

Memory

How to run examples

```
cd /esp/esp-idf/tools/unit-test-app  
idf.py -T driver flahs monitor
```

RAM of ESP32

ESP32 RAM:

- DRAM: Data RAM

- IRAM: Instruction RAM, 32-bit aligned

Add a PSRAM

PSRAM—Pseudo static random access memory, 伪静态随机存储器；

malloc

```
void *malloc(size_t size);
```

说明：**malloc** 向系统申请分配指定**size**个字节(**Byte**)的内存空间，并返回一个指向它的指针。返回类型是* **void** 类型。***void** 表示未确定类型的指针。C,C++规定，**void*** 类型可以强制转换为任何其它类型的指针。具体内容如下：

malloc returns a void pointer to the allocated space, or NULL if there is insufficient memory available. To return a pointer to a type other than void, use a type cast on the return value. The storage space pointed to by the return value is guaranteed to be suitably aligned for storage of any type of object. If size is 0, malloc allocates a zero-length item in the heap and returns a valid pointer to that item. Always check the return from malloc, even if the amount of memory requested is small.

```
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
  
int main(){  
    char *str;  
    /*initial memory allocate*/  
    str = (char *)malloc(15);  
    strcpy(str,"runoob");  
    printf("String = %s, Address = %u\n", str, str);  
    /*re-allocate the memory*/  
}
```

TODO: diff between realloc, malloc, calloc

Direct Memory Access

DMA用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。它允许不同速度的硬件装置来沟通，而不需要依于CPU的大量中断负载。通过DMA数据可以快速地移动。这就节省了CPU的资源来做其他操作。否则，CPU需要从来源把每一片段的资料复制到暂存器，然后把它们再次写回到新的地方。在这个时间中，CPU对于其他的工作来说就无法使用。

每个DMA传送由3个操作组成：

1. 从外设数据寄存器或者从DMA_CMARx寄存器指定地址的存储器单元执行加载操作。
2. 存数据到外设数据寄存器或者存数据到DMA_CMARx寄存器指定地址的存储器单元。
3. 执行一次DMA_CNDTRx寄存器的递减操作。该寄存器包含未完成的操作数目。

DMA可用于主要的外设： SPI, I2C, USART, TIMx,

ESP32 中有 13 个外设都具有 DMA 功能,这 13 个外设是: **UART0**、**UART1**、**UART2**、**SPI1**、**SPI2**、**SPI3**、**I2S0**、**I2S1**、**SDIO slave**、**SD/MMC host**、**EMAC**、**BT** 和 **Wi-Fi**。

DMA控制器具有以下几个特点

- AHB 总线架构
- 支持半双工和全双工收发数据
- 数据传输以 **字节** 为单位,传输数据量可软件编程
- 支持 **4-beat burst** 传输
- **328 KB DMA 地址空间**
- 通过 DMA 实现高速数据传输