NC STATE UNIVERSITY

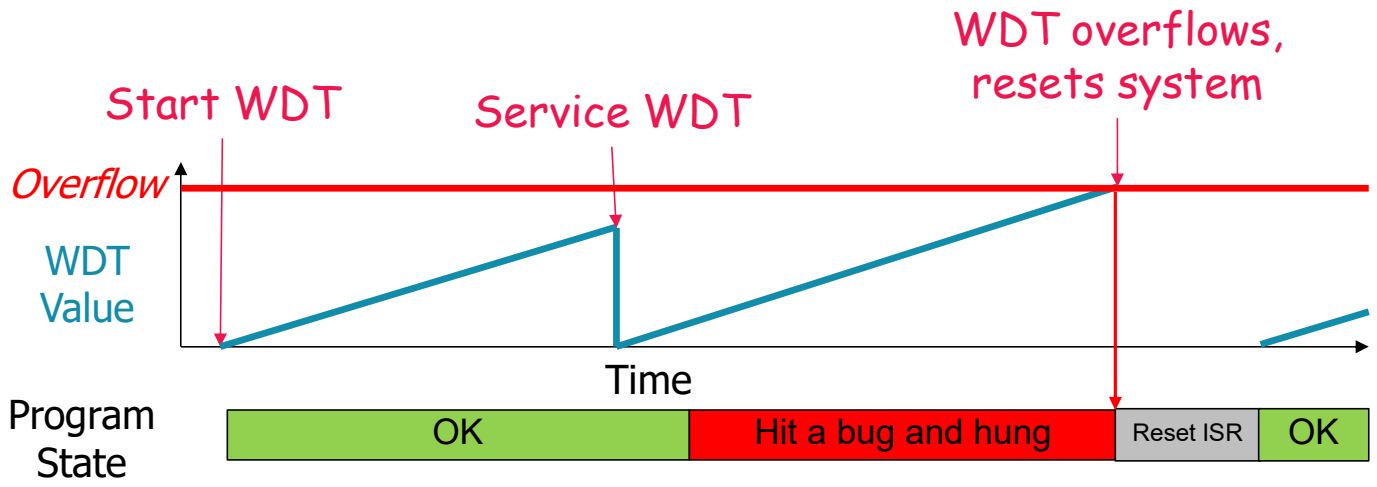
Peripherals for Robust Embedded Systems

NC STATE UNIVERSITY

WATCHDOG TIMER

NC STATE UNIVERSITY

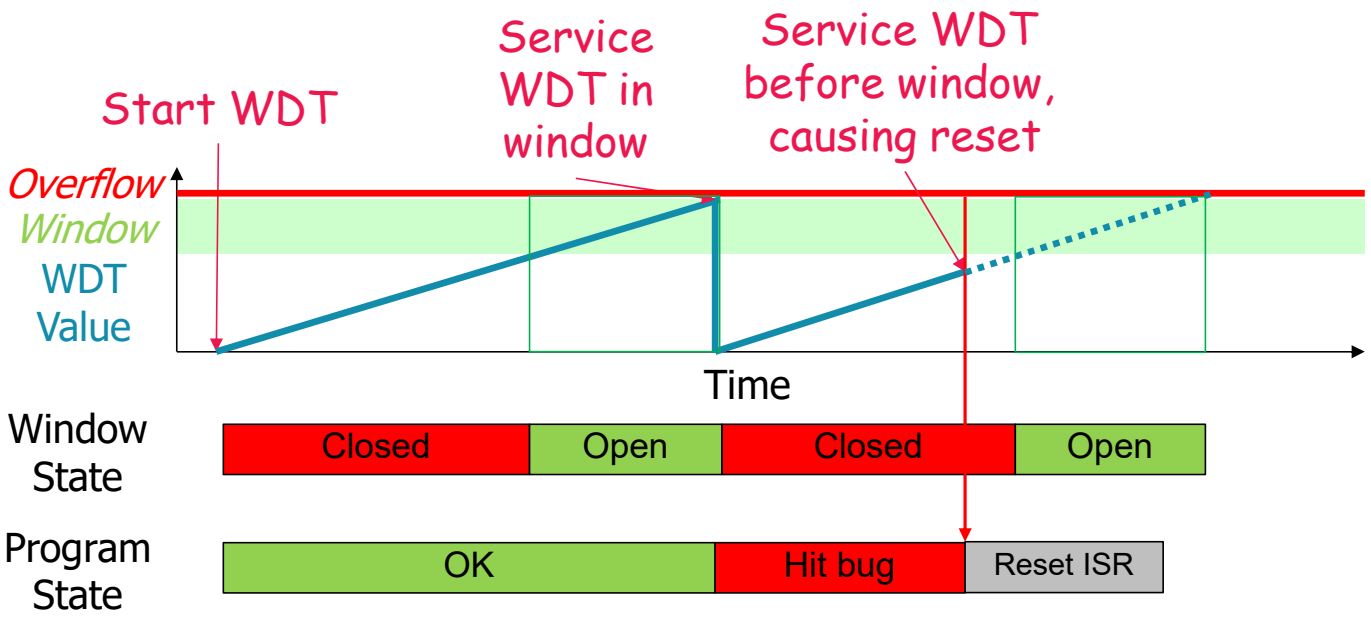
Watch Dog Timer (WDT) Concepts



- Goal: detect if program is not refreshing WDT correctly (≈ malfunctioning?)
- Mechanism:
  - Program periodically tells WDT that it is OK (services the WDT)
  - If timer ever overflows (times out), then program didn't say it was OK soon enough, so reset the MCU
- Typically used as a "last resort" mechanism for forcing system software back into known state

NC STATE UNIVERSITY

Windowed Watch Dog Timer Concepts



- Can only service WDT when WDT counter value is in a certain range
  - Any service attempt outside the window will cause the WDT to reset the system
- Windowed WDT improves reliability
  - ***Almost*** eliminates likelihood of program crash which still services the WDT

NC STATE UNIVERSITY

Kinetis KL25Z WDT (COP)

- WDT is called COP (Computer Operating Properly) by NXP
  - Details in KL25 Subfamily Reference Manual, Chapter 3 (Chip Configuration)
- Registers
  - COPCTRL: COP Control Register (COPC in CMSIS)
  - SRVCOP: Service COP register
- Initializing the COP
  - Write to COPCTRL
    - Write-protected: Only first write to COPCTRL after a reset is recognized, subsequent writes are ignored
  - Can disable COP by clearing COPCTRL[COPT]
- Startup Code Disables WDT by default
  - SystemInit() function in system_MKL25Z4.c disables the COP WDT by default because DISABLE_WDOG is non-zero (details later)

NC STATE UNIVERSITY

COP Configuration

Control Bits (COPCTRL)		Clock Source	COP Window Opens	COP Overflow Count
COPCLKS	COPT			
n/a	00	n/a	n/a	COP is disabled
0	01	1 kHz	n/a	2^5 cycles, 32 ms
0	10	1 kHz	n/a	2^8 cycles, 256 ms
0	11	1 kHz	n/a	2^{10} cycles, 1024 ms
1	01	Bus	6144 cycles	2^{13} cycles
1	10	Bus	49152 cycles	2^{16} cycles
1	11	Bus	196608 cycles	2^{18} cycles

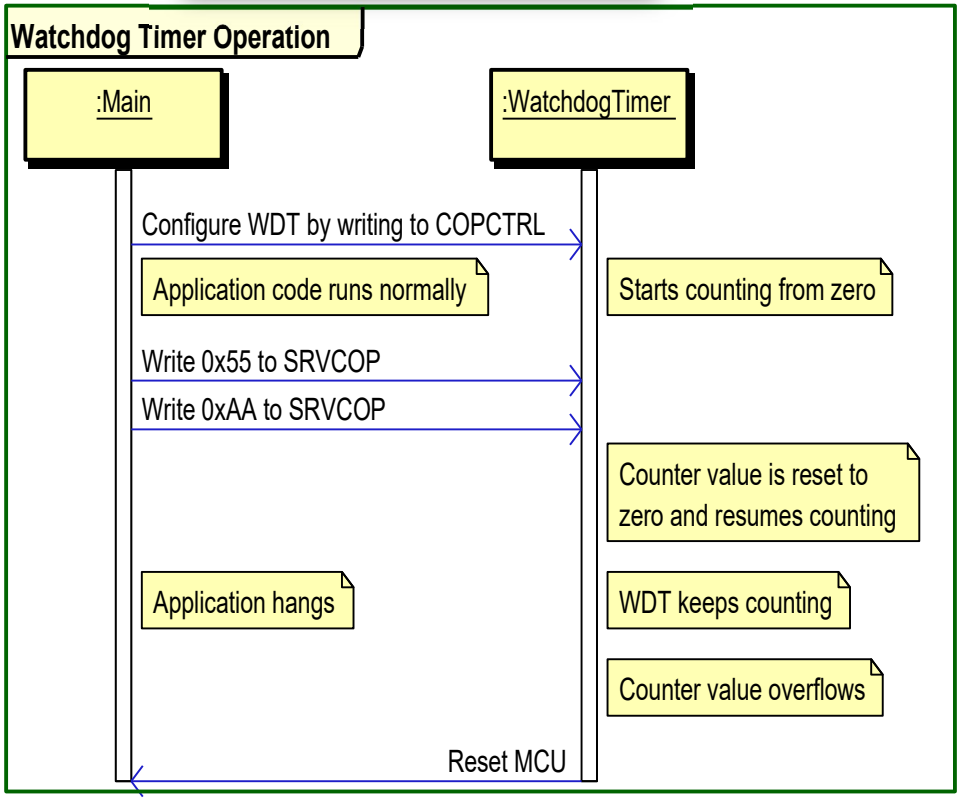
Default setting

- Select clock source with COPCTRL[COPCLKS]
  - 1 kHz or bus clock (e.g. 24 MHz)
- Set time-out divider with COPCTRL[COPT]
- Optional Windowed operation (when using bus clock) by setting COPCTRL[COPW]
  - Program must reset COP within last 25% of time-out period

WDT Service and Operation

- Servicing (resetting) the COP
  - Reset COP timer by writing 0x55 and then 0xAA to SRVCOP during timeout period
  - Writing wrong value causes COP to reset MCU immediately
- Reset actions
  - MCU is reset
  - WDOG bit in Reset Control Module (RCM) System Reset Status Register 0 (SRS0) is set to show watchdog caused reset

```
void Service_COP_WDT(void) {
  SIM->SRVCOP = 0x55;
  SIM->SRVCOP = 0xaa;
}
```



NC STATE UNIVERSITY

Determining Cause of Reset

Bit	7	6	5	4	3	2	1	0
Read	POR	PIN	WDOG	0	LOL	LOC	LVD	WAKEUP
Write								
Reset	1	0	0	0	0	0	1	0

- Reset Control Module (RCM) System Reset Status Register 0 (SRS0) bits
  - POR: power-on reset
  - PIN: external reset pin asserted
  - WDOG: COP watchdog timer timed out
  - LOL: Loss of lock in PLL (clock generation)
  - LOC: Loss of clock detected by MCG clock monitor
  - LVD: Low voltage detected
  - WAKEUP: System was asleep in low-leakage mode when a LLWU source was activated

SystemInit()

- SystemInit() function in system_MKL25Z4.c disables the COP WDT by default because DISABLE_WDOG is non-zero
- All subsequent writes to SIM->COPC are ignored by MCU
  - Including our own initialization code!
- To fix this, redefine DISABLE_WDOG to 0, so the compiler ignores lines 93-95 of the source code

system_MKL25Z4.c | COP_WDT.c | startup_MKL25Z

```
44
45  #define DISABLE_WDOG    1
```

system_MKL25Z4.c | COP_WDT.c | startup_MKL25Z4.s | i2c.c | main.c

```
91  void SystemInit (void) {
92  #if (DISABLE_WDOG)
93    /* Disable the WDOG module */
94    /* SIM_COPC: COPT=0,COPCLKS=0,COPW=0 */
95    SIM->COPC = (uint32_t)0x00u;
96  #endif /* (DISABLE_WDOG) */
```

```
void Init_COP_WDT(void) {
  SIM->COPC = SIM_COPC_COPT(3);
}
```

NC STATE UNIVERSITY

Demonstration Code – WDT Demo

- System flashes red LED on startup
- Loop:
  - Read accelerometer
  - Calculate board inclination
  - Is board flat (within 20^{o} of horizontal)?
    - Yes: Light green LED and service the watchdog
    - No: Light yellow LED and do NOT service the watchdog

```
while (1) {
  read_full_xyz();
  convert_xyz_to_roll_pitch();

  if ((fabs(roll) > MAX_ANGLE) | (fabs(pitch) > MAX_ANGLE)) {
    Control_RGB_LEDs(1, 1, 0); // Light yellow LED as warning
  } else {
    // service watchdog
    Control_RGB_LEDs(0, 1, 0); // Light green LED - OK!
    Service_COP_WDT();
  }
}
```

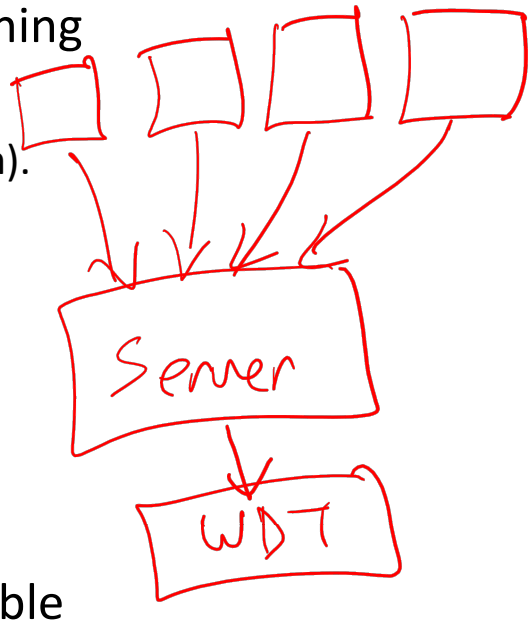
- Holding the board tilted for more than about 1 second will cause the WDT to trigger, resetting the CPU (shown by flashing red LED)

WDT Best Practices

- Don’t scatter WDT service commands throughout your code
  - There should just be one or a few such commands in the entire program
- Choose WDT period appropriately
  - Too long: System is out of control long enough to get into real trouble
  - Too short: You will need to reset WDT frequently in your code, complicating development with code writing and timing analysis overhead
- Frequent WDT resets? (Boot loop)
  - Consider logging abnormal causes of reset for later diagnosis
  - Determine how production code should respond to WDT resets (rare vs. frequent)
- WDT should be difficult to accidentally disable in software

Using a WDT in a Multithreaded Application

- Each periodic task updates a timestamp when it starts running
- Checker thread
  - Checks timestamp for each thread *i* to verify it ran recently (enough).
  - If all threads are ok, restart the WDT.
  - If not, allow WDT to expire.
- Does this detect *every possible problem?*
  - No, but it catches many and is far better than not using a WDT
- How to deal with non-periodic tasks?
  - Application-dependent!
- The checker could be integrated into the scheduler – possible programming project



Debugging with the WDT

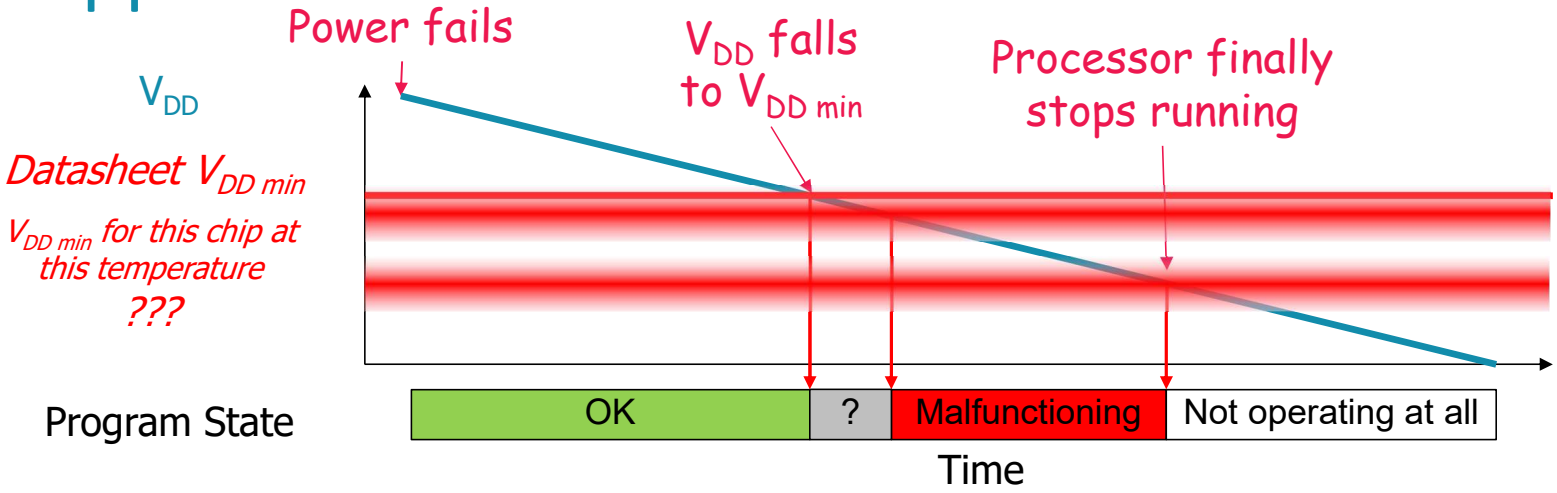
- How do you debug a system with a WDT?
  - Don't want MCU to reset as soon as the code hits a breakpoint!
  - Need to disable WDT somehow if it isn't done automatically

- KL25 COP behavior – handles WDT automatically
  - If using bus clock
    - COP doesn't increment if MCU is in debug mode or stop mode (including VLPS and LLS)
    - COP resumes counting upon exiting these modes
  - If using 1 kHz clock
    - COP cleared to zero as long as MCU is in debug or stop mode (including VLPS and LLS)
    - COP resumes counting (from zero) upon exiting these modes
  - Disabled when in VLLSx mode

NC STATE UNIVERSITY

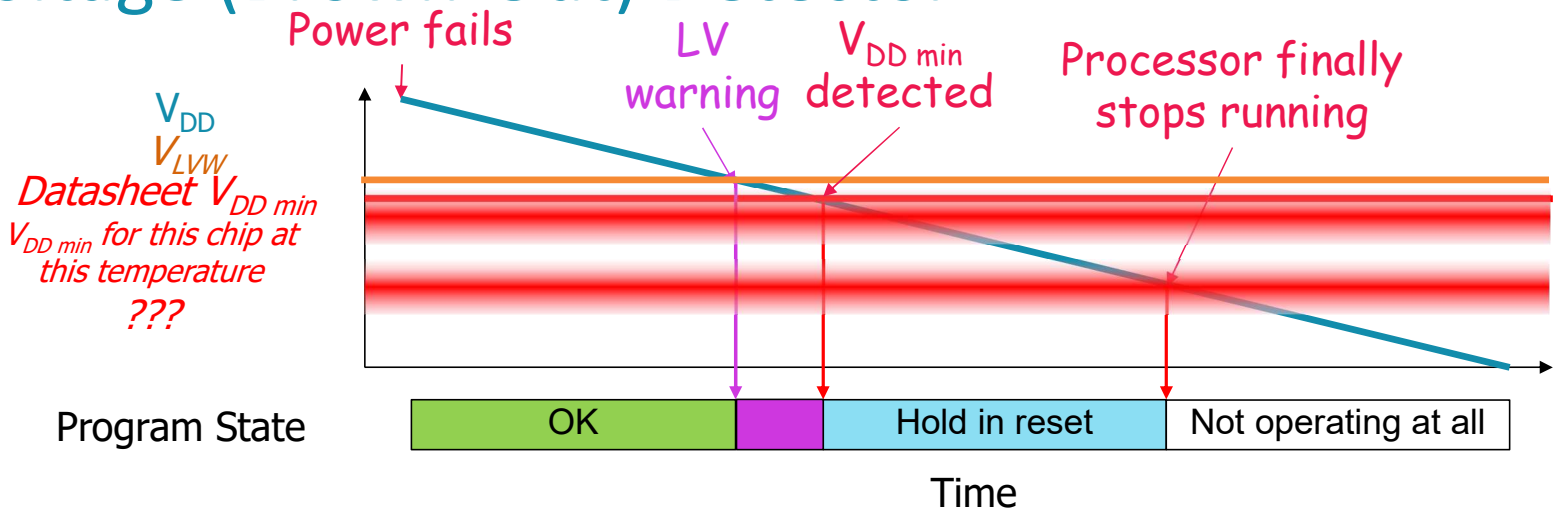
LOW VOLTAGE DETECTOR

What Happens When The Power Fails?



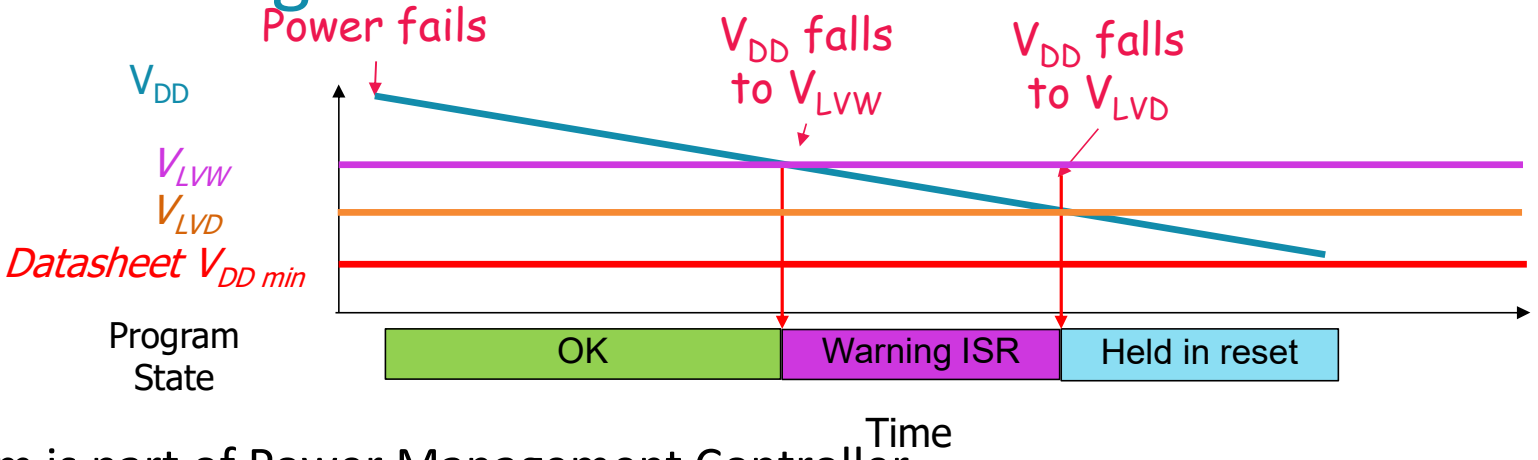
- With supply voltage V_{DD} too low, MCU is *not* guaranteed to work completely
  - Program may go crazy, overwrite configuration data, turn on outputs (motors, solenoids, brakes, spark plugs, ejection seats, lasers, ion cannons, etc.)
  - Want entire system to work properly, or else not at all. *Do no harm!*
- What is $V_{DD\ min}$?
  - Exact value of $V_{DD\ min}$ depends on particular chip, temperature, etc.
  - Datasheet specifies minimum value guaranteed by manufacturer to work

Low Voltage (Brown-Out) Detector



- Detect low voltage with hardware
- Warning at V_{LVW}
  - Set outputs to safe states, then spin in loop waiting for power to fail
- Detection at $V_{DD\ min}$
  - Holds processor in ***reset state*** while voltage is too low to operate correctly

KL25 Low Voltage Detector Overview



- LVD system is part of Power Management Controller
  - Details in KL25 Subfamily Reference Manual, Chapter 14
- Features
  - Configurable voltage levels
  - Two separate detectors: warning and detection
  - Can generate interrupt, reset MCU or set status flags
- Control Registers
  - PMC LVDSC1 (detection) and LVDSC2 (warning)

KL25 LVD Configuration

- Voltage trip point (threshold)
  - Detection: select one of two trip points (V_{LVD}) with LVDV
  - Warning: four trip points (V_{LVW}) with LVWV, based on LVDV (whether higher or lower detection trip point is used)
  - Trip points documented in KL25 subfamily datasheet
- Possible responses to LV warning or detection
  - Interrupt on LVD or LVW condition
    - Enable by setting LVDIE or LVWIE to 1
  - Reset on LVD condition
    - Enable by setting LVDRE bit to 1
    - LVD will reset MCU, set SRS0[LVD] bit in Reset Control Module (RCM) to show cause

LVDV	V_{LVD}	LVWV	V_{LVW}
00	1.6 V	00	1.8 V
		01	1.9 V
		10	2.0 V
		11	2.1 V
01	2.56 V	00	2.7 V
		01	2.8 V
		10	2.9 V
		11	3.0 V

NC STATE UNIVERSITY

LVD Interrupt and Status Flags

- Interrupt
  - PMC generates interrupt indicating LVD or LVW event
    - Vector #22, IRQ #6
    - IRQ number is 6, CMSIS defines name LVD_LVW_IRQn
  - Handled by LVD_LVW_IRQHandler

- Polling
  - Alternative to interrupts
  - Two flags indicate if voltage has fallen below trip point
    - LVDF: LV detect flag, set if $V<V_{LVD}$
    - LVWF: LV warning flag, set if $V<V_{LVW}$
    - Reset a flag by writing 1 to LVDACK or LVWACK

NC STATE UNIVERSITY

LVD Lab Exercise

- Operating behavior
  - Red LED flashes quickly on system start-up (when coming out of reset)
  - Green LED flashes slowly during normal operation ($V_{DD} > V_{LVW}$)
  - Red and green LED turn on if voltage falls below V_{LVW}, so flashing green becomes yellow/red
  - If voltage falls below V_{LVD}, system goes into reset, LEDs turn off
- Lab
  - Compare actual system behavior with expected behavior