| | |
|---|---|
| Used by default after reset | Not used by default; must be explicitly set |
| Used in Handler mode (during exception handling) | Used in Thread mode (when not handling exceptions) |
| CONTROL[1] = 0 | CONTROL[1] = 1 |
| Default visible stack pointer | Only visible when explicitly selected |
| Loaded from the first 32-bit word of the vector table | Undefined; must be initialized by software |
| Used in many simple applications and by the OS kernel | Typically used by application processes in an OS environment |
| Accessed using "R13" or "SP" | Accessed using special register names "PSP" |

What is the role of the Link Register (LR) in the Cortex-M0 processor?

Edit

Delete

R14 is the Link Register (LR). It stores the return address of a subroutine or function call. At the end of the subroutine or function, the return address stored in LR is loaded into the program counter (PC) to resume execution of the calling program. In some cases, such as exceptions, LR provides a special code value for the exception return mechanism.

## What are R13, SP, MSP, and PSP?

- **R13:** Acts as the stack pointer.
- **SP:** Alias for R13, the stack pointer.
- **MSP (Main Stack Pointer):** Default stack pointer after reset and used during exception handling.
- **PSP (Process Stack Pointer):** Used in Thread mode; its stack pointer selection is controlled by the CONTROL register.

What are the types of memory regions defined in Cortex-M0 memory map?

---

- CODE: Mainly used for program code and exception vector table.
- SRAM: Mainly used for data memory (e.g., static RAM).
- External RAM: Used for external memory.
- External Device: Mainly used for peripherals.
- Peripherals: Mainly used for external peripherals.
- System: Private peripherals including built-in interrupt controller (NVIC) and debug components.
- Private Peripheral Bus (PPB) and System Control Space (SCS): Include specific control components.

List and explain the APSR flags in the Cortex-M0 processor with examples.

---

The Application Program Status Register (APSR) has four flags:

- Negative (N): Indicates a negative ALU result.
- Zero (Z): Indicates an ALU result of zero.
- Carry (C): Indicates a carried out or borrowed value in addition/subtraction.
- Overflow (V): Indicates an overflow in arithmetic operations.

Example ALU flag results:

| Operation | Result | Flags |
|---|---|---|
| **Add 1 + 0** | **1** | **Z=0, N=0, C=0, V=0** |
| **Subtract 1 - 1** | **0** | **Z=1, N=0, C=1, V=0** |

Define CMSIS.

Edit

Delete

CMSIS (Cortex Microcontroller Software Interface Standard) is a common software framework developed by ARM in collaboration with microcontroller vendors to provide standardized software interfaces for Cortex-M processors and microcontroller products.

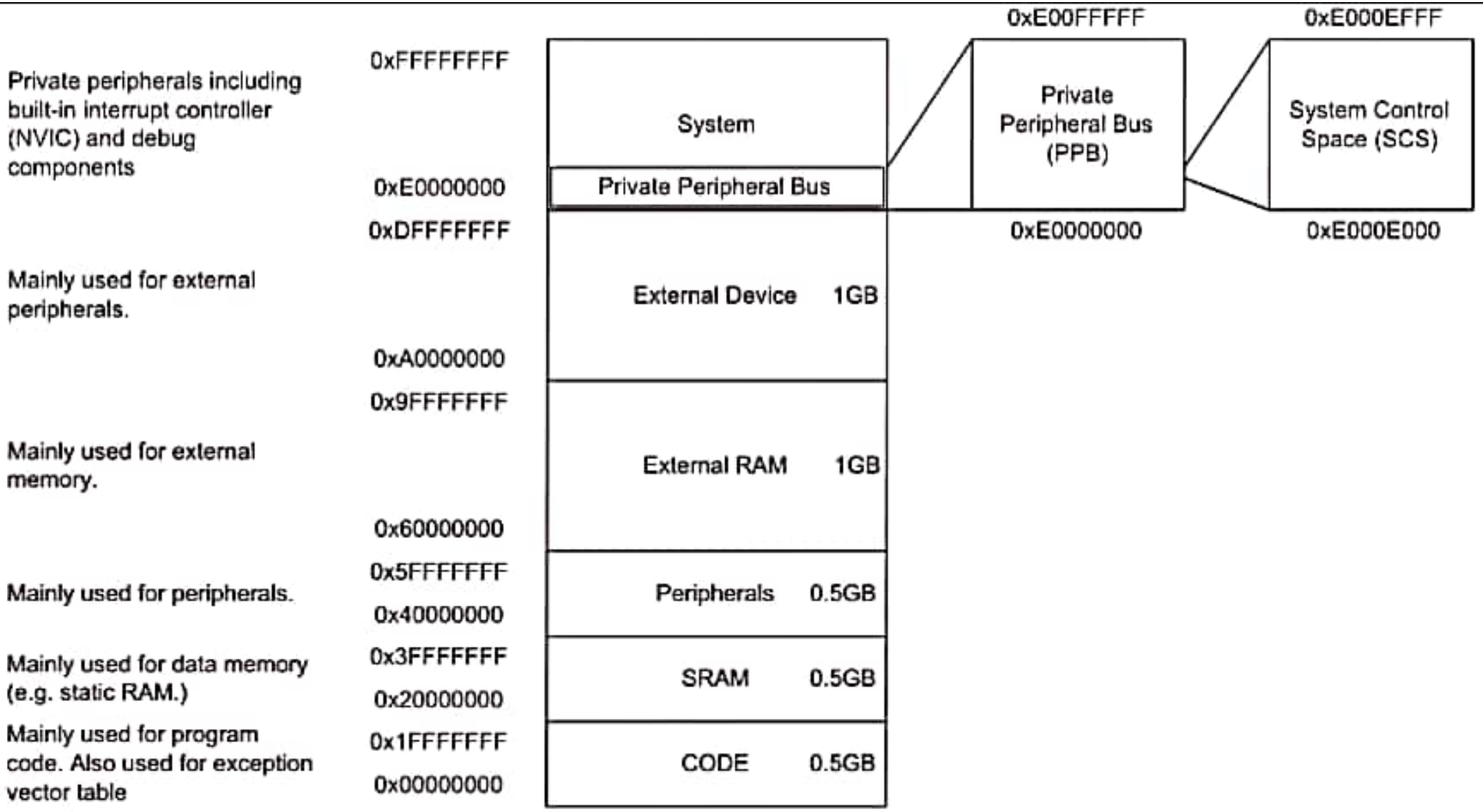



**Figure 3.8:**
Memory map.

List the possible results and flags for the ALU operations given in the examples.

---

- 0x70000000 + 0x70000000 = Result: 0xE0000000, N = 1, Z = 0, C = 0, V = 1
- 0x90000000 + 0x90000000 = Result: 0x20000000, N= 0, Z = 0, C = 1, V = 1
- 0x80000000 + 0x80000000 = Result: 0x00000000, N = 0, Z = 1, C = 1, V = 1
- 0x00001234 - 0x00001000 = Result: 0x00000234, N = 0, Z = 0, C = 1, V = 0
- 0x00000004 - 0x00000005 = Result: 0xFFFFFFFF, N = 1, Z = 0, C = 0, V = 0
- 0xFFFFFFFF - 0xFFFFFFFC = Result: 0x00000003, N = 0, Z = 0, C = 1, V = 0
- 0x80000005 - 0x80000004 = Result: 0x00000001, N = 0, Z = 0, C = 1, V = 0
- 0x70000000 - 0xF0000000 = Result: 0x80000000, N = 1, Z = 0, C = 0, V = 1
- 0xA0000000 - 0xA0000000 = Result: 0x00000000, N = 0, Z = 1, C = 1, V = 0

List the pros and cons of using assembly language for programming.

---

**Pros:**

- Allows direct control to each instruction step and all memory operations
- Allows direct access to instructions that cannot be generated with C

**Cons:**

- Takes longer time to learn
- Difficult to manage data structure
- Less portable (syntax of assembly language in different tool chains can be different)

What is the primary purpose of the Cortex-M0 processor Debug state?

- Halting the processor stops instruction execution and enters the Debug state.
- This allows the debugger to access or change processor register values.
- The debugger can also access system memory locations.

The Debug state provides vital access for debugging purposes.

Explain the components and purpose of the combined Program Status Register (xPSR).

Edit

Delete

The combined Program Status Register (xPSR) is composed of:

- Application PSR (APSR)
- Interrupt PSR (IPSR)
- Execution PSR (EPSR)

APSR contains ALU flags: N (negative flag), Z (zero flag), C (carry or borrow flag), and V (overflow flag). IPSR contains the current executing interrupt service routine (ISR) number. EPSR contains the T-bit indicating the processor's Thumb state. These registers provide essential program execution information and ALU flags for controlling conditional branches and debugging.

List the debug features of the Cortex-M0 processor.

---

- Halt mode debug - Allows the processor activity to stop completely so that register values can be accessed and modified.
- CoreSight technology - Allows memories and peripherals to be accessed from the debugger without halting the processor.
- JTAG and serial wire debug connections - The serial wire debug protocol can handle the same debug features as JTAG, requiring only two wires.
- Configurable number of hardware breakpoints (0 to 4) and watchpoints (0 to 2).
- Breakpoint instruction support for unlimited software breakpoints.
- All debug features can be omitted by chip vendors for minimum size implementations.

Explain the benefits of consistent instruction set and programmer's model among Cortex-M processors.

---

The consistency of instruction set and programmer's model among Cortex-M processors provides the following benefits:

1. It allows better software portability, as C programs can often be transferred between these processors without changes.
2. The debugger and development tool chains can support multiple processors easily due to the similarities.
3. The consistent architecture enables binary images from one processor (e.g., Cortex-M0 or Cortex-M1) to run on another processor (e.g., Cortex-M3) due to upward compatibility.
4. It facilitates easier task migration for embedded programmers between different projects without encountering a steep learning curve.

Explain the Cortex-M0 processor modes and states.

---

The Cortex-M0 processor has two operation modes and two states:

- **Operation Modes:**
  1. Thread Mode: Executes normal code.
  2. Handler Mode: Executes exception handler.
- **States:**
  1. Thumb State: The normal state for running Thumb/Thumb-2 instructions.
  2. Debug State: For debugging purposes, where instruction execution can be halted to allow debugger access to processor registers and system memory.

By default, the processor runs in Thumb state and Thread mode upon power-up.

List the files included in CMSIS and their descriptions.

---

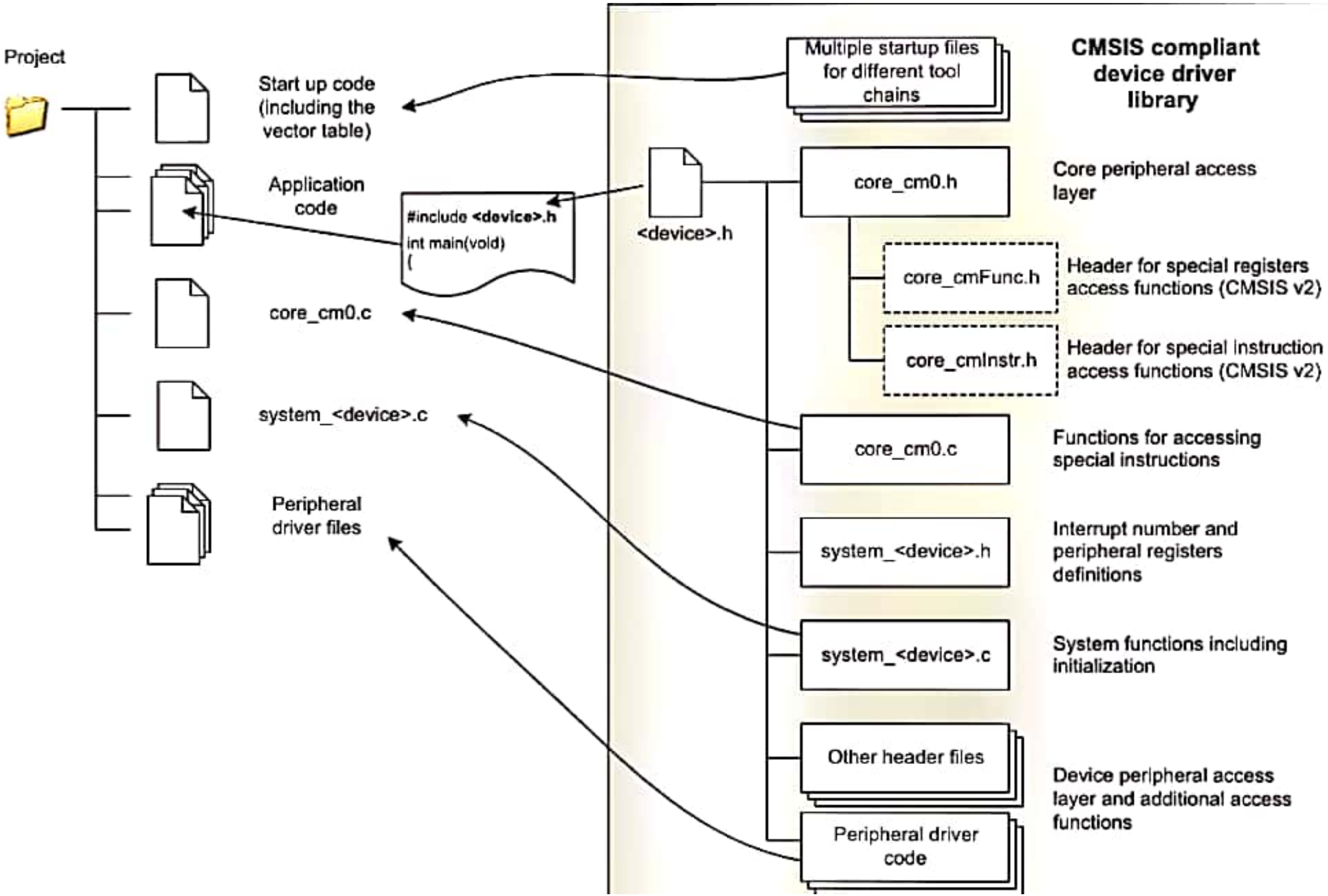
1. **.h:** Provided by the microcontroller vendor, includes other header files, defines device-specific exception types, peripheral registers, and address definitions.
2. **core_cm0.h:** Contains definitions of processor peripherals like NVIC, System Tick Timer, and SCB, along with core access functions. In CMSIS v2, it is split into multiple files.
3. **core_cm0.c:** Provides intrinsic, compiler-independent functions.
4. **Startup Code:** Tools-specific, includes vector table and dummy definitions for system exception handlers. From CMSIS v1.30, it executes SystemInit() before branching to C startup code.
5. **system_.h:** Header file for functions implemented in system_.c.
6. **system_.c:** Implements system initialization, defines SystemCoreClock variable and SystemCoreClockUpdate() function.
7. **Additional Files:** Provides peripheral control code and other helper functions.

Differentiate between APB protocol and AHB protocol.

---

| APB Protocol (Advanced Peripheral Bus) | AHB Protocol (Advanced High-performance Bus) |
|---|---|
| Handles all transfers as 32-bit words. | Supports various transfer sizes. |
| Commonly used for connecting peripherals. | Can be used for connecting different types of devices to the processor bus. |

- **APB Protocol (Advanced Peripheral Bus)**: The APB protocol is specialized for interfacing with low-bandwidth and low-power peripherals. It uses a simpler signal interface which is optimized for minimal communication overhead. All data transfers are performed using 32-bit words.
- **AHB Protocol (Advanced High-performance Bus)**: The AHB protocol is designed for higher performance requirements, accommodating a wider range of data transfer sizes. It provides more complex operations for burst transfer, split transactions, and supports multiple masters which makes it better for connecting high-speed, high-throughput devices.



Describe the techniques used to achieve the low power consumption target of Cortex-M0 processors.

---

ARM achieved the low power consumption target of Cortex-M0 processors through:

1. Small gate count: Various design techniques and optimizations were used to minimize circuit size, achieving a gate count as low as 12k gates.
2. High efficiency: The efficient architecture and optimized hardware implementation allow the processor to run at lower clock frequencies while maintaining performance.
3. Low-power features: The processor includes sleep modes, sleep-on-exit functionality, and the Wakeup Interrupt Controller (WIC).
4. Logic cell enhancement: Introduction of the Ultra Low Leakage (ULL) logic cell library to reduce leakage current and support state retention.

## MSP vs PSP

| Main Stack Pointer (MSP) | Process Stack Pointer (PSP) |
| --- | --- |
| Used by default after reset | Not used by default; must be explicitly set |
| Used in Handler mode (during exception handling) | Used in Thread mode (when not handling exceptions) |
| CONTROL[1] = 0 | CONTROL[1] = 1 |
| Default visible stack pointer | Only visible when explicitly selected |
| Loaded from the first 32-bit word of the vector table | Undefined; must be initialized by software |
| Used in many simple applications and by the OS kernel | Typically used by application processes in an OS environment |
| Accessed using "R13" or "SP" | Accessed using special register names "PSP" |

What are the key system features of the ARM Cortex-M0 processor?

---

- Thumb instruction set offering high efficiency and code density
- High performance, achieving up to 0.9 DMIPS/MHz
- Built-in Nested Vectored Interrupt Controller (NVIC)
- Interrupts with four different programmable priority levels
- Deterministic exception response timing
- Nonmaskable interrupt (NMI) input
- Architectural predefined memory map
- Easy to use and C-friendly