

Placing Static Library in External Flash

To Create static library and use it the project - KBA:-

<https://community.infineon.com/t5/Blogs/Enhance-ModusToolbox-Adding-Custom-Libraries-for-Better-Efficiency/ba-p/436547?lightbox-message-images-436547=62678iAC86DF94E3E94600>

Before Placing static library into external flash :-

After following all the steps and got output

```
0x10008724      _write
.text._read     0x10008758      0x3c C:/Users/sidra/mtw/Stat
0x10008758      _read
.text.cy_retarget_io_init_fc
0x10008794      0x50 C:/Users/sidra/mtw/Stat
0x10008794      cy_retarget_io_init
.text.add       0x100087e4      0x1e add/libadd.a(add.o)
0x100087e4      add
*fill*         0x10008802      0x2
.text          0x10008804      0xa0 c:/users/sidra/modustoo
0x10008804      __aeabi_ldivmod
.text          0x100088a4      0x30 c:/users/sidra/modustoo
```

I don't know why I get 0x2 because I gave add(2,3) and in serial terminal I got 5.

I changed aff(2,3) to add(5,3) after building and programming output is 8 and details in the map file is the same

```
.text._read     0x10008758      0x3c C:/Users/sidra/mtw/Static_library_project/build/A
0x10008758      _read
.text.cy_retarget_io_init_fc
0x10008794      0x50 C:/Users/sidra/mtw/Static_library_project/build/A
0x10008794      cy_retarget_io_init_fc
.text.add       0x100087e4      0x1e add/libadd.a(add.o)
0x100087e4      add
*fill*         0x10008802      0x2
.text          0x10008804      0xa0 c:/users/sidra/modustoolbox/tools_3.0/gcc/bin/..
0x10008804      __aeabi_ldivmod
.text          0x100088a4      0x30 c:/users/sidra/modustoolbox/tools_3.0/gcc/bin/..
```

At this point I haven't placed static library in the external flash so cy_xip region is

```
*(.cy_sflash_public_key)

.cy_toc_part2
*(.cy_toc_part2)

.cy_rtoc_part2
*(.cy_rtoc_part2)

cy_xip          0x18000000      0x0      __cy_xip_start = .
*(.cy_xip)      0x18000000
                0x18000000      __cy_xip_end = .

.cy_efuse
*(.cy_efuse)
```

After building the project I will not see cy_xip region because we haven't used it yet

Calculating memory consumption: CY8C624ABZI-S2D44 GCC_ARM

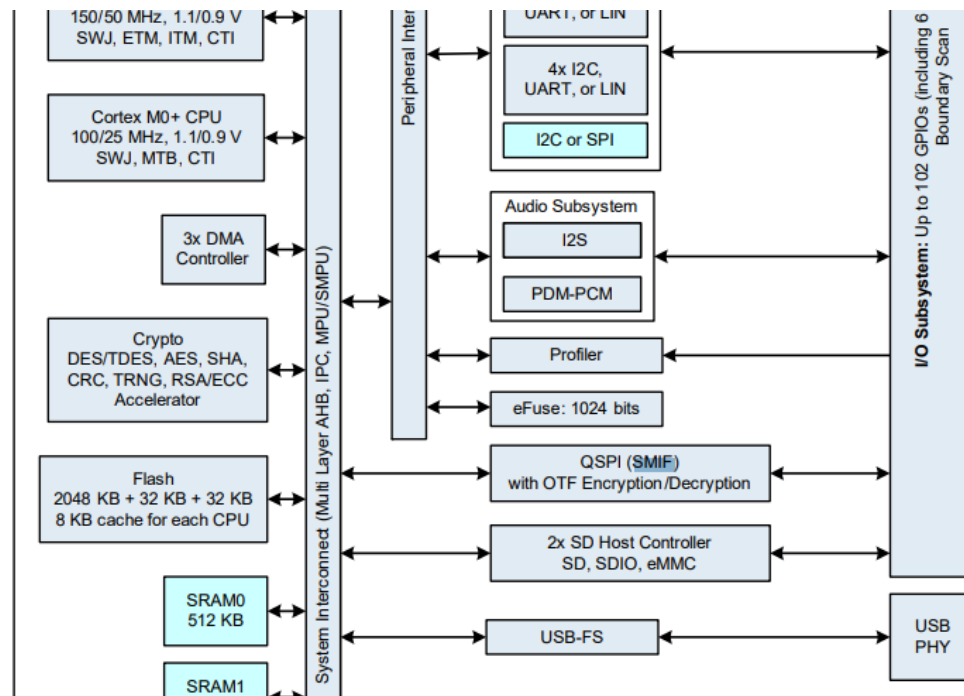
Section Name	Address	Size
.cy_m0p_image	0x10000000	6352
.text	0x10002000	46056
.ARM.exidx	0x1000d3e8	8
.copy.table	0x1000d3f0	24
.zero.table	0x1000d408	8
.data	0x080022e0	1640
.cy_sharedmem	0x08002948	8
.noinit	0x08002950	228
.bss	0x08002a34	1728
.heap	0x080030f8	1029896

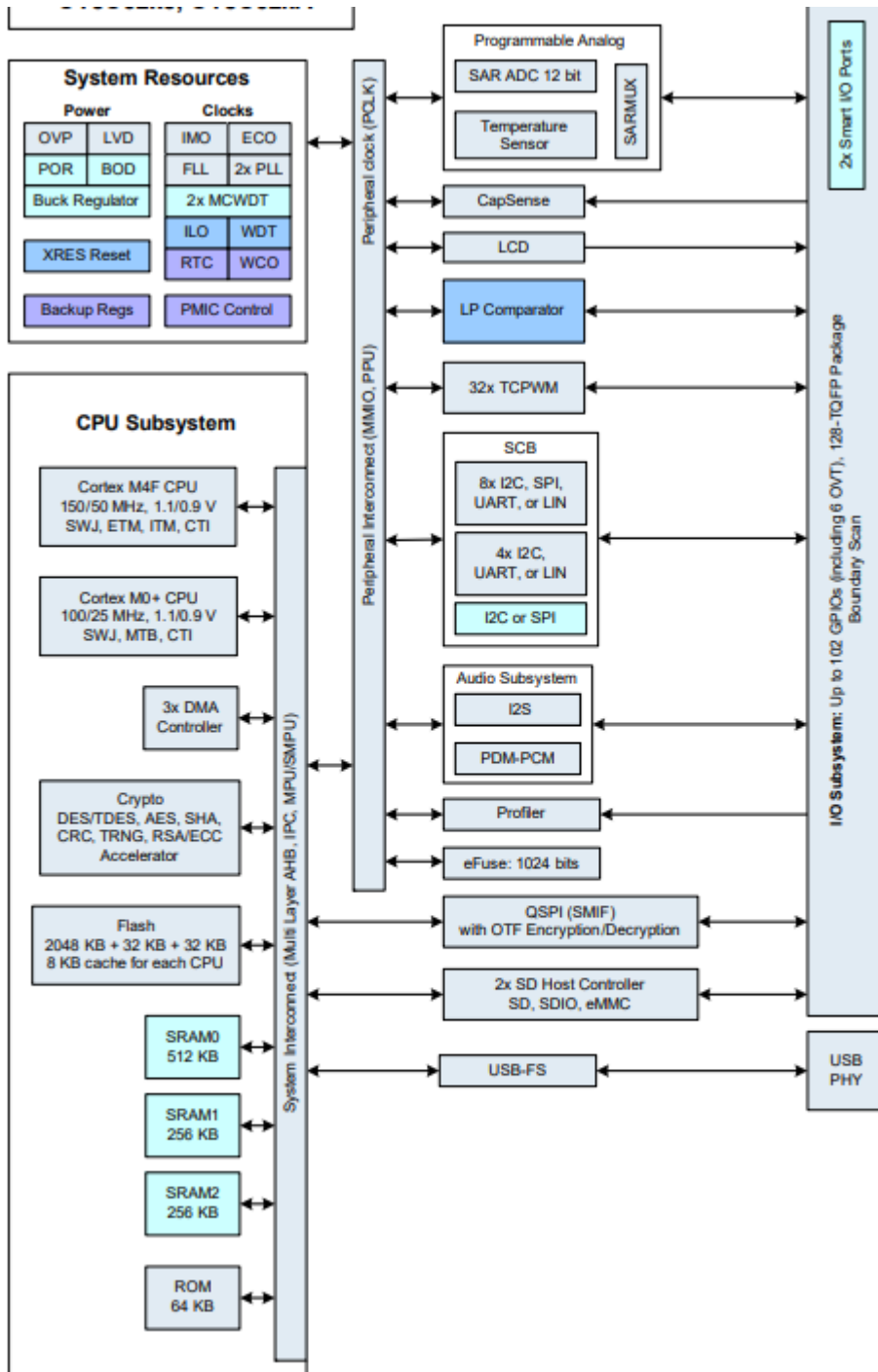
Total Internal Flash (Available) 2097152
Total Internal Flash (Utilized) 55944

17:11:57 Build Finished. 0 errors, 0 warnings. (took 19s.672ms)

From Datasheet :-

Nor flash size is 512Mbits but XIP region is 0x18000000 to 0x80000000 is 128MB





Memory Map

Both CPUs have a fixed address map, with shared access to memory and peripherals. The 32-bit (4 GB) address space is divided into the Arm-defined regions shown in [Table 3](#). Note that code can be executed from the Code and External RAM regions.

Table 3. Address Map for CM4 and CM0+

Address Range	Name	Use
0x0000 0000 – 0x1FFF FFFF	Code	Program code region. Data can also be placed here. It includes the exception vector table, which starts at address 0.
0x2000 0000 – 0x3FFF FFFF	SRAM	Data region. This region is not supported in PSoC 6.
0x4000 0000 – 0x5FFF FFFF	Peripheral	All peripheral registers. Code cannot be executed from this region. CM4 bit-band in this region is not supported in PSoC 6.
0x6000 0000 – 0x9FFF FFFF	External RAM	SMIF or Quad SPI, (see the Quad-SPI/Serial Memory Interface (SMIF) section). Code can be executed from this region.
0xA000 0000 – 0xDFFF FFFF	External Device	Not used.
0xE000 0000 – 0xE00F FFFF	Private Peripheral Bus	Provides access to peripheral registers within the CPU core.
0xE010 0A000 – 0xFFFF FFFF	Device	Device-specific system registers.

The device memory map shown in [Table 4](#) applies to both CPUs. That is, the CPUs share access to all PSoC 6 MCU memory and peripheral registers.

Table 4. Internal Memory Address Map for CM4 and CM0+

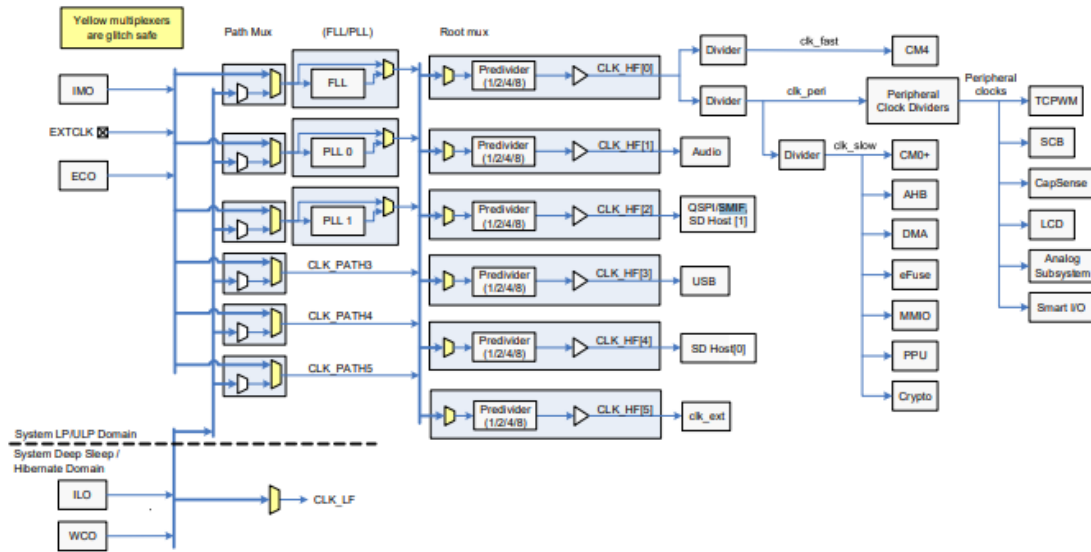
Address Range	Memory Type	Size
0x0000 0000 – 0x0000 FFFF	ROM	64 KB
0x0800 0000 – 0x080F FFFF	SRAM	Up to 1 MB
0x1000 0000 – 0x101F FFFF	Application flash	Up to 2 MB
0x1400 0000 – 0x1400 7FFF	Auxiliary flash, can be used for EEPROM emulation	32 KB
0x1600 0000 – 0x1600 7FFF	Supervisory flash	32 KB

Note that PSoC 6 SRAM is located in the Arm Code region for both CPUs (see [Table 3](#)). There is no physical memory located in the CPUs' Arm SRAM regions.

Quad-SPI (QSPI)/Serial Memory Interface (SMIF)

- Execute-In-Place (XIP) from external quad SPI flash
- On-the-fly encryption and decryption
- 4-KB cache for greater XIP performance with lower power
- Supports single, dual, quad, dual-quad, and octal interfaces with throughput up to 640 Mbps

Figure 3. Clocking Diagram



Quad-SPI/Serial Memory Interface (SMIF)

A serial memory interface is provided, running at up to 80 MHz.

It supports single, dual, quad, dual-quad and octal SPI configurations, and supports up to four external memory devices.

It supports two modes of operation:

- Memory-mapped I/O (MMIO), a command mode interface that provides data access via registers and FIFOs
- Execute in Place (XIP), in which AHB reads and writes are directly translated to SPI read and write transfers.

In XIP mode, the external memory is mapped into the PSoC 6 MCU internal address space, enabling code execution directly from the external memory. To improve performance, a 4-KB cache is included. XIP mode also supports AES-128 on-the-fly encryption and decryption, enabling secured storage and access of code and data in the external memory

From Board user guide :-

1.2 Board Details

- 512-Mbit external Quad SPI NOR Flash that provides a fast, expandable memory for data and Code

Cypress 512-Mbit serial NOR flash memory (S25HL512T, U11): The S25HL512T NOR flash of 512Mb capacity is connected to the serial memory interface (SMIF) of the PSoC 6 MCU. The NOR flash can be used for both data and code memory with execute-in-place (XIP) support and encryption.

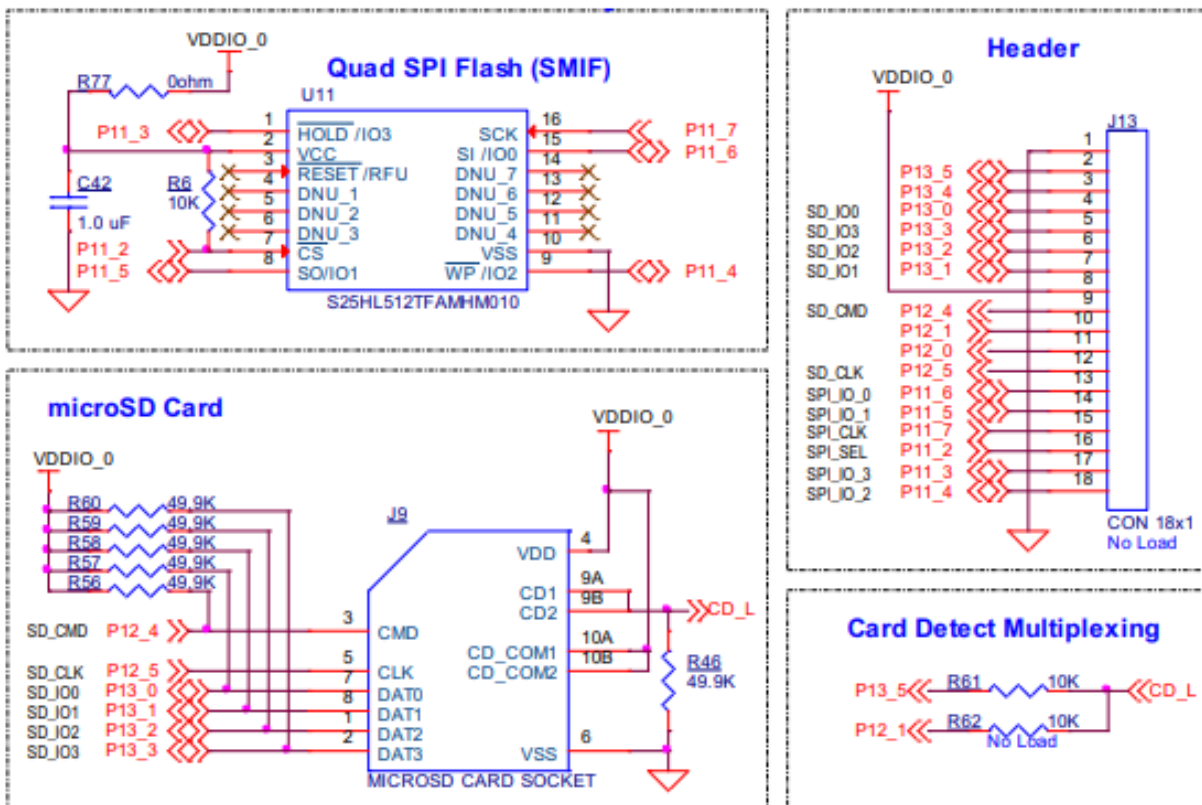
A.2.9 Cypress Quad SPI NOR Flash and microSD card

The board has a Cypress NOR Flash memory (S25HL512TFAMHM010) of 512 Mbit capacity. The NOR Flash is connected to the serial memory interface (SMIF) of the PSoC 6 MCU. The NOR Flash device can be used for both data and code memory with execute-in-place (XIP) support and encryption.

The board contains a slot to insert a microSD card (see Figure A-10), which can be accessed through SDHC interface.

This section can be broken apart from the PSoC 6 MCU section at the built-in perforated edge between J1 and J13. To connect them back, use right angled male-to-female connectors between J13 and J1 (J1.11 to J1.28).

Figure A-10. Schematics of Quad SPI Flash and microSD card holder



From Architecture TRM :-

29. Serial Memory Interface (SMIF)

The SMIF block implements a single-SPI, dual-SPI, quad-SPI, or octal-SPI communication to interface with external memory chips. The SMIF block's primary use case is to set up the external memory and have it mapped to the PSoC 6 MCU memoryspace using the hardware. This mode of operation, called the XIP mode, allows the bus masters in the PSoC 6 MCU to directly interact with the SMIF for memory access to an external memory location.

A graphical interface is provided with the ModusToolbox for configuring the QSPI (SMIF) block. For more information, see the

[ModusToolbox QSPI Configurator Guide.](#)

29.1 Features

The Serial Memory Interface (SMIF) block provides a master interface to serial memory devices that supports the following functionality.

- Interfacing up to four memory devices (slaves) at a time
- SPI protocol
- ☐ SPI mode 0: clock polarity (CPOL) and clock phase (CPHA) are both '0'
- ☐ Support for single, dual, quad, and octal SPI protocols
- ☐ Support for dual-quad SPI mode: the use of two quad SPI memory devices to increase data bandwidth for SPI read and write transfers
- ☐ Support for configurable MISO sampling time and programmable receiver clock
- **Support for device capacities in the range of 64 KB to 128 MB**
- eExecute In Place (XIP) enables mapping the external memory into an internal memory address
- Command mode enables using the SMIF block as a simple communication hardware
- Supports a 4-KB read cache in memory mapped (XIP) mode
- Supports on-the-fly 128-bit encryption and decryption

After placing static library into the external flash and invoking function within it and executing using XIP.

Steps: -

1) We should place the below lines in the starting of the SECTIONS of linker script

```
/*Placing a static library in external flash */
.cy_xip_code :
{
    KEEP(*(.cy_xip_code))
    add/libadd.a:(.text .text* .rodata .rodata*)
} > xip
```

2) Remove ./add from INCLUDES and -L.add yadd from LDFLAGS and add CY_ENABLE_XIP_PROGRAM in DIFINES.

3)Open library manager and add serial flash library and update

4)Do following changes in main.c

a) Add below header files

#include "cy_serial_flash_qsapi.h"

```
#include "cycfg_qspi_memslot.h"
```

b) Add below macros in macros section

```
#define MEM_SLOT_NUM      (0u)    /* Slot number of the memory to use */
```

```
#define QSPI_BUS_FREQUENCY_HZ (50000000lu)
```

c) Initialize the QSPI block after User LED initialization

```
/* Initialize the QSPI block */  
result = cy_serial_flash_qspi_init(smifMemConfigs[MEM_SLOT_NUM], CYBSP_QSPI_D0,  
CYBSP_QSPI_D1,CYBSP_QSPI_D2, CYBSP_QSPI_D3, NC, NC, NC, NC, CYBSP_QSPI_SCK,  
CYBSP_QSPI_SS, QSPI_BUS_FREQUENCY_HZ);
```

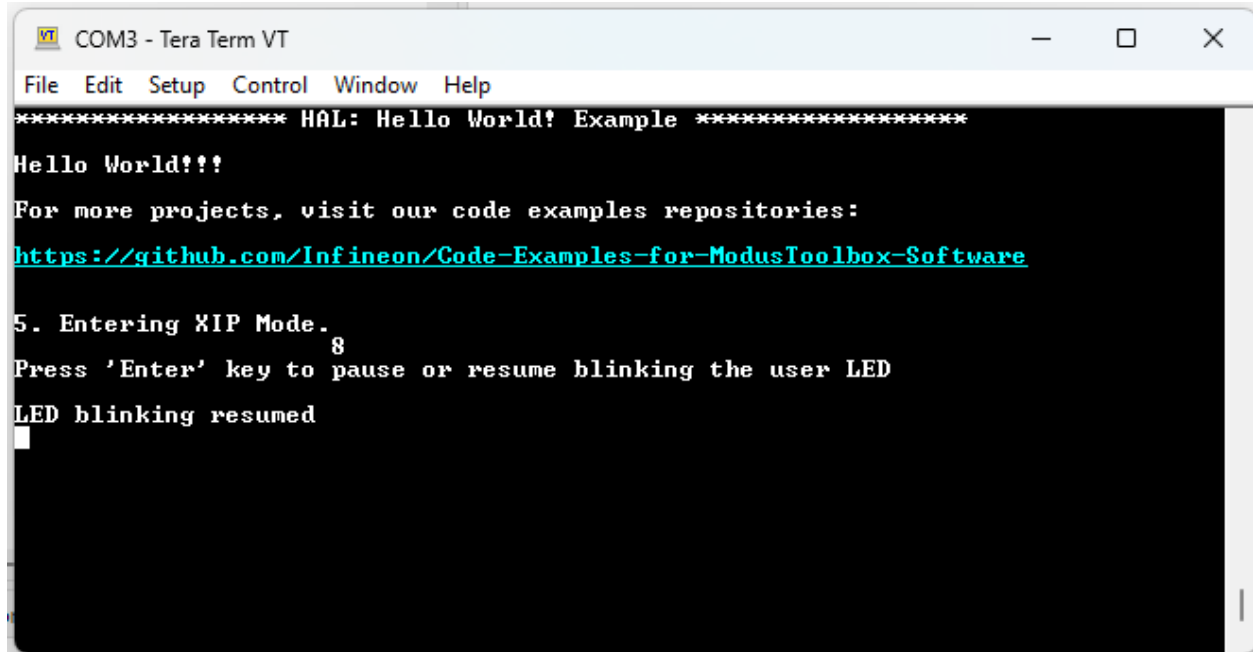
d) enable XIP mode

```
/* Put the device in XIP mode */  
printf("\n5. Entering XIP Mode.\n");  
cy_serial_flash_qspi_enable_xip(true);
```

e) Invoke add function

```
int a=add(5,3);  
printf("%d \r\n",a);
```

Now you can see value of a printing in external flash using xip



```
COM3 - Tera Term VT  
File Edit Setup Control Window Help  
***** HAL: Hello World? Example *****  
Hello World!!!  
For more projects, visit our code examples repositories:  
https://github.com/Infineon/Code-Examples-for-ModusToolbox-Software  
5. Entering XIP Mode.  
8  
Press 'Enter' key to pause or resume blinking the user LED  
LED blinking resumed  
█
```


See in the above screenshot we can't see sum like before placing static library into external flash

```
.text._write 0x1000c528 0x34 C:/Users/sidra/mtw/Static_library_project/buil
               0x1000c528 _write
.text._read 0x1000c55c 0x3c C:/Users/sidra/mtw/Static_library_project/buil
               0x1000c55c _read
.text.cy_retarget_io_init_fc
               0x1000c598 0x50 C:/Users/sidra/mtw/Static_library_project/buil
               0x1000c598 cy_retarget_io_init_fc
.text._value_to_byte_array
               0x1000c5e8 0x24 C:/Users/sidra/mtw/Static_library_project/buil
.text._init_dma
               0x1000c60c 0xa0 C:/Users/sidra/mtw/Static_library_project/buil
.text._read_next_chunk
               0x1000c6ac 0xa4 C:/Users/sidra/mtw/Static_library_project/buil
.text._rx_dma_irq_handler
               0x1000c750 0xec C:/Users/sidra/mtw/Static_library_project/buil
.text._deinit_dma
               0x1000c83c 0x64 C:/Users/sidra/mtw/Static_library_project/buil
.text.cy_serial_flash_qspi_deinit
               0x1000c8a0 0x4c C:/Users/sidra/mtw/Static_library_project/buil
```

And XIP region is present and sum is executing in that memory location

```
LOAD c:/users/sidra/modustoolbox/tools_3.0/gcc/bin/./lib/gcc/arm-non
END GROUP

.cy_xip_code 0x18000000 0x1e
*(.cy_xip_code)
add/libadd.a:(.text .text.* .rodata .rodata*)
.text.add 0x18000000 0x1e add/libadd.a(add.o)
               0x18000000 add

.cy_m0p_image 0x10000000 0x18d0
               0x10000000 . = ALIGN (0x4)
               0x10000000 __cy_m0p_code_start = .
*(.cy_m0p_image)
.cy_m0p_image 0x10000000 0x18d0 C:/Users/sidra/mtw/Static_library
               0x10000000 cy_m0p_image
               0x100018d0 __cy_m0p_code_end = .
               0x00000001 ASSERT ((__cy_m0p_code_end -
```

Here we can see after build we can see cy_xip_code region

Calculating memory consumption: CY8C624ABZI-S2D44 GCC_ARM

Section Name	Address	Size
.cy_xip_code	0x18000000	30
.cy_m0p_image	0x10000000	6352
.text	0x10002000	64144
.ARM.exidx	0x10011a90	8
.copy.table	0x10011a98	24
.zero.table	0x10011ab0	8
.data	0x080022e0	1800
.cy_sharedmem	0x080029e8	8
.noinit	0x080029f0	228
.bss	0x08002ad4	1916
.heap	0x08003250	1029552
PROGBITS	0x16000800	8
.cy_toc_part2	0x16007c00	512

```
Total Internal Flash (Available) 2097152
Total Internal Flash (Utilized) 74192
```

While placing static library in the external memory I took help of below code example: -

<https://github.com/Infineon/mtb-example-psoc6-qspi-xip/tree/master>

Q) given xip addresses (rx) : ORIGIN = 0x18000000, LENGTH = 0x8000000
what will be the its size?

Ans) To determine the size of a memory region given its origin and length, you can directly use the length provided. In this case, the length is given in hexadecimal as 0x8000000.

Hexadecimal 0x8000000 is equivalent to 134217728 in decimal.

Therefore, the size of the memory region is **134,217,728 bytes**.

If you want this size in more common units:

- **Kilobytes (KB):** $134,217,728 \text{ bytes} \div 1024 = 131,072 \text{ KB}$
- **Megabytes (MB):** $131,072 \text{ KB} \div 1024 = 128 \text{ MB}$

So, the memory region is **128 MB** in size.