Vol 56, No 1—January 2022

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

AnalogDialogue

# How to Accelerate Peripheral Monitoring in Low Power Wearables with DMA

**Brandon Hurst**, Hardware and Embedded Firmware Engineer

## Abstract

This article explains the use cases, advantages, and disadvantages of utilizing direct memory access (DMA) in embedded systems programming. The article describes how DMA interacts with peripheral and memory modules for more efficient operation of CPUs. The article will also introduce the reader to different DMA bus access architectures, and the advantages of each.

One task that is common for embedded systems is managing external input. Managing input can put a lot of unnecessary computational strain on the processor, causing longer periods in active power modes and slow response times. For optimizing power, preserving quick responses to events, and managing large continuous data transfers, a microcontroller with direct memory access (DMA) may offer the best solution.

## Direct Memory Access (DMA)

In system applications involving peripherals, there are many points at which a microprocessor can become bottlenecked. For instance, when managing an ADC that is constantly sending data, a processor can be interrupted so often that it struggles to accomplish other tasks. DMA is a method of moving data and minimizing processor involvement in large or fast data transactions. You can think of the DMA controller as a coprocessor whose sole purpose is to interact with memory and peripherals. This allows the main processor to successfully manage a greedy peripheral, focus on another task, or even go to sleep and conserve power while data transactions happen in the background. For example, on Arm® architectures, a DMA module can operate during LP2 (sleep) or LP3 (run) modes. This can give a distinct advantage in applications that require extended battery life, such as wearable sensor hubs and smart watches.

## Advantages and Drawbacks

DMA is useful in many digital systems, and sometimes it is even required to manage large amounts of bus traffic. It has been used in network cards, graphics cards, and even some of the original IBM PCs. That being said, incorporating DMA into a design does have some trade-offs.

**Table 1. Advantages of Using DMA**

| Advantages of Using DMA | |
|---|---|
| CPU Time | DMA minimizes the need for processor execution and interrupts, decreasing the required CPU time for data transactions. |
| Power Consumption | Using DMA can yield opportunities to minimize power consumption if it allows the processor to sleep during DMA transfers. |
| Parallel Operation | Depending on the architectural details of the system bus, the processor may be able to execute other operations while peripheral transactions are taking place. |

**Table 2. Disadvantages of Using DMA**

| Disadvantages of Using DMA | |
|---|---|
| Cost | Incorporating a system with DMA requires a DMA controller, and this can make a system more expensive. |
| Complexity | While DMA can reduce the frequency of interrupts, it can increase the size and complexity of application firmware. |
| Platform Dependence | DMA controllers have differing internal architectures between and within manufacturers and can have different behavior depending on their native bus access schemes. |
| Cache Incoherency | DMA transactions can cause logical errors to occur by writing to a cached layer of the memory hierarchy. This can be solved by using cache-coherent system architectures or by invalidating cache storage upon DMA completion. |

## Bus Access and CPU Cycles

While DMA controllers can be incredibly effective at conserving power or speeding up embedded systems, their implementation is not heavily standardized. There are multiple schemes for making sure that internal bus access is not granted simultaneously with the CPU. The goal of the bus access scheme is primarily to avoid concurrent access to the same memory locations, which can lead to cache incoherency and logical errors. A single DMA controller will usually be configured to employ one of these schemes, since different hardware or firmware control may be required to use each of them. The bus access schemes used by most DMA controllers are burst, cycle-stealing, and transparent DMA.

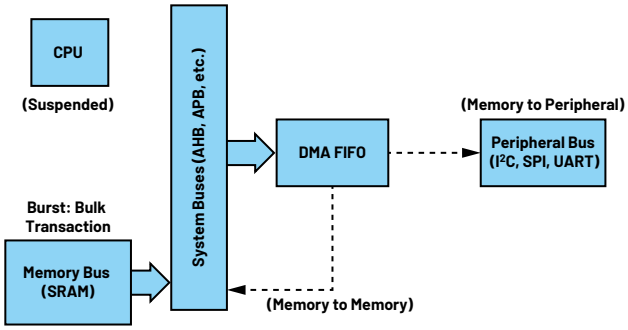


*Figure 1. An architectural diagram of burst DMA during DMA operations.*

Burst DMA occurs through infrequent large bursts, where the DMA controller sends as much data to the destination buffer as the buffer can hold. The DMA controller blocks CPU operation for a very short period to move a large chunk of memory, and then relinquishes the bus back to the main CPU, repeating until the transfer is complete. Burst DMA is generally considered the fastest type.

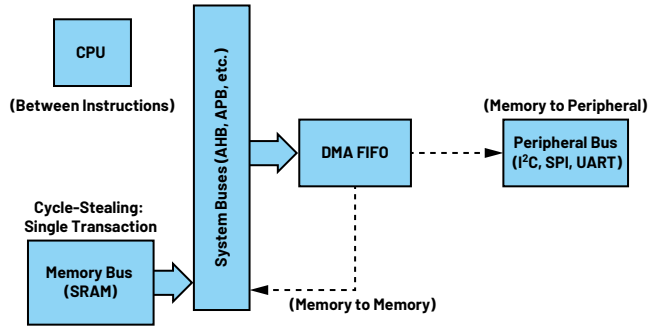


*Figure 2. Cycle-stealing DMA during DMA operations occurs between two CPU cycles.*

Conversely, single byte transfer or cycle-stealing DMA takes a cue from the CPU and only carries out operations between CPU instructions. It inserts a single operation between two CPU cycles, and thus is in effect "stealing" CPU time. Due to the limitation of executing one operation at a time, it is generally slower than burst DMA.

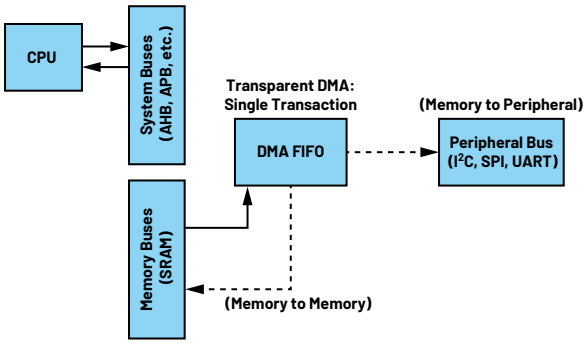


*Figure 3. Transparent DMA during DMA operations occurs while the processor works on tasks that do not access the data or address buses.*

Transparent DMA can only execute a single operation at a time, but it must also wait for the processor to execute instructions in which it yields access to the desired data or address buses. Extra logic is required to verify this access restriction, and this type of DMA is generally the slowest. Transparent DMA may be advantageous in applications where one has extra processing to do that does not require access to the memory buses. The advantage in this case would be the elimination of throttling the CPU, since the processor does not have to stop operating completely.

### Table 3. Summary of DMA Types and Their Pros/Cons

| Type of DMA | Pros | Cons |
|---|---|---|
| Burst DMA | Fastest type of DMA | Relatively long periods of CPU idle time |
| Cycle-Stealing DMA | CPU is not idle for long contiguous periods | Slower than burst DMA |
| Transparent DMA | No throttling of CPU use needed | Slowest form of DMA |

## Example of a Burst DMA Architecture

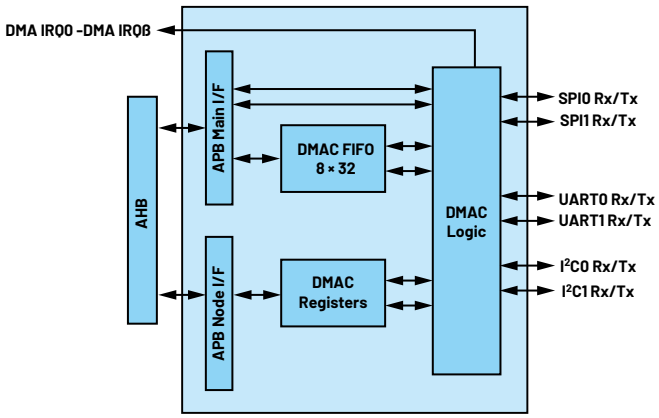


*Figure 4. An architectural diagram of the DMA controller on the MAX32660.*

An example of a burst DMA controller can be found on the MAX32660 (see Figure 4). The upper path corresponds to data flow, and the lower path represents control/status flow between the advanced high performance bus (AHB) and the DMA logic. The DMA controller can behave as a buffer interface between the AHB and memory or peripheral modules, depending on how it is configured. DMA logic sits between the DMA buffer and each peripheral to independently manage each unique peripheral bus during transactions. A DMA operation can move up to 32 bytes at a time, provided the source/destination buffers can contain this much data. The buffer can hold up to 16 MB and is configurable to transmit or receive I²C, SPI, I²S, and UART in addition to internal memory transfers. Programming DMA control may vary slightly between protocols, but the peripheral transactions are managed exclusively by the DMA controller. An arbiter module controls the bus access restrictions between the four DMA channels and the CPU, granting requests according to a priority system.

## Modern DMA Options

In summary, DMA is a critical feature for modern embedded systems that manage an abundance of sensors and require high throughput, efficiency, and low power operation. It behaves like a coprocessor dedicated exclusively to memory and peripheral bus transactions.

Using DMA is imperative for many applications to minimize power consumption and lighten processor loads. For example, health and wearable devices handle large amounts of data throughput, but they also must conserve as much battery charge as possible, all while handling sensitive data. Analog Devices offers fast burst DMA architectures on microcontrollers well-equipped for low power wearable designs, such as the MAX32660 and MAX32670. In addition, DARWIN Arm microcontrollers such as the MAX32666 are built for wearable and IoT applications with integrated Bluetooth® 5. These devices have two 8-channel burst DMA controllers with integrated support for event-based transactions. They even feature best-in-class security hardware with a secure bootloader and trust protection unit (TPU) for accelerating ECDSA, SHA-2, and AES encryption. From the early IBM PCs to network cards, and now to secure, low power wearable and IoT devices, DMA is an essential feature of modern digital systems.


## About the Author

Brandon Hurst is a hardware and embedded firmware engineer working with the Training and Technical Services Group for Maxim Integrated, now part of Analog Devices. He graduated with a bachelor's in electrical engineering from Cal Poly, San Luis Obispo and joined Maxim in January 2021. Previously, Brandon did an internship with both Maxim's TTS team and the Product Safety Engineering team at Apple, Inc. He can be reached at brandon.hurst@analog.com.