INTERFACING

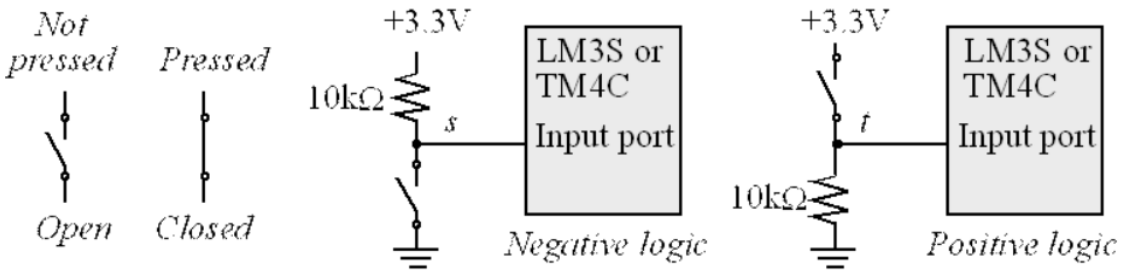


Figure 8.2. Single Pole Single Throw (SPST) Switch interface.

Maximum voltage on I/O pins: Typically **3.3V**.

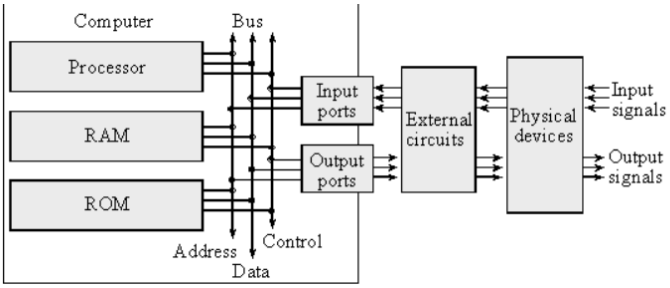
8 mA (standard) or up to **16 mA (high drive strength option)**.

Typically limited to **100-200 mA**, depending on the microcontroller variant.

Von Neumann: use single program

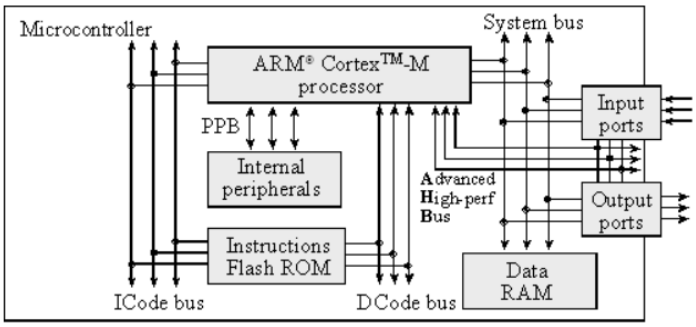
And data space, making them

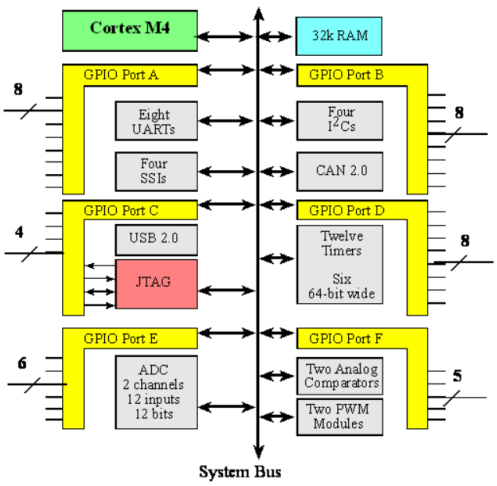
Transferred sequentially



Harvard : separate the memory which

allow simultaneous access





A , B, D ,C → 8

E →6

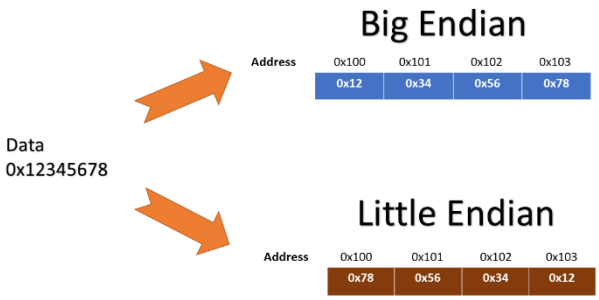
F→5

Little-Endian : the least significant byte (LSB) is stored first

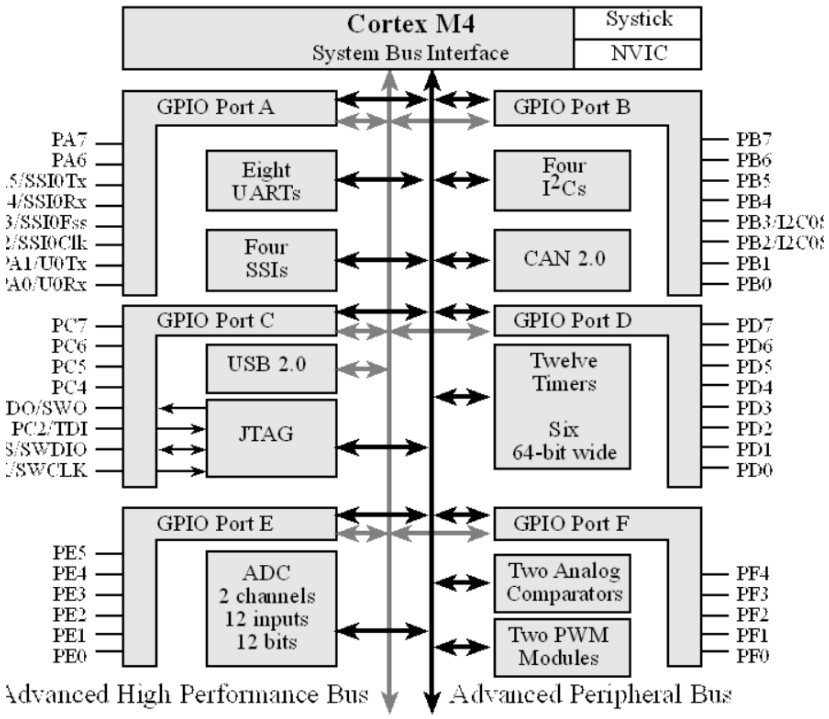
Big-Endian : The most significant byte (MSB) is stored first

Texas Instruments use the **little-endian** but

Support both



- Nibble 4 bits
- Byte 8 bits
- Halfword 16 bits
- Word 32 bits



get a bus fault if you access a port without enabling its clock.

Port Initialization Steps

- **Step 1**: Activate the clock for the port by setting the appropriate bit in SYSCTL_RCGCGPIO_R. This step provides power to the port.
- **Step 2**: Unlock the port (only for specific pins like PC3-0, PD7, PF0). Write the value 0x4C4F434B to the GPIO_PORTx_LOCK_R register to enable changes.
- **Step 3**: Disable analog functions on the port by clearing bits in GPIO_PORTx_AMSEL_R.

- **Step 4**: Configure the pin for GPIO functionality by clearing bits in the PCTL register (GPIO_PORTx_PCTL_R).
- **Step 5**: Set pin direction using GPIO_PORTx_DIR_R (0 for input, 1 for output).
- **Step 6**: Disable alternate functions by clearing bits in GPIO_PORTx_AFSEL_R.
- **Step 7**: Enable digital I/O by setting bits in GPIO_PORTx_DEN_R.

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I^2C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

RCGCGPIO: Enables clock for GPIO ports.

SCGCGPIO / DCGCGPIO: Clock control in Sleep/Deep-sleep modes.

GPIODIR: Sets pin direction (1 for output, 0 for input).

GPIOAFSEL: Selects GPIO (0) or alternate functions (1).

GPIOPCTL: Specifies peripheral function for alternate mode.

GPIOADCCTL / GPIODMACTL: Configures ADC or µDMA triggers.

GPIODR2R / GPIODR4R / GPIODR8R: Sets drive strength (2/4/8 mA).

GPIOPUR / GPIOPDR: Enables pull-up/pull-down resistors.

GPIOODR: Enables open-drain mode.

GPIOSLR: Controls slew rate.

GPIODEN: Enables digital I/O.

GPIOAMSEL: Enables analog mode.

GPIOIS / GPIOIBE / GPIOIEV / GPIOIM: Configures interrupts.

GPIOLOCK: Locks critical pin configurations.

Default State on Reset

- **GPIOAFSEL = 0, GPIODEN = 0, GPIOPDR = 0, GPIOPUR = 0**

RCGC2 is an older register for clock control; it mirrors changes to **RCGCGPIO** but lacks support for newer peripherals.

Writing to **RCGC2** affects **RCGCGPIO** and vice versa, but **RCGC2** cannot read changes for newer modules reflected only in **RCGCGPIO**.

Run Mode Clock Gating Control Register 2 (RCGC2)
Base 0x400F.E000
Offset 0x108
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															USB0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved							GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

If a bit in the GPIOCR register is **set** (1), the corresponding bit in the configuration registers is **committed**.

If a bit is **cleared** (0), the value written to the corresponding configuration register is **not committed** and retains its previous value.

When you write to certain configuration registers (e.g., **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, **GPIODEN**), these changes **won't take effect immediately**. They will only take effect if the corresponding bit in the **GPIOCR** register is set.

If the **GPIOCR** register is **not set** for a particular bit, the change you try to make to a pin's configuration will **not be committed**. The configuration for that pin will stay the same.

- R0 – R12: general purpose registers
    - Low registers (R0 – R7) can be accessed by any instruction
    - High registers (R8 – R12) sometimes cannot be accessed e.g. by some Thumb (16-bit) instructions

---

- A **breakpoint** halts the program at a specific line of code, allowing you to inspect the program's state at that point.
- A **watchpoint** pauses the program when a specific variable or memory location changes its value, helping track when and why a variable's value changes.

- **AHB** (Advanced High-performance Bus) is a high-speed bus for memory and high-performance peripherals, supports faster data transfer and is more complex.
- **APB** (Advanced Peripheral Bus) is a slower bus used for low-speed peripherals. APB is simpler, lower power, and optimized for devices like GPIO and timers.

Early ARM Instruction Set (ARM Instructions):

- **32-bit instruction set** known as **ARM instructions**.
- Offers **powerful performance** and **efficiency** for computational tasks.
- **Larger program memory** requirements compared to **8-bit** and **16-bit** processors.
- **Higher power consumption** due to 32-bit instruction width.

Thumb-1 Instruction Set:

- **16-bit instruction set** introduced with the **ARM7TDMI processor** in 1995.
- **Subset of ARM instructions**, providing better **code density** compared to the full 32-bit ARM set.
- **Code size reduced by ~30%**, but **performance decreased by ~20%** as a trade-off.
- Thumb-1 offers **compact code** but at a **lower performance**.

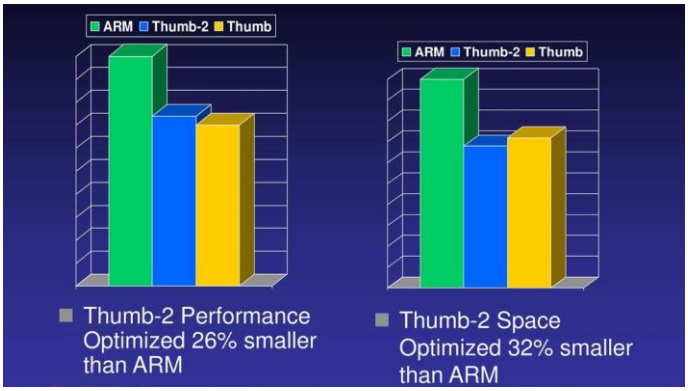
Mix of ARM and Thumb-1 Instruction Sets:

- Combination of **32-bit ARM** (for **high performance**) and **16-bit Thumb-1** (for **high code density**).
- A **multiplexer** is used to switch between the two modes:
    - **ARM state** (32-bit mode)
    - **Thumb state** (16-bit mode).
- **Switching between modes introduces overhead**, impacting efficiency.

Thumb-2 Instruction Set:

- Combines **32-bit Thumb instructions** and **16-bit Thumb-1 instructions** into one unified set.

- **Code size reduced by ~26%** compared to the full 32-bit ARM instructions, offering **improved memory efficiency**.
- **Performance** is similar to the ARM instruction set, without the significant loss seen in Thumb-1.
- **Single operation state**: Capable of handling all processing needs with the **Thumb-2 instruction set**, which balances **compactness** and **performance**.



Bit-banding

enables accessing a single bit in memory using a single load/store operation.

Normal Operation (Read-Modify-Write): Involves reading a 32-bit value, modifying a bit, and then writing it back. Inefficient as it requires extra operations.

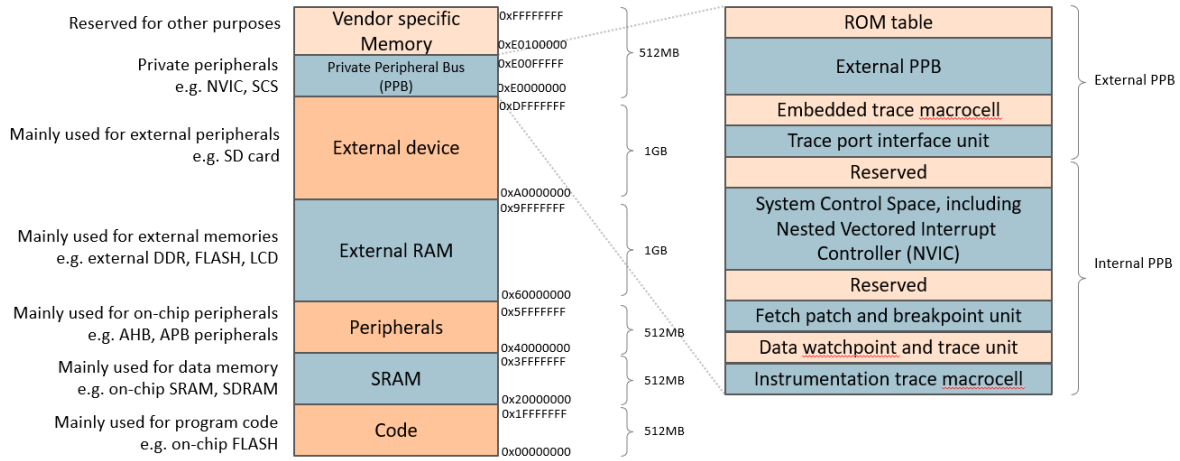
Bit-Band Operation: Directly manipulates individual bits using alias addresses. For example, writing to **0x2200000C** sets bit 3 of data at **0x20000000** without modifying other bits. This reduces overhead and improves performance.

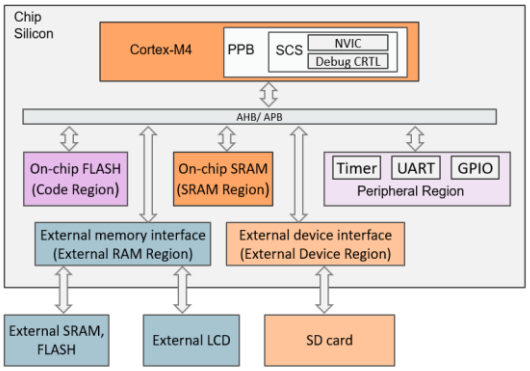
Benefits 1- faster 2- fewer instructions 3- Atomic operation

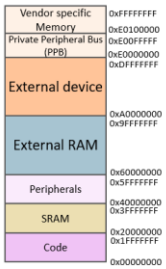
bit_word_offset = (byte_offset x 32) + (bit_number x 4)

bit_word_addr = bit_band_base + bit_word_offset

Assembly code : <operation> <destination>, <source1>, <source2>







Cortex has 4 GB divided into

1. Code (0x00000000 - 0x1FFFFFFF)

- **Purpose:** Stores program code, typically located in on-chip FLASH memory.
- **Size:** 512MB.
- **Usage:** This region contains the executable firmware and may also include bootloaders.

2. SRAM (0x20000000 - 0x3FFFFFFF)

- **Purpose:** Provides on-chip data memory for variables and stack.
- **Size:** 512MB.
- **Usage:** Used for runtime data, including the heap and stack space.

3. Peripherals (0x40000000 - 0x5FFFFFFF)

- **Purpose:** Reserved for on-chip peripherals (e.g., AHB and APB peripherals).

- **Size:** 512MB.
- **Examples:** Includes GPIOs, timers, UART, SPI, I2C, and ADC peripherals.

4. External RAM (0x60000000 - 0x9FFFFFFF)

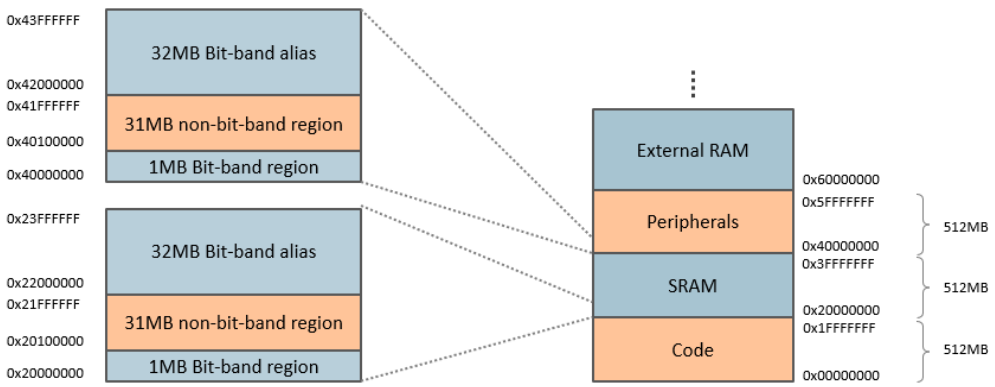
- **Purpose:** Allows interfacing with external memory devices like DDR, FLASH, or LCD modules.
- **Size:** 1GB.
- **Usage:** Expands the system's memory capability for large applications.

5. External Device (0xA0000000 - 0xDFFFFFFF)

- **Purpose:** Designated for external peripherals like SD cards or other external controllers.
- **Size:** 1GB.

6. Vendor-Specific Memory (0xE0000000 - 0xFFFFFFFF)

- **Purpose:** Contains private peripheral bus (PPB) memory for core peripherals, system control, and debugging components.
- **Size:** 512MB.



SRAM Bit-Band Region (0x20000000 - 0x201FFFFF):

- **Size:** 1MB.
- This is a part of the on-chip SRAM where bit-banding is enabled.
- Each bit in this region can be accessed via the corresponding alias address in the **SRAM Bit-Band Alias Region (0x22000000 - 0x23FFFFFF)**.

Accessing bit 5 of address `0x20000004` (in the SRAM bit-band region)

Formula:

Alias Address=0x22000000+(ByteOffset×32) +(BitPosition×4)

plug in the values:

Alias Address=0x22000000+(0x00000004×32) +(5×4)

Perform the calculations:

- 0x00000004×32=0x00000080
- 5×4=20=0x00000014

  Add these to the base address:

  Alias Address = 0x22000000+0x00000080+0x00000014=0x22000094

Peripherals Bit-Band Region (0x40000000 - 0x401FFFFF):

- **Size:** 1MB.
- This region includes on-chip peripherals (e.g., GPIO, UART, etc.), where bit-banding is supported.
- Each bit in this region can be accessed via the corresponding alias address in the **Peripherals Bit-Band Alias Region (0x42000000 - 0x43FFFFFF)**.

After reset

1- **Read MSP from First Address**

- The processor reads the **Main Stack Pointer (MSP)** from address 0x00000000 in the vector table.

2- **Read Reset Vector**

- The processor reads the **Reset Handler address** from address 0x00000004 (second word in the vector table).

3- **Branch to Reset Handler**

- The processor loads the Reset Handler address into the **Program Counter (PC)** to start execution.

4- **Begin Execution**

- Execution begins from the Reset Handler, which initializes the system and calls the main() function.