HOCHSCHULE HEILBRONN
HEILBRONN UNIVERSITY
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

Chapter 2: Stellaris® family of microcontrollers

Stellaris® Cortex™-M3 - Microcontroller Family

Texas Instruments

Texas Instruments - University Program

Heilbronn University, Campus Künzelsau

Prof. Dr.-Ing. Ralf Gessler

Rev. 1.0

Content

- Chapter 2: Stellaris® family of microcontrollers

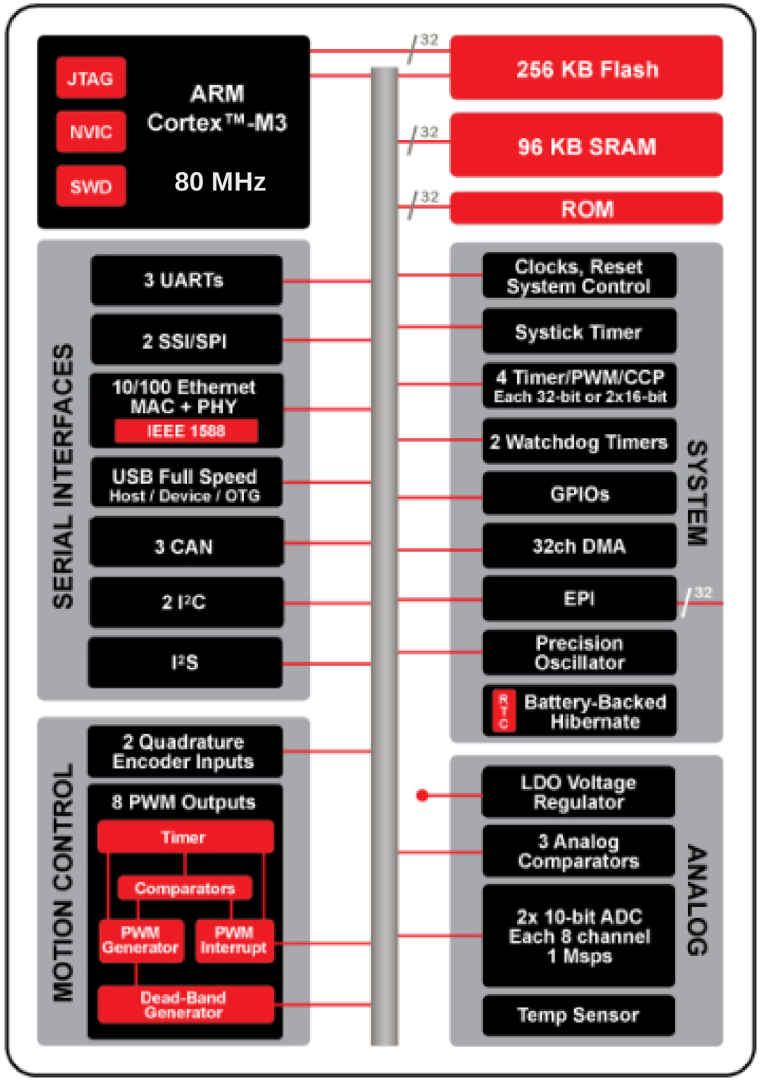
- **Topics:** MCU, ARM® processor core, on-chip memory, **system peripherals**, serial interface peripherals, motion control peripherals, analog peripherals

Learning Objectives

- The chapter describes the basics (theory) of the Stellaris® Family
  - Architecture (common)
  - Stellaris® Module "System".
- Additional Links for the other Stellaris® Modules are given.
- Structure and questions:
  - What are the features of the Stellaris® family?
  - What are the Stellaris® product lines?
  - What does a Stellaris® Module "System" looks like?

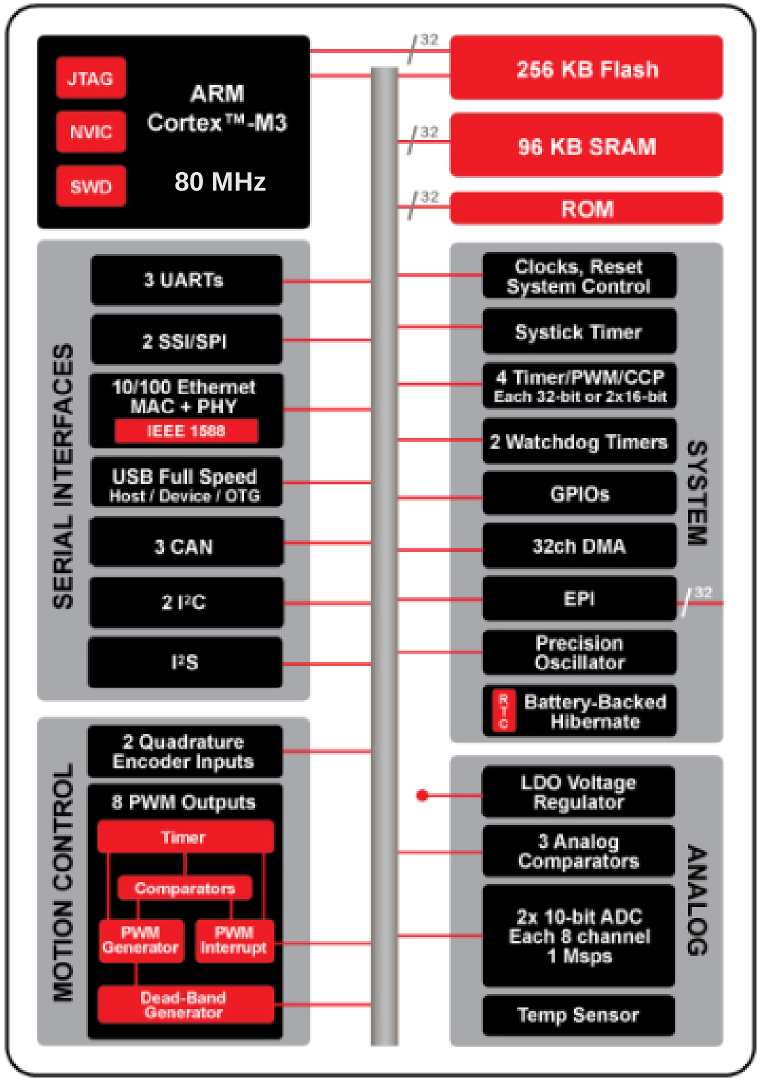
Overview MCU

- Microcontroller Unit (MCU)
- ARM® Cortex™ -M3 Processor Core
  - Up to 100 MHz
  - Joint Test Action Group (JTAG) Interface
  - Nested Vectored Interrupt Ctrl. (NVIC)
  - Serial Wire Debug (SWD)
- On-chip Memory
  - 256 KB Flash, 96 KB SRAM
  - ROM loaded with Stellaris® DriverLib, Bootloader, AES tables, and CRC
- System Peripherals
  - 32-channel DMA Controller
  - Internal Precision 16MHz Oscillator
  - Two watchdog timers with separate clock domains
  - ARM Cortex SysTick Timer
  - 4x 32-bit timers (up to 8 16-bit) with RTC capability



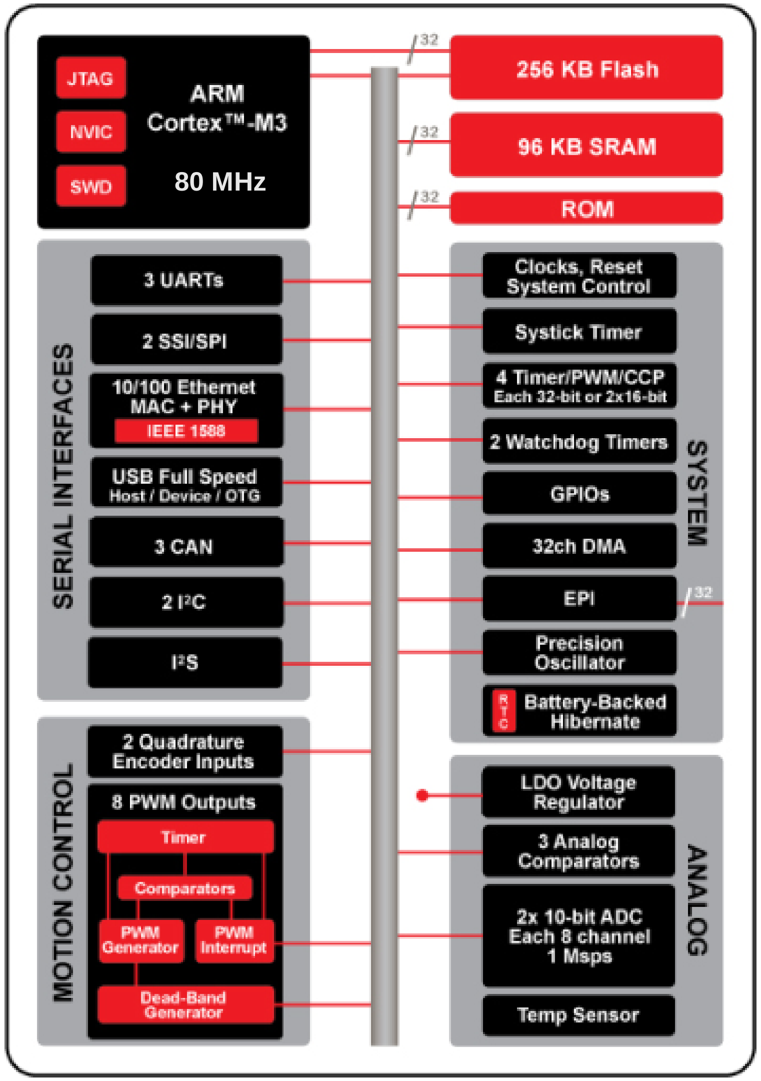
Overview (cont.)

- System Peripherals (cont.)
  - Lower-power battery-backed hibernation module
  - External Peripheral Interface (EPI)
    - 32-bit dedicated parallel bus for external peripherals
    - Supports SDRAM, SRAM/Flash, M2M
- Serial Interface Peripherals
  - 3 Univ. Asyn. Rec./Trans. (UARTs) with Infrared Data Association (IrDA) and ISO 7816 support
  - 2 Synchronous Serial Interfaces (SSI)
  - 10/100 Ethernet MAC and PHY
  - Universal Serial Bus (USB) (full speed) OTG / Host / Device
  - 3 Controller Area Network (CAN) 2.0 A/B Controllers
  - 2 Inter-Integrated Circuits (I^2Cs)
  - Integrated Interchip Sound (I^2S)



Overview (cont.)

- Motion Control Peripherals
  - 8 advanced PWM outputs for motion and energy applications
  - 2 Quadrature Encoder Inputs (QEI)
- Analog Peripherals
  - 2 x 8-ch 10-bit ADC (for a total of 16 channels)
  - 3 analog comparators
  - On-chip voltage regulator (1.2V internal operation)
  - Low-Drop-Out (LDO)
- Flexible pin-muxing capability



Device Choice

- Product lines: LM3S1nnn to LM3S9nnn

- ARM® Cortex™-M3 Processor Core
  - Same Cortex™ -M3 core
  - CPU frequency depends on product lines
- Memory
  - Size of internal Flash and SRAM depends on product lines
- System peripherals
  - Quantity depends on product lines

same functionality

Device Choice (cont.)

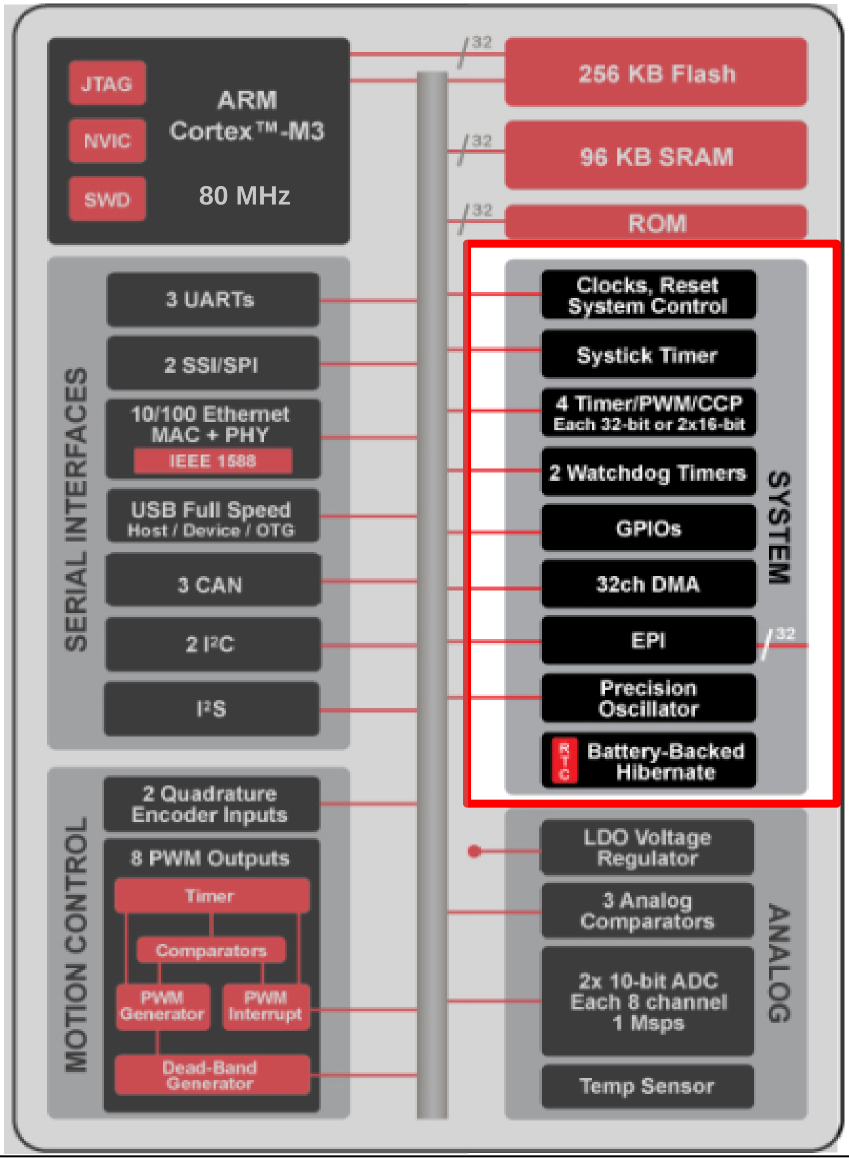
- Serial interface peripherals
  - Features depends on product lines
- Motion control peripherals
  - Features depends on product lines
- Analog peripherals
  - Features depends on product lines
- Note: For each target application a product line

functionality depends on product line

Section Links: Peripherals

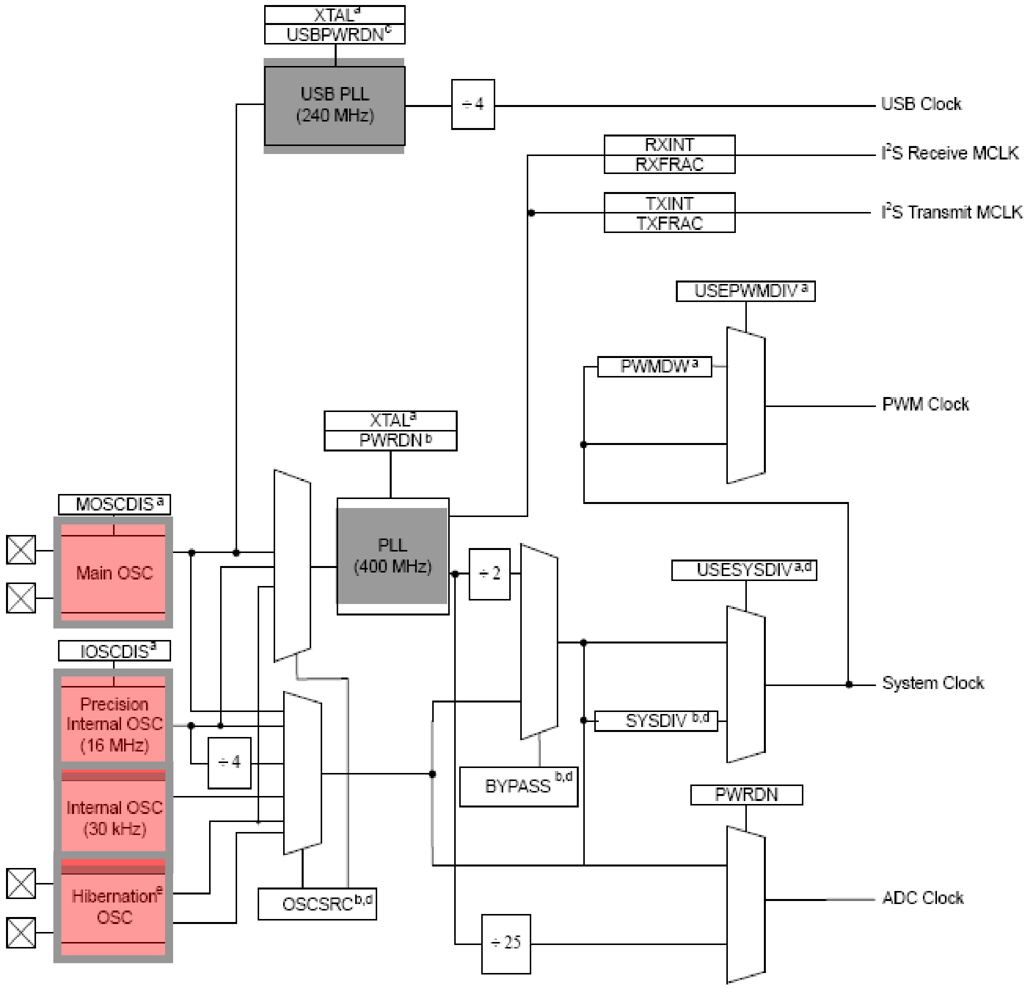
- ARM® Cortex™ -M3 Processor Core
  - More Details see: Chapter 2.3 “Cortex® -M3: Programmer model (ISA)”
- On-Chip Memory
  - More Details see: Chapter 2.2 “Cortex® -M3: Processor and Peripherals”
- Serial Interfaces Peripherals
  - More Details see:
    Module 2.4 “Peripherals”: UART, SSI, I^{2}C - LM3S1968
    Module 5.3 “Connectivity”: Ethernet, USB - LM3S9B92
- Motion Control Peripherals
  - More Details see: Chapter 2.4 “Peripherals” - LM3S1968
- Analog Peripherals
  - More Details see: Chapter 2.4 “Peripherals” - LM3S1968

System Peripherals



Clock architecture

- Main OSCillator (OSC)
  - 3.579545 MHz through 16.384 MHz if PLL used
- Precision Internal Oscillator (PIOSC)
  - On-chip precision 16 MHz clock (clk) (±1% at room temperature)
  - Does not require any external components
- Internal OSC
- Hibernation OSC



Example: System Control

- System Control determines the overall operation of the device.
  It controls the clocking of the device, the set of peripherals that are enabled, configuration of the device, its resets and provides information about the device.

- System Control is part of Stellaris® Peripheral Driver Library.
  - Link: see Chapter 3.4 “StellarisWare®”

- Example:

```
…
//
// Set the clocking to run directly from the crystal.
//
SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
             SYSCTL_XTAL_8MHZ);
…
```

  - Link: see Lab “lab21a.zip”

Reset

- Flexible Reset Sources
  - Power-on Reset (POR)
  - Reset pin assertion
  - Brown-Out Reset (BOR) detector alerts to system power drops
    - The system provides a brown-out detection circuit that triggers if the power supply (VDD) drops below a Voltage Brown-out THreshold (VBTH). If a brown-out condition is detected, the system may generate a controller interrupt or a system reset.
    - This is initially disabled and may be enabled by software.
  - Software reset
  - Watchdog timer reset

System Control

- For power-savings purposes, the RCGCn , SCGCn , and DCGCn registers control the clock gating logic for each peripheral or block in the system while the controller is in Run, Sleep, and Deep-Sleep mode, respectively.

- There are four levels of operation for the device defined as:
  - Run Mode: the controller actively executes code
  - Sleep Mode: the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code.
    Sleep mode is entered by the Cortex-M3 core executing a WFI (Wait for Interrupt) instruction.
    Any properly configured interrupt event in the system will bring the processor back into Run mode.

System Control (cont.)

- There are four levels of operation for the device defined as (cont.):
  - Deep-Sleep Mode: the clock frequency of the active peripherals may change (depending on the Run mode clock configuration) in addition to the processor clock being stopped.
    An interrupt returns the device to Run mode from one of the sleep modes.
  - Hibernate Mode: the power supplies are turned off to the main part of the device and only the Hibernation module's circuitry is active.
    An external wake event or Real Time Clock (RTC) event is required to bring the device back to Run mode.

System Timer

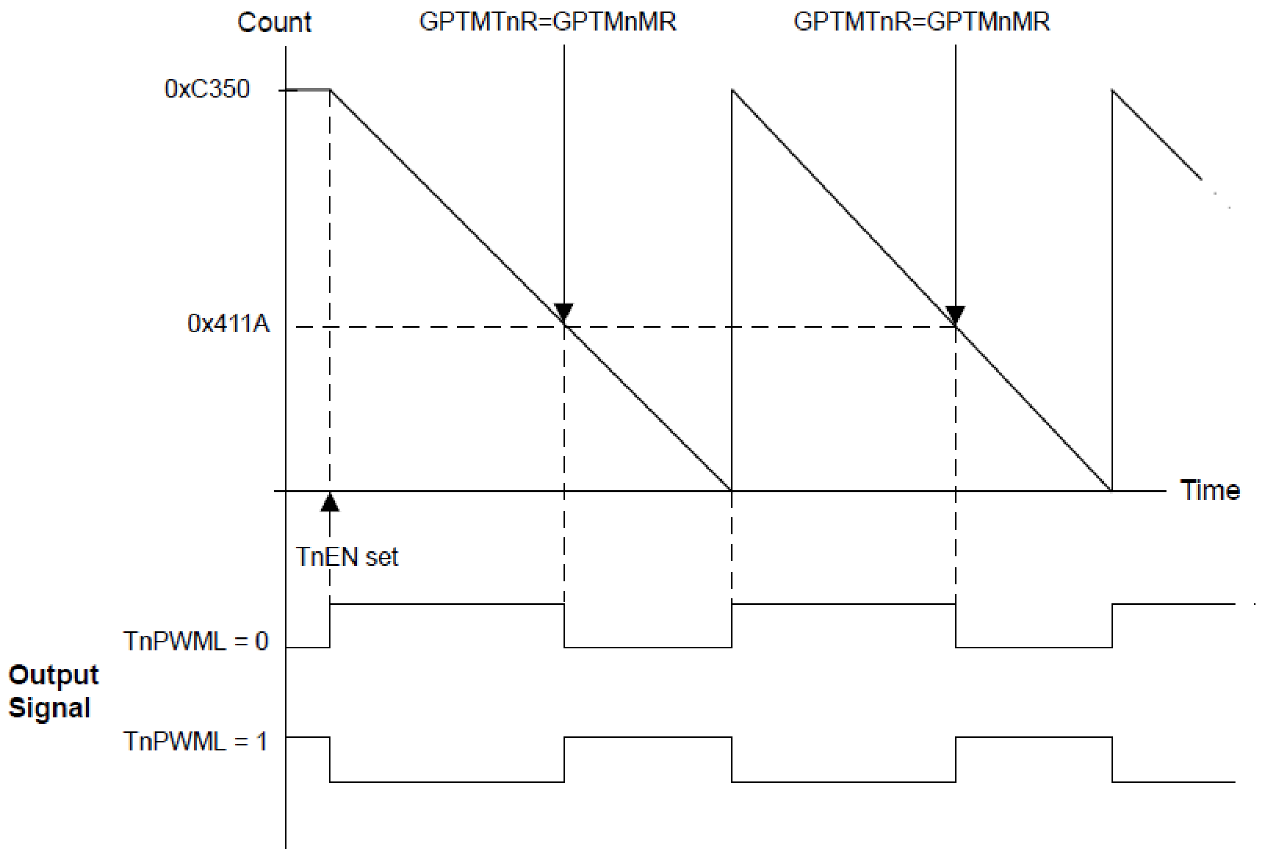
- Cortex^{TM} -M3 includes an integrated system timer, SysTick. SysTick provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.
- Link: see Chapter 2.2 “Cortex^{TM} -M3: Processor and Peripherals”

Timers

- Up to 4 timer modules
  - Each configurable as single 32-bit timer, or two 16-bit timers
- 32-bit Timer Modes
  - 32-bit programmable one-shot timer with selectable clock
  - 32-bit programmable periodic timer with selectable clock
  - RTC using a 32.768 kHz input clock
  - Software controlled debug event stalling (except for the RTC function)
- 16-bit Timer Modes
  - 16-bit general-purpose timer with 8-bit pre-scaler and input clock selection
  - General-purpose free running timer modes: one-shot or periodic
  - Selectable debug event stalling
- 16-bit Capture Modes (CCP)
  - Input edge count capture
  - Input edge time capture
- PWM mode
  - Simple PWM mode with programmable output negation

Timers: PWM

- PWM signals are driven through CCPx output



Watchdog Timers

- Allows software to regain control when there is a system failure caused by software error or an external device
- Generates an interrupt or system reset on time-out
- Configuration can be locked (key) to prevent inadvertent changes
- User-enabled stalling for software debugging
- Some Stellaris® MCUs feature two Watchdog Timer Modules, where :
  - One module is clocked by the system clock (Watchdog Timer 0)
  - One module is clocked by the PIOSC (Watchdog Timer 1).

Example: Watchdog Timer

- Watchdog Timer is part of Stellaris® Peripheral Driver Library.
- The Watchdog Timer Application Programming Interface (API) provides a set of functions for using the Stellaris® watchdog timer modules. Functions are provided to deal with the watchdog timer interrupts, and to handle status and configuration of the watchdog timer.
- Link Chapter 3.4 “StellarisWare®”
- The following example shows how to set up the watchdog timer API. If the watchdog is not periodically fed, it will reset the system. Each time the watchdog is fed, the LED is inverted so that it is easy to see that it is being fed, which occurs once every second.
- Link see Lab “lab21a.zip”
  - Note: example, not a complete application!

Example: Watchdog Timer (cont.)

```
…
//
// Enable the watchdog interrupt.
//
IntEnable(INT_WATCHDOG);

//
// Set the period of the watchdog timer.
//
WatchdogReloadSet(WATCHDOG0_BASE, SysCtlClockGet());

//
// Enable reset generation from the watchdog timer.
//
WatchdogResetEnable(WATCHDOG0_BASE);

//
// Enable the watchdog timer.
//
WatchdogEnable(WATCHDOG0_BASE);

//
// Loop forever while the LED winks as watchdog interrupts are handled.
//
while(1)
{
}
…
```

GPIOs

- General Purpose Input/Outputs (GPIOs)
- Programmable pad configuration through GPIO module
- Any GPIO can be full featured external interrupt
- Bit addressable pins, atomic operation
- Fast output toggling: Toggle rate up to ½ the CPU clock
- 5-V-tolerant inputs

GPIOs (cont.)

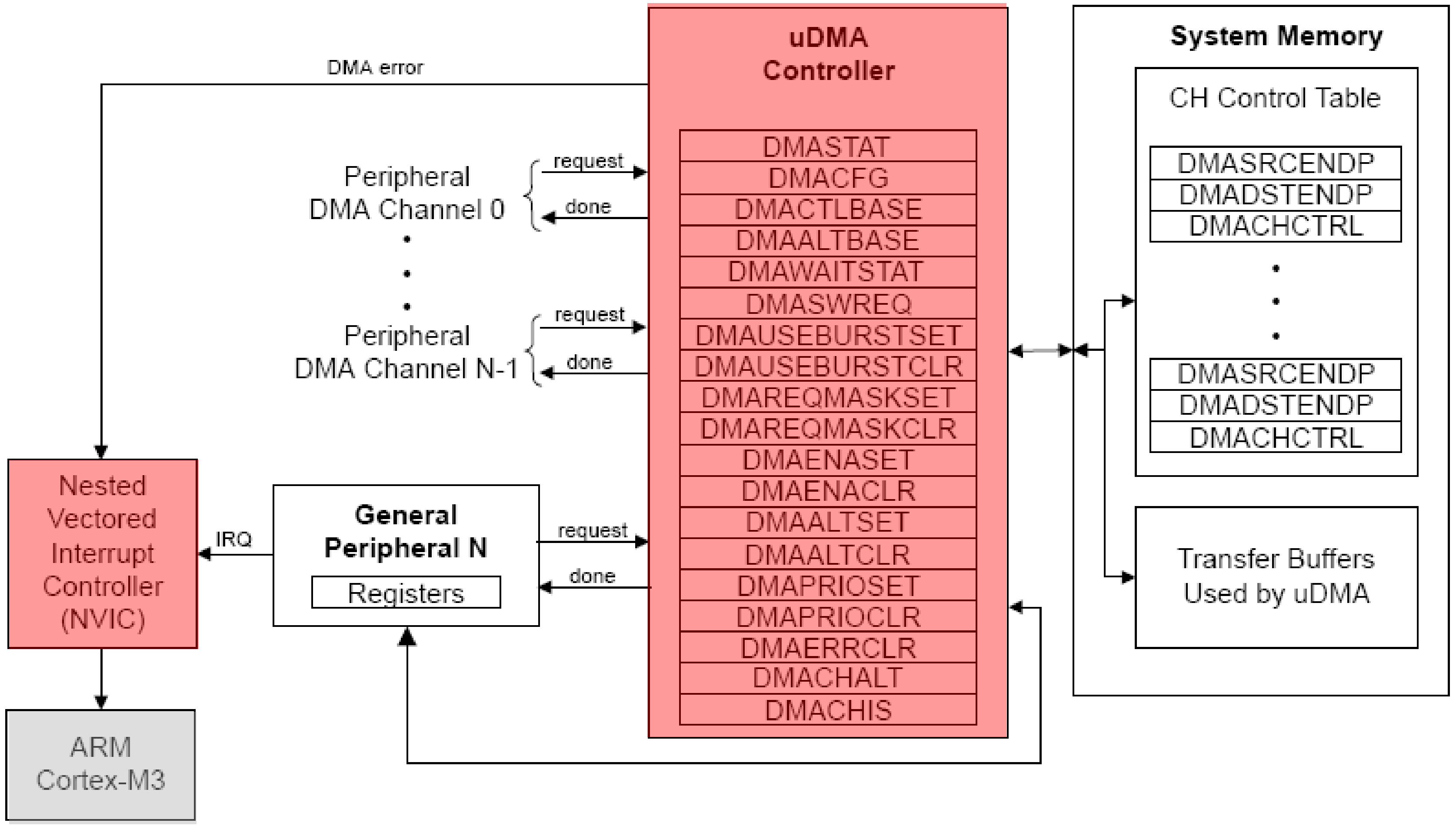
- Up to 61 GPIO available on a single device
- Programmable Drive Strength
  - 2 mA, 4 mA, 8 mA or 8 mA with slew rate control
- Programmable weak pull-up, pull-down
- Open drain enables
- Digital input enables

- Link: see Chapter 4.1 “General Purpose Input/Output (GPIO)”

Direct Memory Access

- Direct Memory Access (DMA)
- 32-channel configurable µDMA controller
- Dedicated channels for supported peripherals
  - One channel each for receive and transmit path for bidirectional peripherals
  - Multiple data sizes, Two levels of priority, Maskable device requests
- Interrupt on transfer completion with a separate interrupt per channel
- Support for multiple transfer modes:
  - Basic, for simple transfer scenarios
  - Ping-pong, for continuous data flow to/from peripherals
  - Scatter-gather, from a programmable list of arbitrary transfers initiated from a single request
- DMA requests
  - UART, Timer, USB, Ethernet, ADC, SSI, External Peripherals I/F

DMA (cont.)

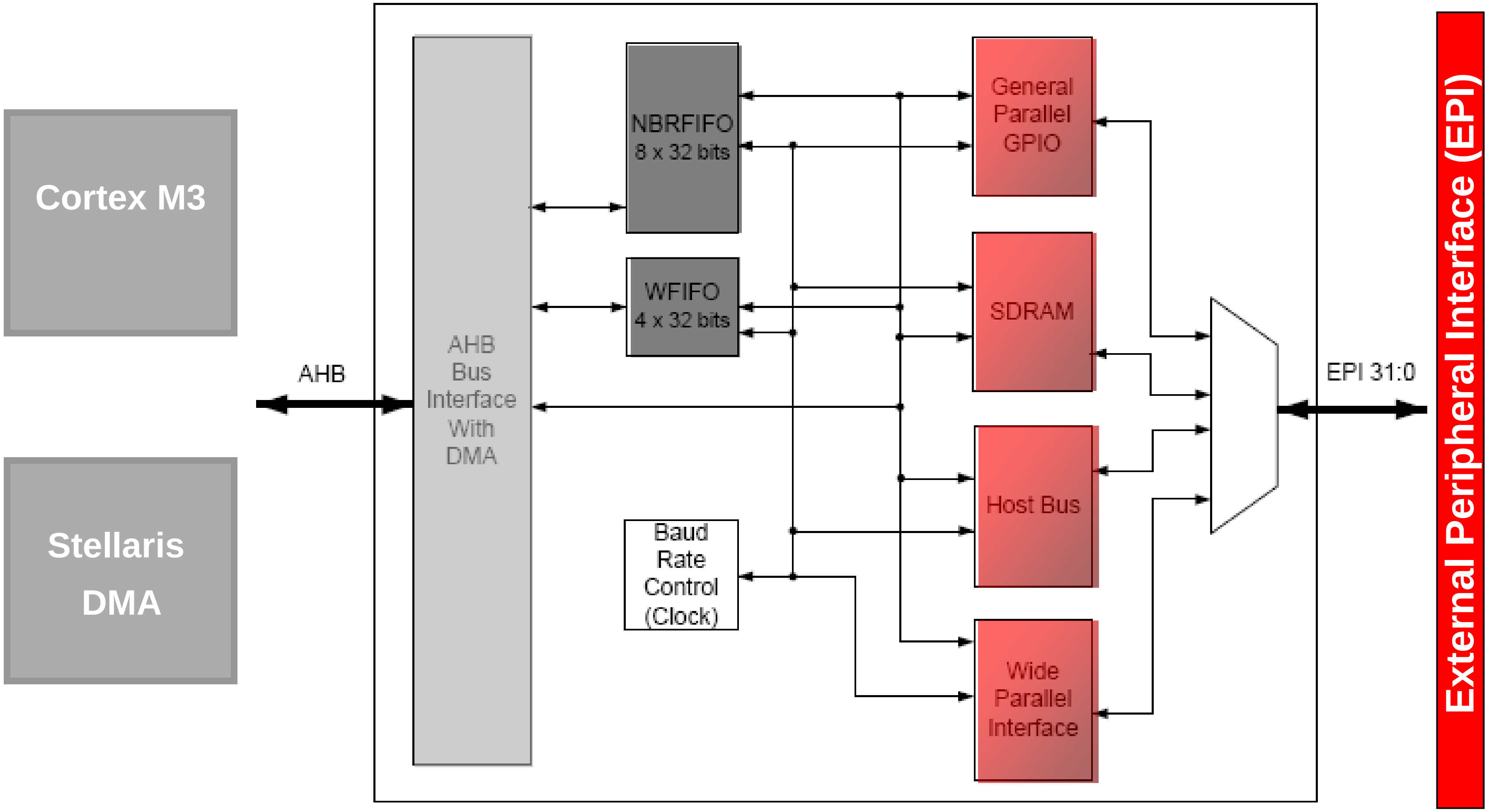


uDMA Controller
DMASTAT
DMACFG
DMACTLBASE
DMAALTBASE
DMAWAITSTAT
DMASWREQ
DMAUSEBURSTSET
DMAUSEBURSTCLR
DMAREQMASKSET
DMAREQMASKCLR
DMAENASET
DMAENACLR
DMAALTSET
DMAALTCLR
DMAPRIOSET
DMAPRIOCLR
DMAERRCLR
DMACHALT
DMACHIS
System Memory
CH Control Table
DMASRCENDP
DMADSTENDP
DMACHCTRL
DMASRCENDP
DMADSTENDP
DMACHCTRL
Transfer Buffers Used by uDMA
DMA error
Peripheral DMA Channel 0
request
done
Peripheral DMA Channel N-1
request
done
Nested Vectored Interrupt Controller (NVIC)
IRQ
General Peripheral N
Registers
request
done
ARM Cortex-M3

External Peripheral Interface

- External Peripheral Interface (EPI)
- Multiple device types supported
  - SDRAM: Supports x16 (Single Data Rate) at up to 50MHz
    - Supports low-cost SDRAMS up to 64 MB
    - Includes automatic refresh and access to all banks/rows
    - Includes a sleep/standby mode to keep contents alive with minimal power draw
  - Host-Bus Interface: Traditional x8 and x16 MCU bus interface capabilities
    - Support of both muxed and de-muxed address and data
    - Access to SRAM, NOR Flash, and other devices, with up to 1MB of addressing or up to 256 MB for muxed mode
    - Speed controlled, with read and write data wait-state counters
  - Machine-to-Machine (M2M): Wide parallel interfaces for fast communications
    - For instance, CPLDs and FPGAs
    - Data widths up to 32-bits, data rates up to 150 Mbytes/second
    - Optional “address” sizes from 4-bits to 16-bits
    - Optional clk output, read/write strobes, framing (with counter-based size), and clk-enable input
- Other features
  - General parallel GPIO, FIFOed with speed control
  - Blocking and non-blocking reads
  - FIFOed writes separate the processor from timing details
  - Direct memory access (DMA)

EPI block diagram

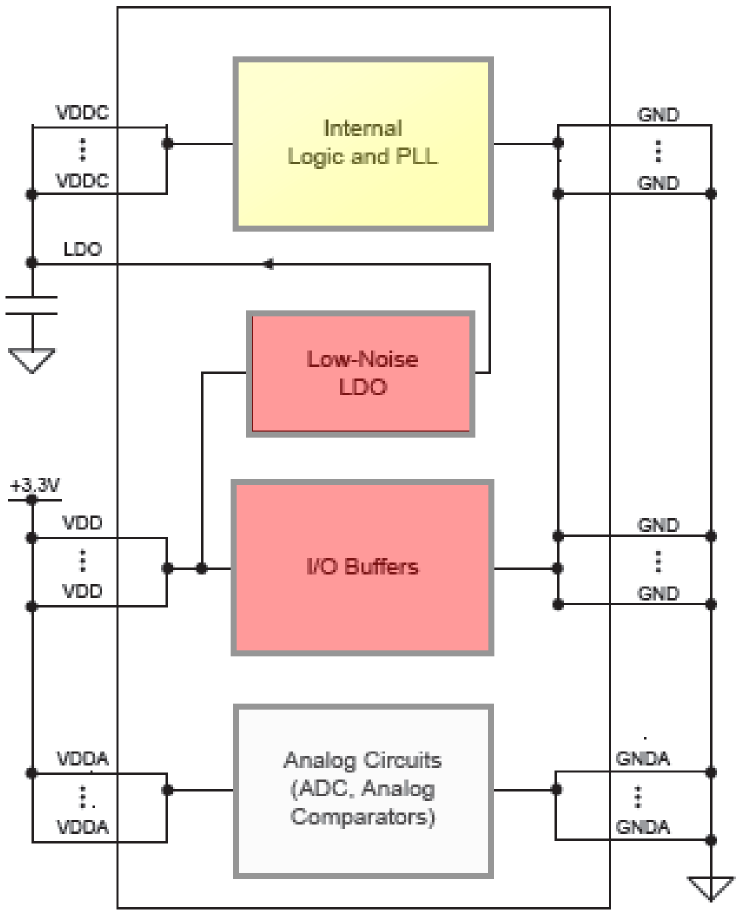


Hibernation module

- Two mechanisms for power control
  - System power control using discrete external regulator
  - On-chip power control using internal switches under register control
- Dedicated pin for waking using an external signal
- Low-battery detection, signaling, and interrupt generation
- 32-bit real-time counter (RTC)
  - Two 32-bit RTC match registers for timed wake-up and interrupt generation
  - RTC pre-divider trim for making fine adjustments to the clock rate
- Clock source from a 32.768 kHz external oscillator or a 4.194304 MHz crystal - 32.768 kHz external oscillator can be used for main clock
- 64 32-bit words of non-volatile memory to save state during hibernation
- Programmable interrupts for RTC match, external wake, and low battery events

Power architecture

- 3 power domains
  - Core Supply: VDDC
  - Supply Voltage: VDD
  - Analog circuitry: VDDA



Power consumption

- Battery-backed Hibernation Module (Standby current as low as 10µA)
  - 32-bit real-time counter (RTC)
    - Programmable 32.768 kHz external oscillator or a 4.194304 MHz crystal
    - RTC software trim for making fine adjustments to the clock rate
  - Power-switching logic to discrete external regulator (switch to battery)
  - Low-battery detection, signaling, and interrupt generation
  - Wake on RTC match and / or external pin
- On-chip Low Drop-Out (LDO) voltage regulator
- Low-power options on controller: Sleep and Deep-sleep modes
- Low-power options for peripherals: software controls shutdown of individual peripherals
- 3.3V supply brown-out detection and reporting via interrupt or reset

Power consumption (cont.)

	SRAM saved	Lowest Ibat	Lowest type wake-up
Running mode	Yes	1.12 mA/MHz	N/A
Sleep mode	Yes	8 mA	0.427µs
Deep-Sleep mode	Yes	550 µA	0.427µs
Hibernate RTC ON	256 bytes saved	18 µA	62 µs
Hibernate RTC OFF	256 bytes saved	8 µA	10 ms

- Note: Battery supply current (Ibat)

Example: Sleep Mode

- SysCtlSleep and SysTick are parts of System Control (Stellaris® Peripheral Driver Library).
- CPUUsage is part of LM3S1968 Firmware Development Package.
- Link: see Chapter 3.4 “StellarisWare®”
- Link: see Lab “lab21b.zip”
  - Note: example, not a complete application!
- The following example shows how to use the CPU usage module to measure the CPU usage where the foreground simply burns some cycles.

Example: Sleep Mode

```
…
//
// The CPU usage for the most recent time period.
//
unsigned long g_ulCPUUsage;

//
// Handles the SysTick interrupt.
//
void
SysTickIntHandler(void)
{
//
// Compute the CPU usage for the last time period.
//
g_ulCPUUsage = CPUUsageTick();
}

//
// The main application.
//
int
main(void)
{
//
// Initialize the CPU usage module, using timer 0.
//
CPUUsageInit(8000000, 100, 0);
…
```

```
…
//
// Initialize SysTick to interrupt at 100 Hz.
//
SysTickPeriodSet(8000000 / 100);
SysTickIntEnable();
SysTickEnable();
//
// Loop forever.
//
while(1)
{
//
// Delay for a little bit so that CPU usage is not zero.
//
SysCtlDelay(100);
//
// Put the processor to sleep.
//
SysCtlSleep();
}
}
…
```

Questions and Exercises

1. What are the main modules of the Stellaris® family?
2. Explain a Watchdog Timer.
3. Explain Hibernation.
4. What is DMA?

Summary and Outlook

- Summary
  - From the Stellaris® Family Architecture in common to Stellaris® family product lines
  - From the Device Choice to Module “System”
- Outlook/How to go on?
  - Chapter 4: “Peripheral Programming in C” shows the practical use in C

References

- [1] Henri, G.; Texas Instruments: *MCU Training Module - System and Peripherals.* EMEA, Oct. 7, 2010.
- [2] Henri, G.; Texas Instruments: *MCU Training Module - Solution overview*. EMEA, Jul. 21, 2010.
- [3] Texas Instruments: *Data Sheet - Stellaris® LM3S1968.* Chapter 1: “Architectural Overview”, Chapter 5.1.5: “System Control”, spms037f.pdf, 2011.
- [4] Texas Instruments: *User’s Guide - Stellaris Peripheral Driver Library*. SW-DRL-UG-6075.pdf, 2010.
- [5] Texas Instruments: *User’s Guide - EK-LM3S1968 Firmware Development Package*. SW-EK-LM3S1968-UG-6075.pdf, 2010.