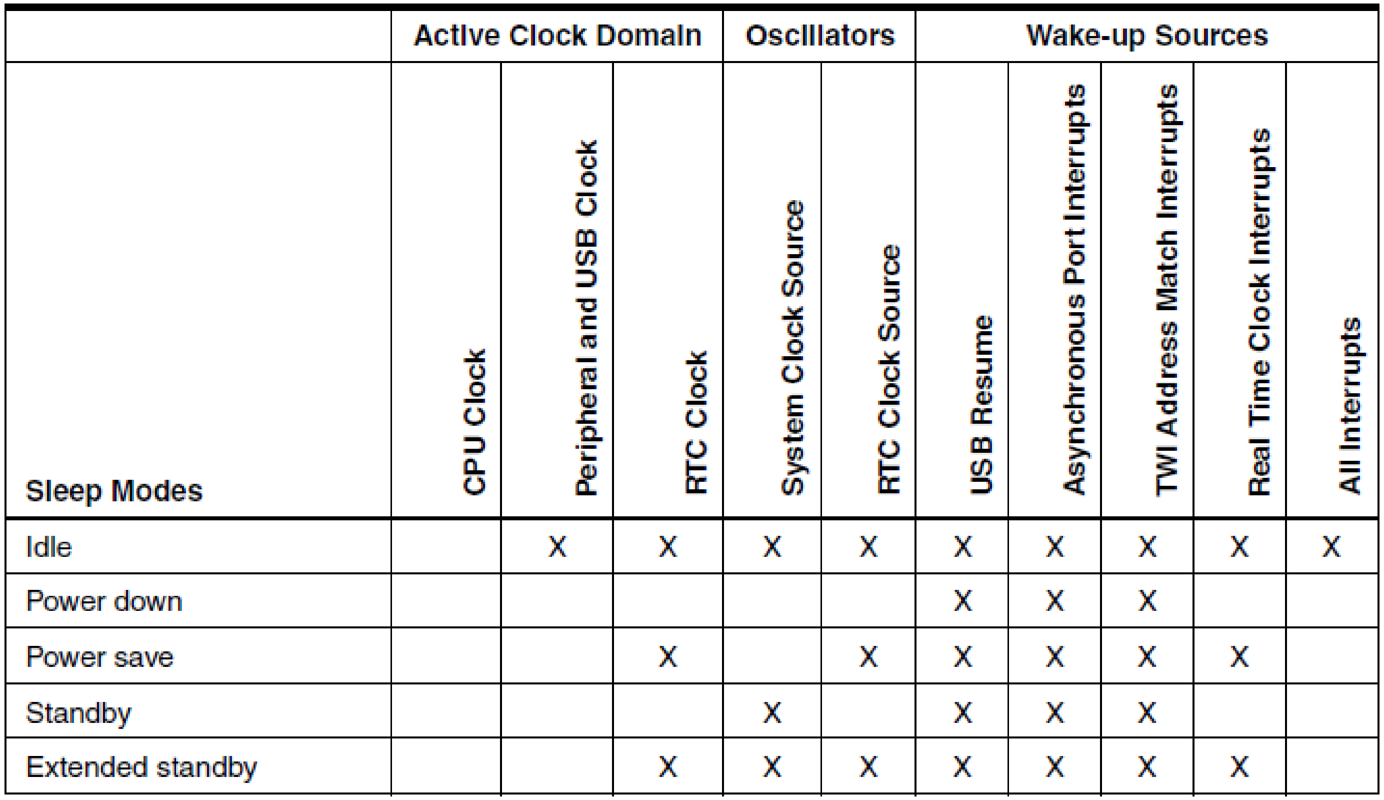
Sleep Modes - IDLE



	Active Clock Domain			Oscillators		Wake-up Sources				
Sleep Modes	CPU Clock	Peripheral and USB Clock	RTC Clock	System Clock Source	RTC Clock Source	USB Resume	Asynchronous Port Interrupts	TWI Address Match Interrupts	Real Time Clock Interrupts	All Interrupts
Idle		X	X	X	X	X	X	X	X	X
Power down						X	X	X		
Power save			X		X	X	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

- IDLE Mode (SM2..0 = 000) - Stops the CPU clock but all peripherals are kept running
  - Which modules will be stopped?
  - Interrupt requests from all enabled interrupts will cause wake up

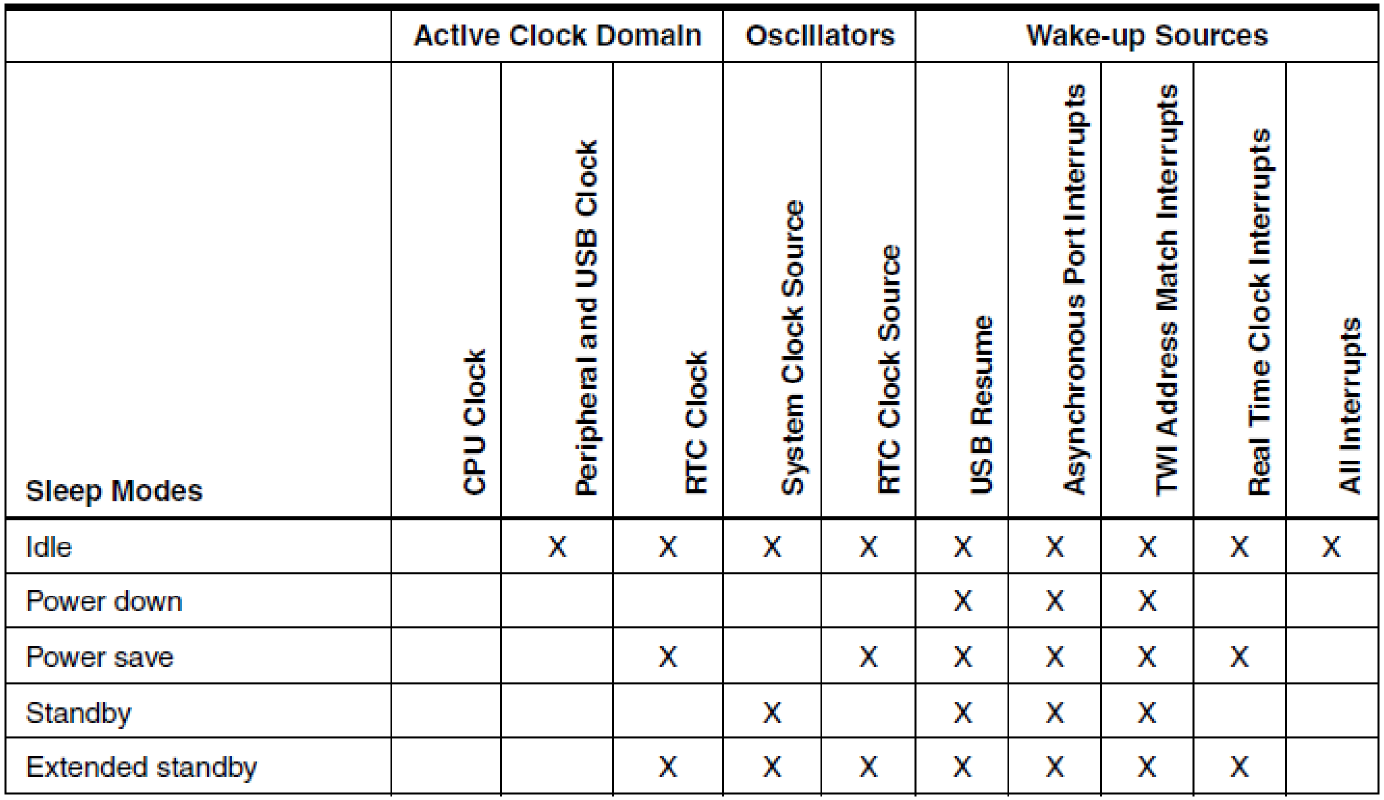
Sleep Modes - Power-down Mode



	Active Clock Domain			Oscillators		Wake-up Sources				
Sleep Modes	CPU Clock	Peripheral and USB Clock	RTC Clock	System Clock Source	RTC Clock Source	USB Resume	Asynchronous Port Interrupts	TWI Address Match Interrupts	Real Time Clock Interrupts	All Interrupts
Idle		X	X	X	X	X	X	X	X	X
Power down						X	X	X		
Power save			X		X	X	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

- Power-down Mode (SM2..0 = 010): Halts all generated clocks, allowing operations of asynchronous modules only
- TWI Interrupt and asynchronous port interrupts cause wake up

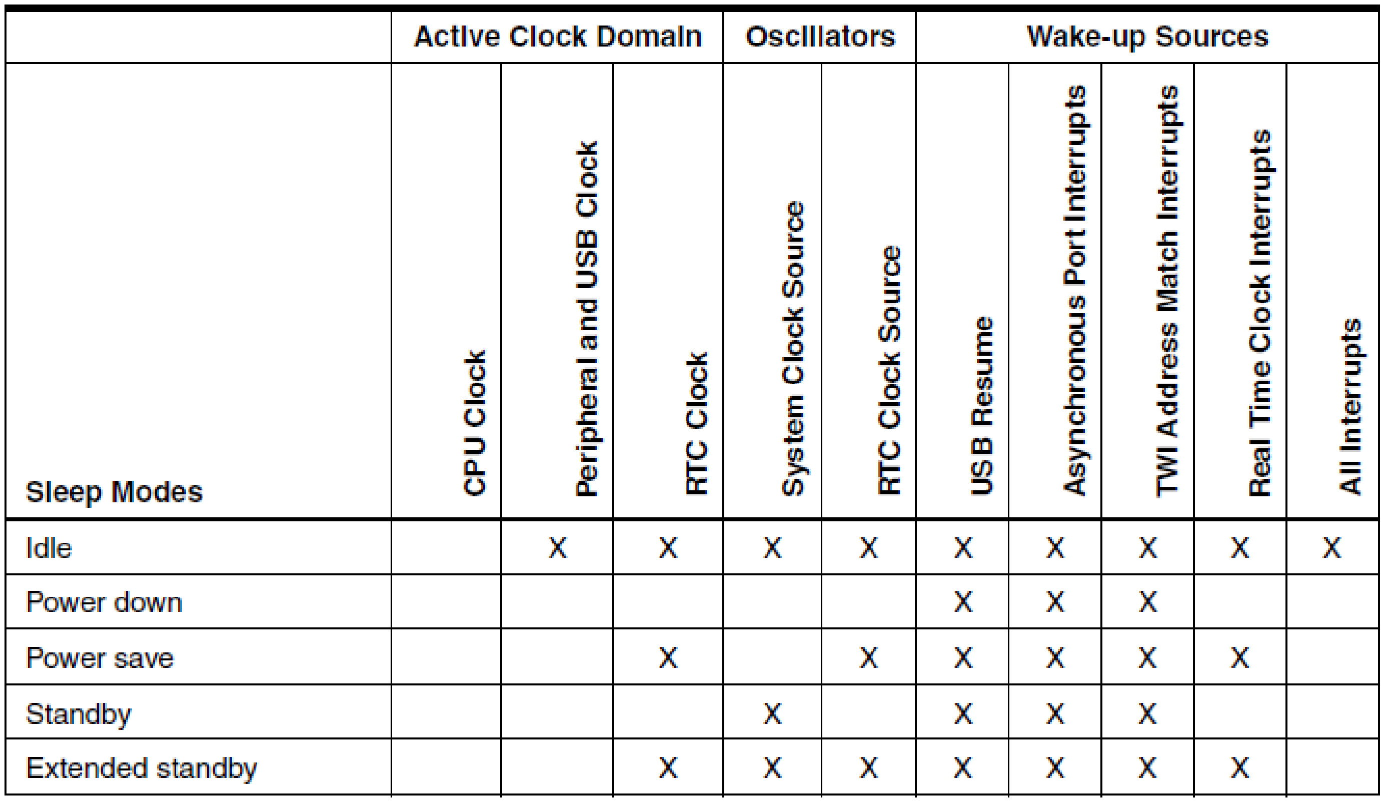
Sleep Modes - Power-save Mode



	Active Clock Domain			Oscillators		Wake-up Sources				
Sleep Modes	CPU Clock	Peripheral and USB Clock	RTC Clock	System Clock Source	RTC Clock Source	USB Resume	Asynchronous Port Interrupts	TWI Address Match Interrupts	Real Time Clock Interrupts	All Interrupts
Idle		X	X	X	X	X	X	X	X	X
Power down						X	X	X		
Power save			X		X	X	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

- Power-save Mode (SM2..0 = 011) - Similar to Power-down mode
  - If RTC is enabled, it will keep running during sleep
- Device can also wake up from RTC interrupts

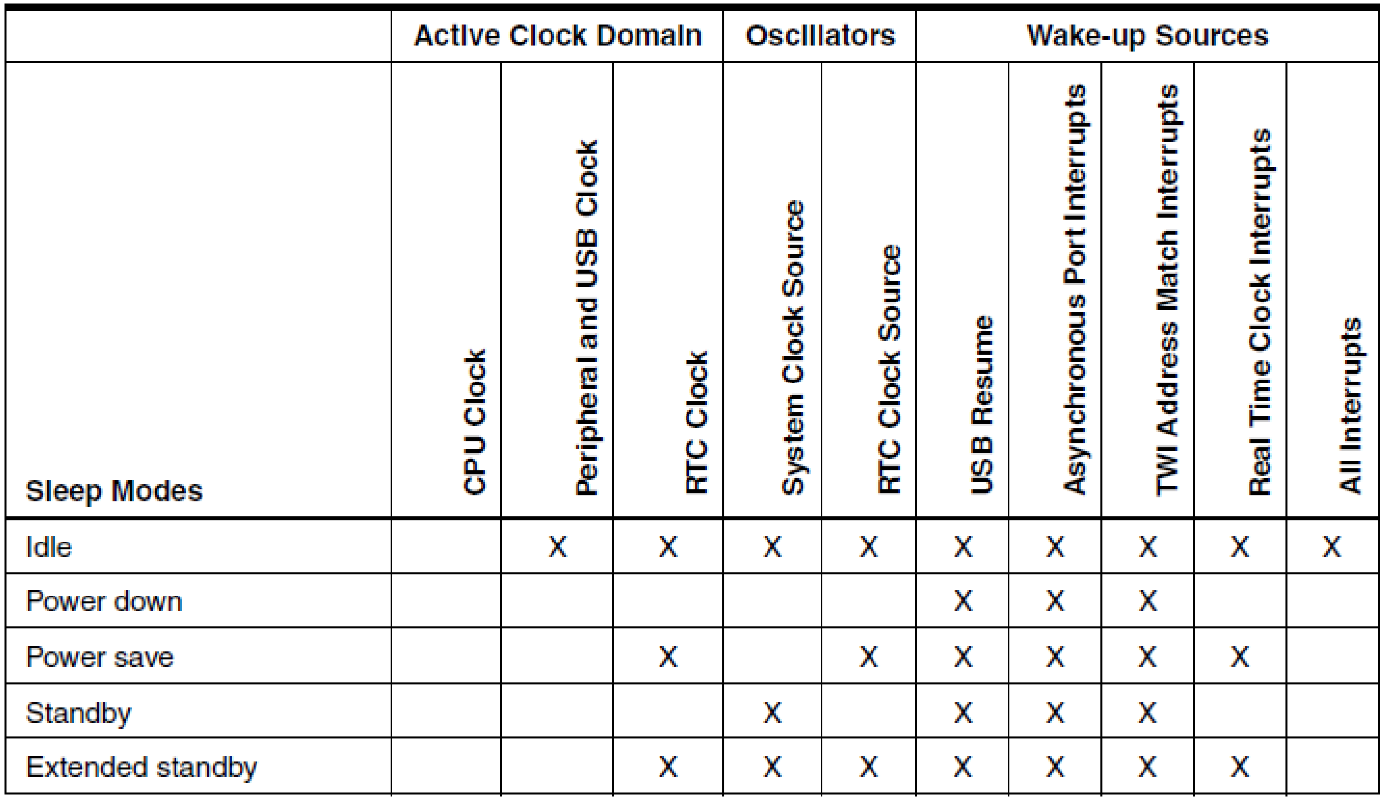
Sleep Modes - Standby Mode



	Active Clock Domain			Oscillators		Wake-up Sources				
Sleep Modes	CPU Clock	Peripheral and USB Clock	RTC Clock	System Clock Source	RTC Clock Source	USB Resume	Asynchronous Port Interrupts	TWI Address Match Interrupts	Real Time Clock Interrupts	All Interrupts
Idle		X	X	X	X	X	X	X	X	X
Power down						X	X	X		
Power save			X		X	X	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

- Standby Mode (SM2..0 = 110) - Similar to Power-down mode
  - Exception – System clock sources kept running
- Reduces wake up time when external crystals are used

Sleep Modes - Extended Standby Mode



	Active Clock Domain			Oscillators		Wake-up Sources				
Sleep Modes	CPU Clock	Peripheral and USB Clock	RTC Clock	System Clock Source	RTC Clock Source	USB Resume	Asynchronous Port Interrupts	TWI Address Match Interrupts	Real Time Clock Interrupts	All Interrupts
Idle		X	X	X	X	X	X	X	X	X
Power down						X	X	X		
Power save			X		X	X	X	X	X	
Standby				X		X	X	X		
Extended standby			X	X	X	X	X	X	X	

- Extended Standby Mode (SM2..0 = 111) - Similar to Power-save mode
  - Exception – System clock sources are kept running
- Reduces wake-up time when external clock sources are used

Minimizing Power Consumption

- Some of the functions if enabled, remain so in all/most sleep modes. These should be explicitly disabled before entering any sleep mode
  - Analog to Digital Converter
  - Analog Comparator - When entering Idle mode, Analog Comparator should be disabled, if not used
  - Brown-out Detector
  - Watchdog Timer

- Port pins: When entering a sleep mode, all port pins should be configured to consume minimum power
  - In sleep modes where clk_{PER} is stopped, input buffers will be disabled ensuring no power consumed by input logic
  - Input buffer enabled when input logic is needed for detecting wake-up conditions

xMega: Power Reduction Registers

- Power Reduction Registers (PR) provide a method to stop the clock to individual peripherals to reduce power consumption
  - Current state of peripheral is frozen and I/O registers can not be read or written
  - Resources in use when clock is stopped will remain occupied
    - Peripheral should be disabled before stopping the clock
  - Waking up a module (by clearing the bit in PR) puts the module in same state as before shutdown

- Can be used in idle/active mode to significantly reduce the overall power consumption
  - In all other sleep modes peripheral clock is already stopped

xMega: Power Reduction Registers

- PRGEN: General power reduction register

Bit	7	6	5	4	3	2	1	0	
+0x00	–	USB	–	AES	EBI	RTC	EVSYS	DMA	PRGEN
Read/Write	R	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- PRPA/B: Power reduction Port A/B register

Bit	7	6	5	4	3	2	1	0	
+0x01/+0x02	–	–	–	–	–	DAC	ADC	AC	PRPA/B
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Note: Disabling of analog modules stops the clock to the analog blocks themselves and not only the interfaces.

- PRPC/D/E/F: Power reduction Port C/D/E/F register

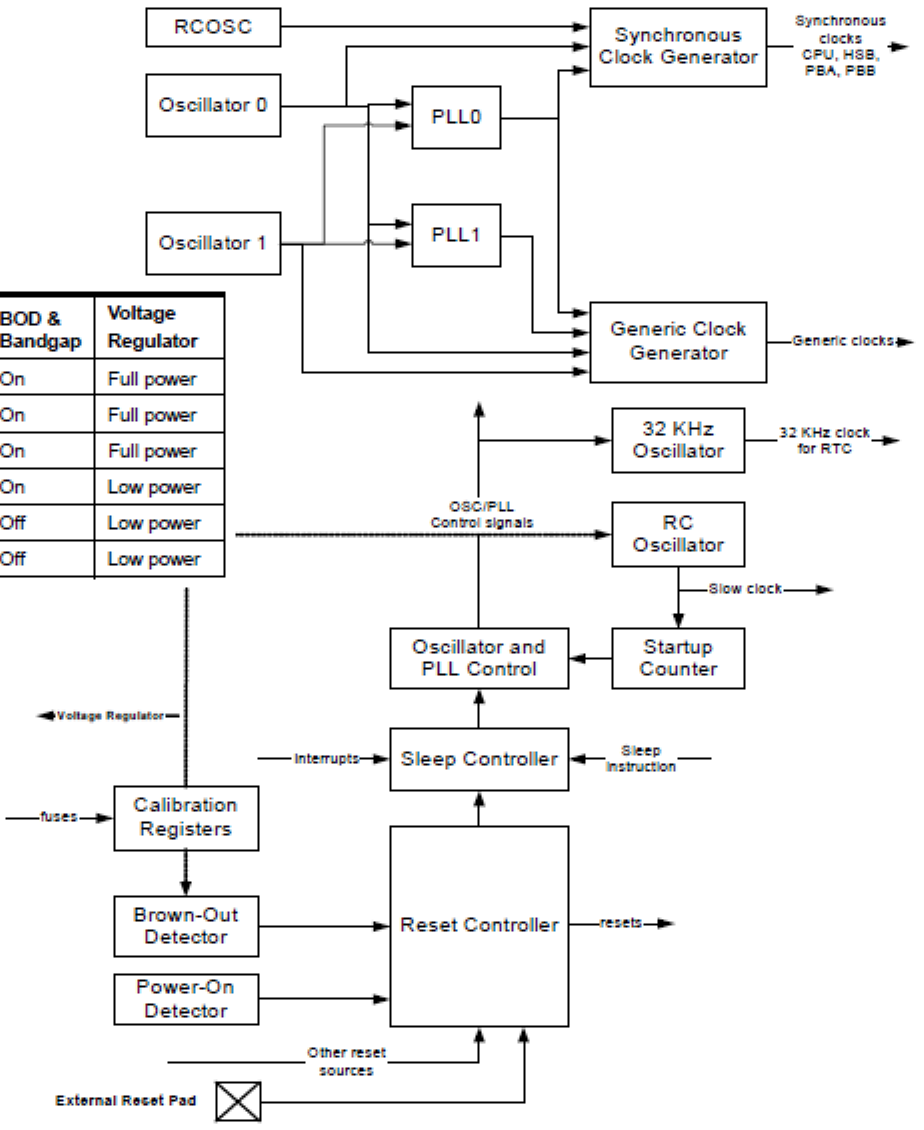
Bit	7	6	5	4	3	2	1	0	
+0x03/+0x04/ +0x05/+0x06	–	TWI	USART1	USART0	SPI	HIRES	TC1	TC0	PRPC/D/E/F
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PIC24F: Power Management

- PWRSAV command for sleep mode
- Two different modes defined – Sleep and Idle
- Reset or interrupts wake up the device
- Selective peripheral module control using one of the control registers

AVR32: Clocks and Resets

Index	Sleep Mode	CPU	HSB	PBA,B GCLK	Osc0,1 PLL0,1	Osc32	RCOsc	BOD & Bandgap	Voltage Regulator
0	**Idle**	Stop	Run	Run	Run	Run	Run	On	Full power
1	**Frozen**	Stop	Stop	Run	Run	Run	Run	On	Full power
2	**Standby**	Stop	Stop	Stop	Run	Run	Run	On	Full power
3	**Stop**	Stop	Stop	Stop	Stop	Run	Run	On	Low power
4	**DeepStop**	Stop	Stop	Stop	Stop	Run	Run	Off	Low power
5	**Static**	Stop	Stop	Stop	Stop	Stop	Stop	Off	Low power



xMega - Interrupts

- Interrupts signal change of state in peripherals

- Peripherals can have one or more interrupt, each individually configured and enabled

- Programmable Multilevel Interrupt Controller (PMIC) controls the handling and prioritizing of interrupt requests

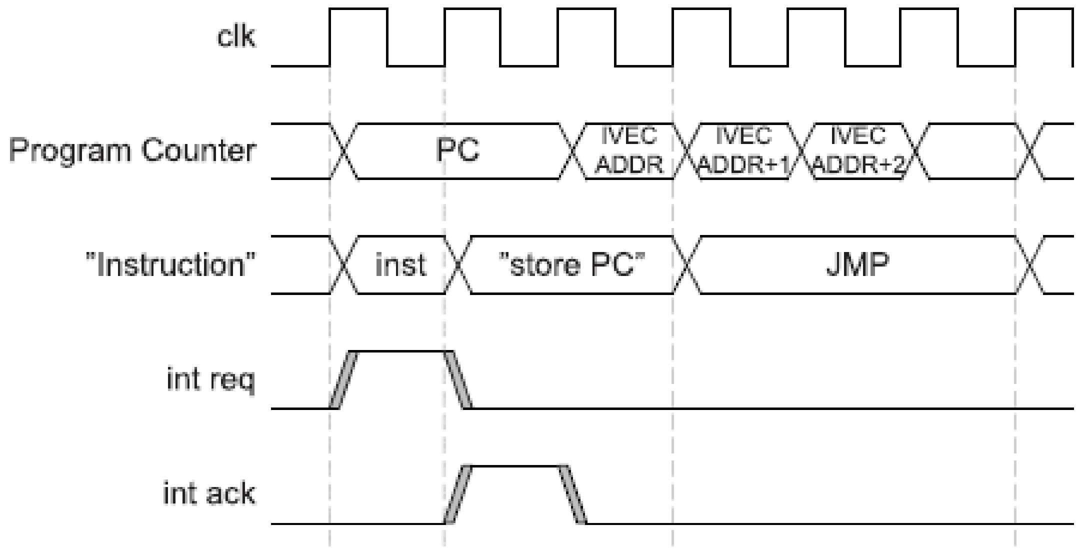
- All peripherals can select between three different priority levels – low, medium, high

- Within each level, priority is decided based on interrupt vector address (lower address corresponds to higher priority)

- All interrupts are assigned individual enable bits, together with a global interrupt enable bit which should be written logic one together to enable the interrupt

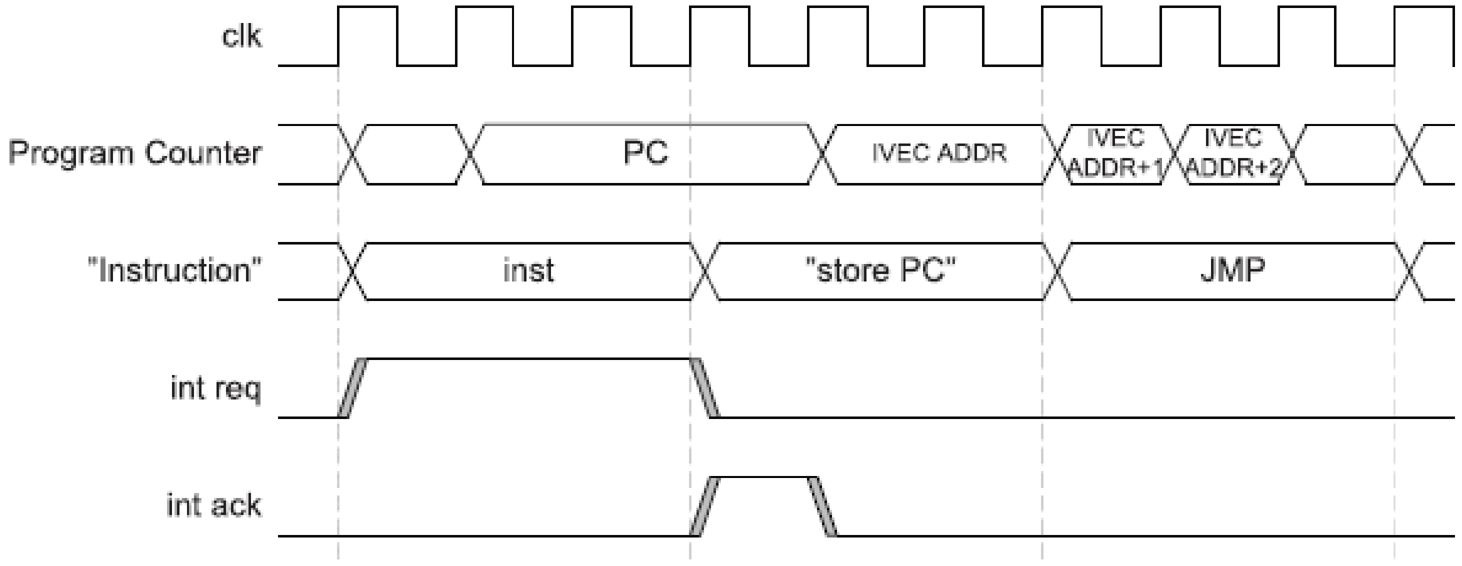
xMega – Interrupt Response Time

- Interrupt execution response is minimum 3 clock cycles
  - One cycle to complete the ongoing instruction and 2 clock cycles to store the program counter
  - After three clock cycles program vector address for the actual interrupt handling routine is executed
  - The vector is normally a jump to interrupt routine which takes three clock cycles
  - Return from interrupt takes 4-5 clock cycles – what happens then?



xMega – Interrupt Response Time

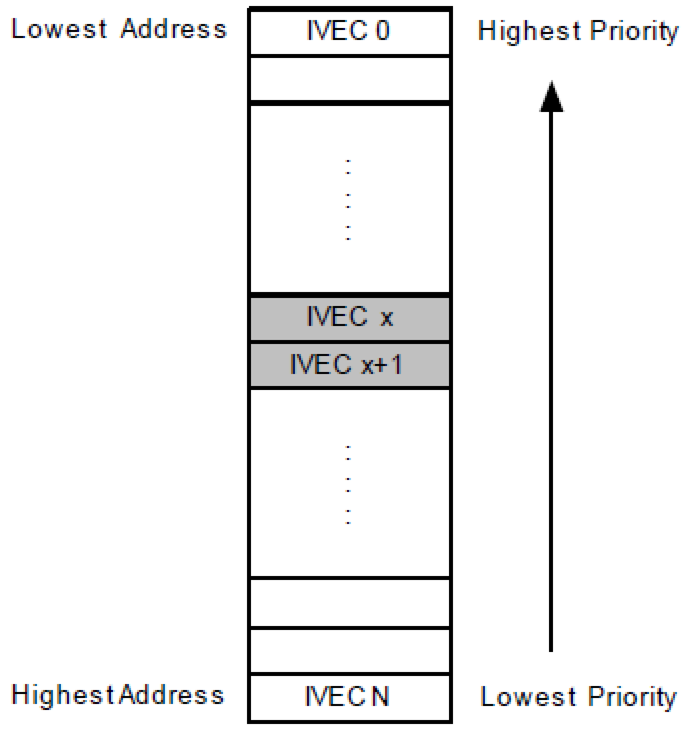
- If interrupt occurs during execution of multicycle instruction, this instruction is completed before interrupt is serviced



xMega – Interrupt Priority

- Low level interrupts can be organized in a static or dynamic (round-robin) priority scheme
- High and medium level interrupts will always have static priority

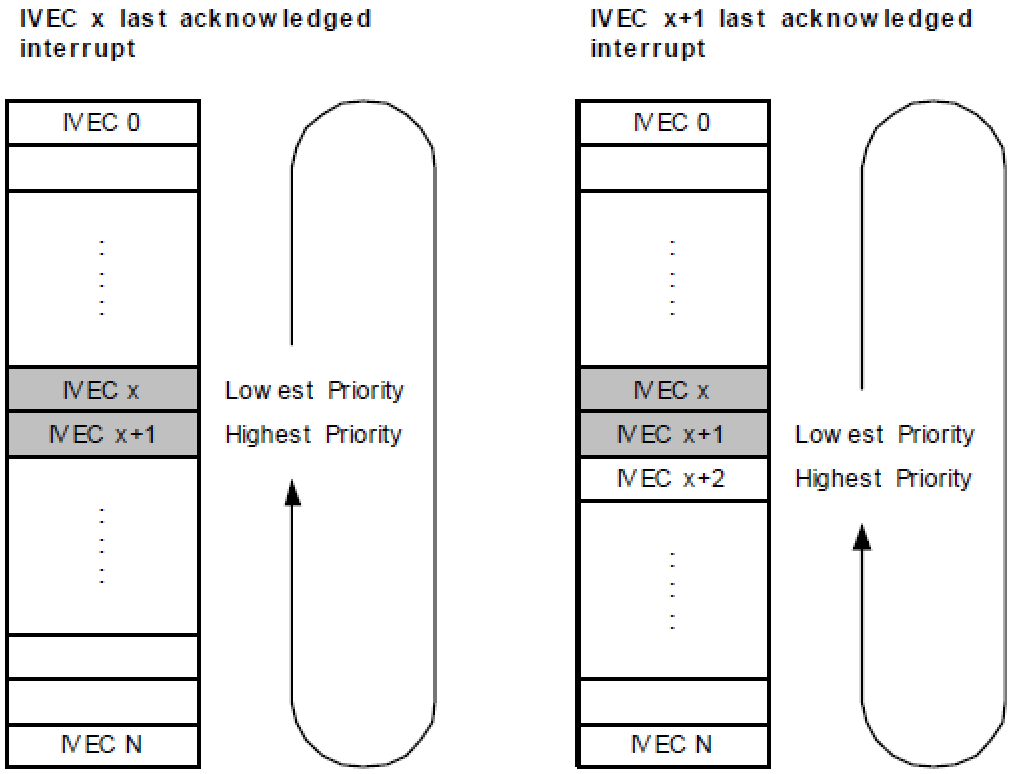
Static Priority



xMega – Interrupt Priority

- Round robin scheduling to avoid possible starvation problem for low-level interrupts
- Interrupt vector address for the last acknowledged low-level interrupt will have the lowest priority the next time one or more interrupts from the low level is requested

Round Robin Scheduling



xMega Interrupts: Associated Registers

- Status: Flag is set when the corresponding interrupt is executing; cleared when returning from the interrupt handler

Bit	7	6	5	4	3	2	1	0	
+0x00	NMIEX	–	–	–	–	HILVLEX	MEDLVLEX	LOLVLEX	STATUS
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- Control: To enable the corresponding interrupts

Bit	7	6	5	4	3	2	1	0	
+0x02	RREN	IVSEL	–	–	–	HILVLEN	MEDLVLEN	LOLVLEN	CTRL
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Interrupt Vector Select (IVSEL)
  - 0 implies IVs are placed at the start of Flash memory
  - 1 implies IVs are places at the start of Boot Loader Section

Pic24F: Interrupt Controller

- 7 user selectable priority levels
- Fixed priority within the specified user level
- Lower address in Interrupt Vector Table implies higher priority
- Interrupt control and status registers

What did you learn today?

- On a small sheet of paper, give me a short (2-3 sentences) description of what you learned today
  - You can use it to give any broad comments on the class as well