Feasibility of Em_EEPROM Shared between Bootloader and Bootloadable

Ryan Zhao

FEB. 2020



Feasibility Analysis

Analysis of customer requirement:

If the end customer request is the case like this:

1. Shared Em_EEPROM between bootloader and bootloadable;
2. Em_EEPROM should
3. Agree to use last N rows only in the Flash as Em_EEPROM.

That means we don't need to change the start address of checksum_excluded area, because checksum_excluded area will always be last N rows by default.

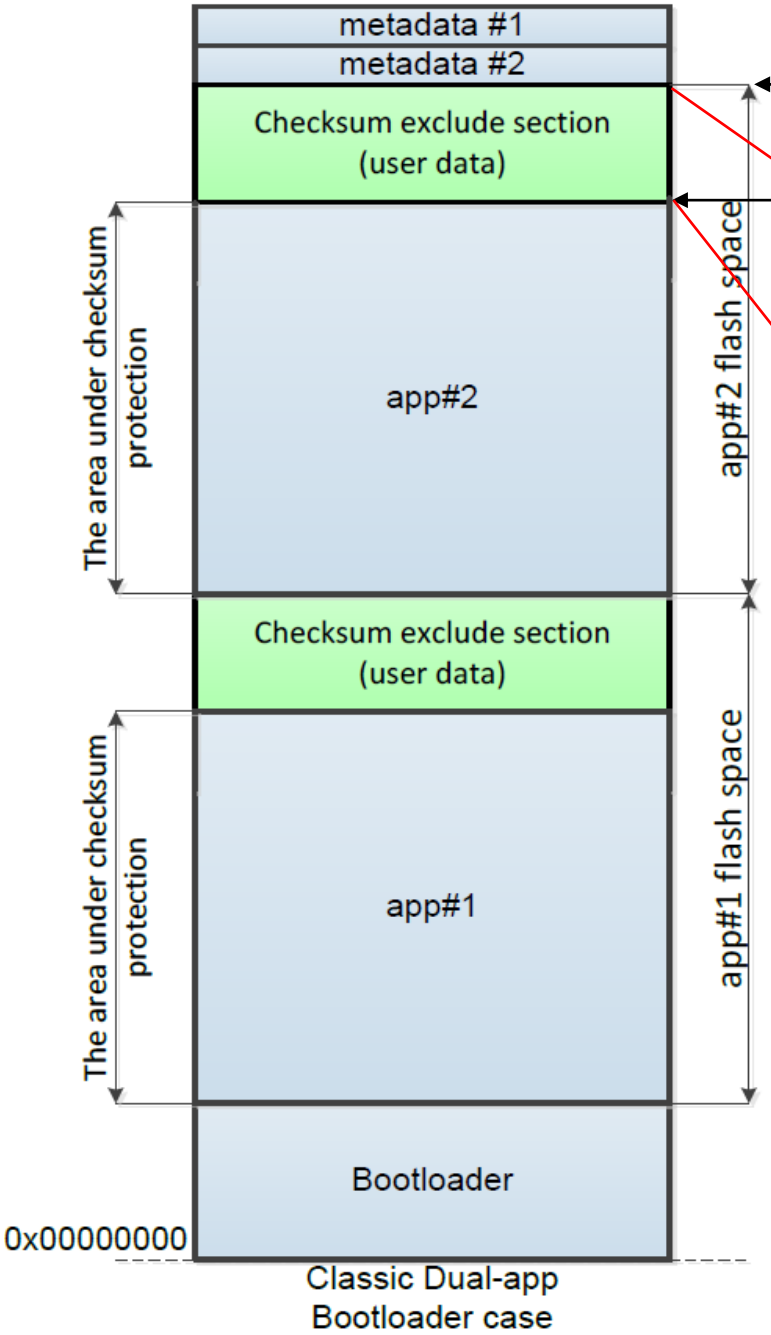
Em_EEPROM(shared) area consideration (Page-3 Figure):

Based on the above case:

1. Metadata is started from address 0xff00 (last 2 rows) for the dual-app bootloader project.
2. There is 1024-Byte from address 0xfb00(8 rows) to metadata area(0xff00)
3. This 1024-Byte belongs to APP2, so set checksum_exclude area of APP2 as 1024 byte.

Test on CY8CKIT-041-41XX

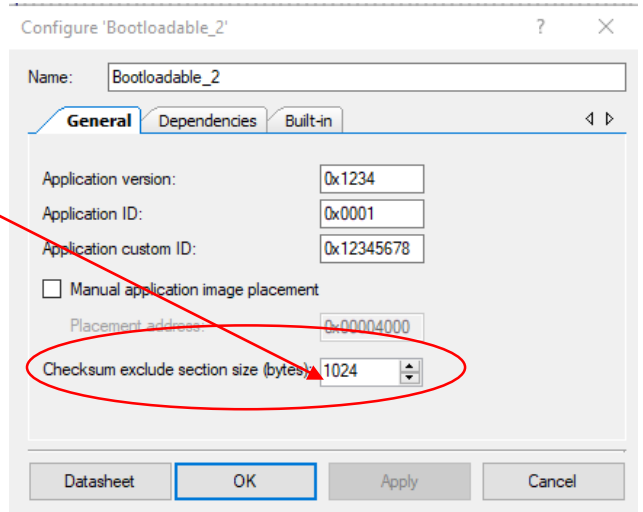
- EEPROM Component v2.20
- Bootloader/Bootloadable v1.60
- PSoC Creator 4.2 Build641



0x0000ff00

0x0000fb00

1024-byte



0x0000ff00

Shared Em_EEPROM By Bootloader and Apps

Either Bootloader or Apps can access the Em_EEPROM:

In bootloader, initialize

```
109     /* Bootloader*/
110     Em_EEPROM_1_Init(0xfb00u);
111     Em_EEPROM_1_Write(0u, array_bootloader, 16u);
```

In App1

```
97      /*App_1*/
98      Em_EEPROM_2_Init(0xfb00u);
99      Em_EEPROM_2_Write(16u, array_app_1, 16u);
```

In App2

```
98      /*App_2*/
99      Em_EEPROM_3_Init(0xfb00u);
100     Em_EEPROM_3_Write(24u, array_app_2, 16u);
```

0x0000fb00

How to specify the start address of the shared Em_EEPROM?

In Em_EEPROM datasheet, it is documented as following.

> For non-PSoC 6 devices, you must statically allocate the memory that will be used for Em_EEPROM storage.
>
> To do this, declare an array in flash aligned to the size of the device flash row. The following is an example of such array declaration for GCC and MDK compilers:

```
const uint8 emEeprom[Em_EEPROM_1_PHYSICAL_SIZE]
  __ALIGNED(CY_FLASH_SIZEOF_ROW) = {0u};
```

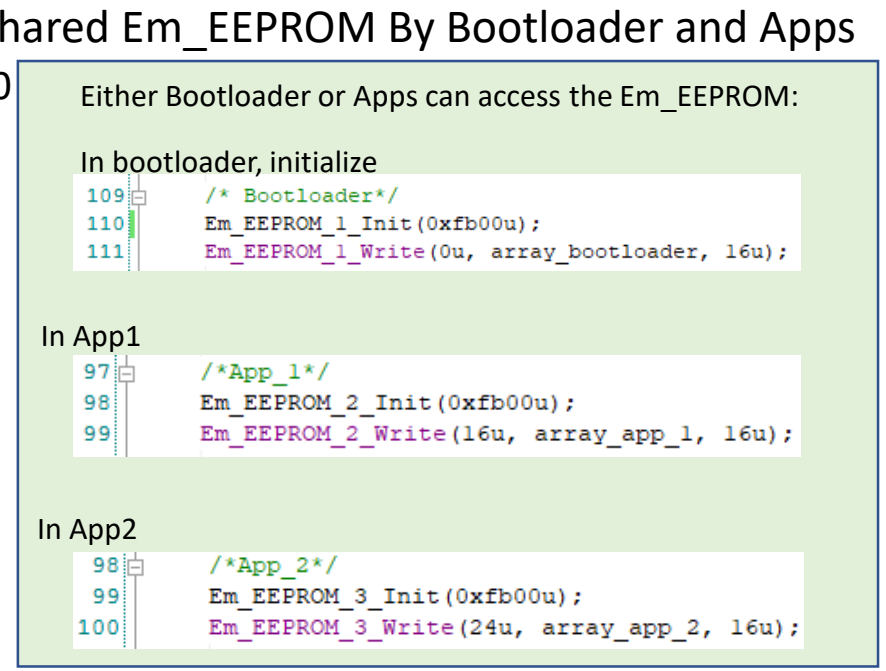
Indeed, for non-P6 devices, the array "emEeprom[Em_EEPROM_PHYSICAL_SIZE]" is ONLY used for ***Em_EEPROM_Init(uint32 startAddress)*** to specify the start address of Em_EEPROM, like below:

```
Em_EEPROM_Init((uint32_t) emEeprom);
```

Actually we can use the code following to specify the physical start address of Em_EERPOM instead:

```
Em_EEPROM_Init((uint32_t) 0xfb00);
```

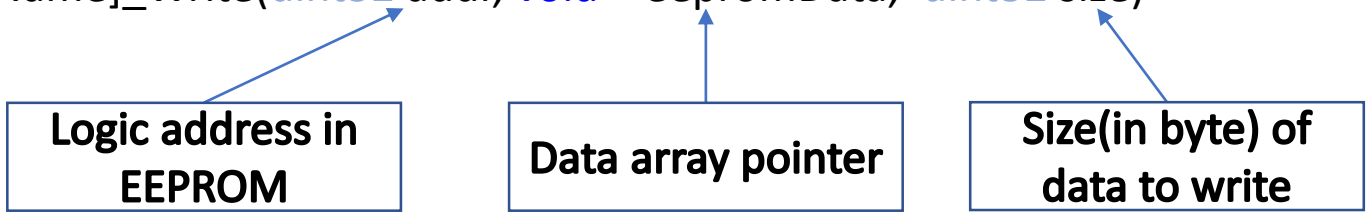
In this case, we don't need to define "emEeprom[Em_EEPROM_PHYSICAL_SIZE]" any more.



How to WRITE data in the shared Em_EEPROM in Firmware?

Writing API:

[Em_EEPROM_Instance_Name]_Write(uint32 addr, void * eepromData, uint32 size)



Writing Operation in each firmware:

In bootloader, write **16-byte(array_bootloader)** from **logic address 0** of em_eeprom(Em_EEPROM①):

```
58  uint8_t array_bootloader[16] = {0x62, 0x6c, 0x64, 0x72, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31, 0x31};
59                                 /*   b     l     d     r     1     1     1     1     1     1     1     1     1     1     1     1  */
```

```
109     /* Bootloader*/
110     Em_EEPROM_1_Init(0xfb00u);
111     Em_EEPROM_1_Write(0u, array_bootloader, 16u);
```

In App1, write **16-byte(array_app_1)** from **logic address 16** of em_eeprom(Em_EEPROM②):

```
55  uint8_t array_app_1[16] = {0x61, 0x70, 0x70, 0x31, 0x32, 0x32, 0x32, 0x32, 0x32, 0x32, 0x32, 0x32, 0x32, 0x32, 0x32, 0x32};
56                            /*   a     p     p     1     2     2     2     2     2     2     2     2     2     2     2     2  */
```

```
97      /*App_1*/
98      Em_EEPROM_2_Init(0xfb00u);
99      Em_EEPROM_2_Write(16u, array_app_1, 16u);
```

In App2, write **16-byte(array_app_2)** from **logic address 24** of em_eeprom (Em_EEPROM③):

```
55  uint8_t array_app_2[16] = {0x61, 0x70, 0x70, 0x32, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33};
56                            /*   a     p     p     2     3     3     3     3     3     3     3     3     3     3     3     3  */
```

```
 98     /*App_2*/
 99     Em_EEPROM_3_Init(0xfb00u);
100     Em_EEPROM_3_Write(24u, array_app_2, 16u);
```

How to READ data in the shared Em_EEPROM in Firmware?

Reading API:

[Em_EEPROM_Instance_Name]_Read(uint32 addr, void * eepromData, uint32 size)

Logic address in EEPROM

Data array pointer

Size(in byte) of data to write

In App2, read 16*3 byte from address 0 of em_eeprom:

```
103     Em_EEPROM_3_Read(0u, bldr_read, 16u);
104     Em_EEPROM_3_Read(16u, appl_read, 16u);
105     Em_EEPROM_3_Read(24u, app2_read, 16u);
```

In App2, print(UART) first 4-byte of the array written by bootloader, app1 and app2

```
114     for(;;)
115     {
116         UART_PutArray(bldr_read, 4u);
117         UART_PutString("\n\r");
118         UART_PutArray(appl_read, 4u);
119         UART_PutString("\n\r");
120         UART_PutArray(app2_read, 4u);
121         UART_PutString("\n\r");
```

Test Results:

Physical address storage:

Note：
In bootloader of the test project, need to tie P0[0] to VDDA to enter the device into “waiting for boot load command” mode.
Other wise Bootloader Host won’t work.