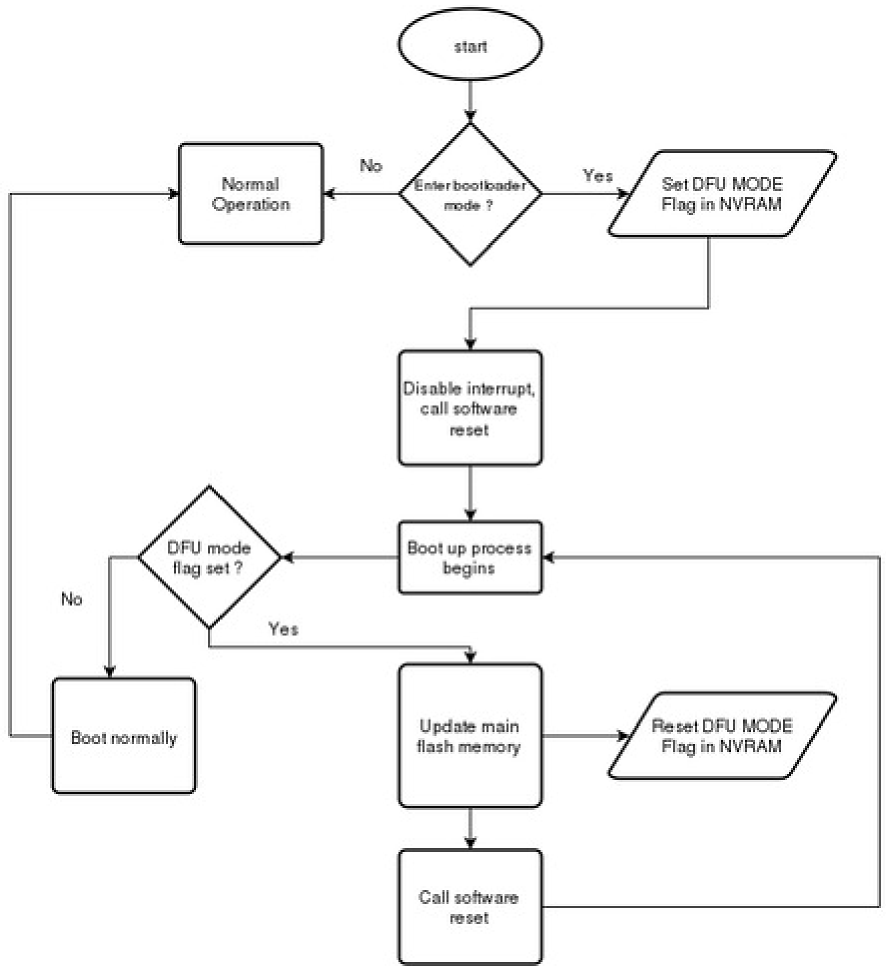
Lately, I've been trying IAP over UART using Ymodem protocol for STM32F072C8 SOC based custom board. The goal is to try IAP with this mode with different upgrade scenarios and later transform these into my actual requirement. I was able to successfully perform IAP using ST provided libraries, IAP driver and template (More information can be found Query on IAP over UART for STM32F072C8)

After this, I tried to put two different application firmware along with IAP driver in the main flash memory and was also able to jump from IAP driver to application f/w1 or f/w2. The memory layout that I'm using currently is as below:
Table1: memory map of main flash memory (Total 64 kB)

Table2: memory map of SRAM (Total 16 kB)
</colgroup>

Component	main flash memory	Block number	Size
Bootloader	0x08000000 ? 0x08003FFF	0	16 kB
fw 1	0x08004000 ? 0x08009FFF	0	24 kB
fw 2	0x0800A000 ? 0x0800FFFF	0	24 kB

Note



```
#if   (defined ( __CC_ARM ))
  __IO uint32_t VectorTable[48] __attribute__((at(0x20000000)));
#elif (defined (__ICCARM__))
#pragma location = 0x20000000
  __no_init __IO uint32_t VectorTable[48];
#elif defined   (  __GNUC__  )
```

```
  __IO uint32_t VectorTable[48] __attribute__((section(".RAMVectorTable")));
#elif defined ( __TASKING__ )
  __IO uint32_t VectorTable[48] __at(0x20000000);
#endif

void FLASH_If_Init(void)
{
  /* Unlock the Program memory */
  FLASH_Unlock();

  /* Clear all FLASH flags */
  FLASH_ClearFlag(FLASH_FLAG_EOP|FLASH_FLAG_WRPERR | FLASH_FLAG_PGERR | FLASH_FLAG_BSY);
}

int main(void)
{
  uint32_t i = 0;

  FLASH_Status status;
  for(i = 0; i < 48; i++)
  {
    VectorTable[i] = *(__IO uint32_t*)(APPLICATION_ADDRESS + (i<<2));
  }

  /* Enable the SYSCFG peripheral clock*/
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

  /* Remap SRAM at 0x00000000 */
  SYSCFG_MemoryRemapConfig(SYSCFG_MemoryRemap_SRAM);


/***************** Add your application code here   ***************************/
  FLASH_If_Init();
 /* set flag to trigger IAP after a reset. this flag will be cleared by IAP driver after a
successful update */
  status = FLASH_ProgramWord(0x0800E000, 0x1);
  while (status != FLASH_COMPLETE);

  while (1)
  {
  // do something
  }

}
```

Please advise on what could be going wrong here and also any pointers on this overall design. Basically I would like to have

1. The IAP driver do the upgrade part and application fimrwares should be only triggering IAP process by setting some flag.
2. Safely store this flag(0x1) at some address that is beyond code or data region of any of the three images
3. Flow should be System startup -> bootloader -> Initiate IAP or jump to fw1 or fw2 (based on flag value)

Environment

```
status = FLASH_ErasePage(0x0800E000);
while (status != FLASH_COMPLETE){}
status = FLASH_ProgramWord(0x0800E000, 0xDEADBEEF);
while (status != FLASH_COMPLETE){}
```

0x08004000, instead of the default location (0x08000000)? </colgroup> that I'm using the same SRAM location for both application firmware (fw1, fw2)I'm able to put the three(IAP driver, fw1, fw2) images in the main flash memory and was also able to jump from IAP driver to fw1 or fw2. Now I would like to implement the following logic (See the flow below)To achieve this, I should be able to store a variable in main flash memory at an address accessible from each of the three images (IAP driver, fw1, fw2). However, when I tried to set a flag at some address, which I think is not being utilized at the moment,in flash memory from application firmware 1 (fw1), the system hung (I think I'm getting a hard fault here, but I need to check). Code snip for storing 0x1 at some random address (note that this main flash memory address is within the overall range i.e. 0x0800 0000 - 0x0800 FFFF)SOC: STM32F072C8 64kB Flash memory (ARM cortex M0 based SOC)Main flash memory requirement: IAP driver + fw1 + fw2Application f/w size: ~20 kB (Theoratically, I should be able to fit in IAP driver and two differnt application firmware in 64kB flash memory)IDE: KEIL uVision5 for windowsIAP Mode: UARTUtilities: fromelf.exe (For converting .hex files into .bin file), ST Programmer (for In-circuit programming using SWD interface) and Hyperterminal [1]Reference document: AN4065 [2]Update #1:I'm able to program the main flash memory from the IAP driver, i.e. the following works when I invoke these in the IAP driver, but not when I try to do the same in application firmware.Also, now I'm not able to debug the application fimrware using the KEIL uVision5 debugger window. None of the single stepping, step over, etc tabs are getting activated. Do I need to configure some changes for debugger, since my application now is loaded at

Component	Physical	Block number	Size	Purpose
Bootloader	NA	NA	NA	NA
fw {1|2}	0x20000000	1	0x4000	Vector table relocation to SRAM