

## This is FIRST SWAP

Just before first SWAP

We have (check image below):

- 0x8000 = {0x11, 0x12...}
- 0x108000 = {0x00, 0x01...}

Debug Console: Thread #1 57005 (Suspended : Breakpoint) at MK26FN2M0xxx18\_Project.c:310 0x1214

```
main() {  
    // ...  
    result = FLASH_Program(Ks_FlashDriver, UPPER_PFLASH_BASE, &tempData[0], sizeof(tempData));  
    result = FLASH_Program(Ks_FlashDriver, UPPER_PFLASH_BASE + sizeof(tempData),  
        (uint8_t *) (LOWER_PFLASH_BASE + sizeof(tempData)), EXAMPLE_IMAGE_SIZE - sizeof(tempData));  
    if (kStatus_FTFx_Success != result)  
    {  
        error_trap();  
    }  
    result = FLASH_VerifyProgram(Ks_FlashDriver, UPPER_PFLASH_BASE + sizeof(tempData), EXAMPLE_IMAGE_SIZE - sizeof(tempData),  
        (uint8_t *) (LOWER_PFLASH_BASE + sizeof(tempData)), kFTFx_MarginValidUser, &failedAddress, &failedData);  
    if (kStatus_FTFx_Success != result)  
    {  
        error_trap();  
    }  
    result = FLASH_Erase(Ks_FlashDriver, UPPER_PFLASH_BASE + EXAMPLE_IMAGE_SIZE, sizeof(swapPertData),  
        kFTFx_ApiEraseKey);  
    if (kStatus_FTFx_Success != result)  
    {  
        error_trap();  
    }  
    result = FLASH_Program(Ks_FlashDriver, UPPER_PFLASH_BASE + EXAMPLE_IMAGE_SIZE, &swapPertData[0],  
        sizeof(swapPertData));  
    printf(\"\\n\\n Finish programming backup example image \\n\\n\");  
    // Enable swap system  
    printf(\"\\n\\n Start to swap the system \\n\\n\");  
    result = FLASH_Swap(Ks_FlashDriver, swapIndicatorAddress, true);  
    NVIC_SystemReset();  
}  
  
#else  
printf(\"\\n\\n Current device doesn't support flash swap feature \\n\\n\");  
app_finalize();  
#endif
```

Memory Browser: 0x8000

Address	Value
0x00000000	44332211 88776655 93915190 07969504 FFFFFFFF FFFFFFFF ... 30Uf... 9999 9999
0x00000010	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000020	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000030	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000040	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000050	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000060	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000070	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000080	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000090	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000A0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000B0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000C0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000D0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999

Memory Browser: 0x108000

Address	Value
0x00108000	03820100 07969504 000A0908 0F0E0D0C FFFFFFFF FFFFFFFF ... 30Uf... 9999 9999
0x00108010	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108020	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108030	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108040	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108050	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108060	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108070	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108080	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108090	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080A0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080B0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080C0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080D0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080E0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080F0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108100	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108110	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999

We have (check image below):

- 0x8000 = {0x00, 0x01...}
- 0x108000 = {0x11, 0x12...}

So SWAP was OK, now we are in UPPER

Debug Console: Thread #1 57005 (Suspended : Breakpoint) at MK26FN2M0xxx18\_Project.c:108 0xa6

```
main() {  
    // ...  
    ftfx_security_state_t securityStatus = kFTFx_SecurityStateNotSecure; /* Return protection status */  
    status_t result; /* Return code from each Flash driver function */  
    uint32_t pFlashTotalSize = 0;  
    uint32_t pFlashLockCount = 0;  
    uint32_t pFlashSectorSize = 0;  
    // Init hardware  
    BDM0_InitHardware();  
    // Clean up structures  
    memset(Ks_FlashDriver, 0, sizeof(FlashConfig));  
    // Setup Flash driver structure for device and initialize variables.  
    result = FLASH_Init(Ks_FlashDriver);  
    if (kStatus_FTFx_Success != result)  
    {  
        error_trap();  
    }  
    // Get Flash properties  
    FLASH_GetProperty(Ks_FlashDriver, kFLASH_PropertyPFlashTotalSize, &pFlashTotalSize);  
    FLASH_GetProperty(Ks_FlashDriver, kFLASH_PropertyPFlashLockCount, &pFlashLockCount);  
    FLASH_GetProperty(Ks_FlashDriver, kFLASH_PropertyPFlashSectorSize, &pFlashSectorSize);  
    // Print welcome message  
    printf(\"\\n\\n PFlash Swap Example Start \\n\\n\");  
    // Print flash information - PFlash  
    printf(\"\\n\\n PFlash Information: \");  
    printf(\"\\n\\n Total Program Flash Size: %d KB, Hex: (0x%08X), (pFlashTotalSize / 1024), pFlashTotalSize);  
    printf(\"\\n\\n Total Program Flash Block Count: %d, pFlashLockCount);  
    printf(\"\\n\\n Program Flash Sector Size: %d KB, Hex: (0x%08X), (pFlashSectorSize / 1024), pFlashSectorSize);  
    // Check security status  
    result = FLASH_GetSecurityState(Ks_FlashDriver, &securityStatus);  
    if (kStatus_FTFx_Success != result)  
    {  
        error_trap();  
    }  
}
```

Memory Browser: 0x8000

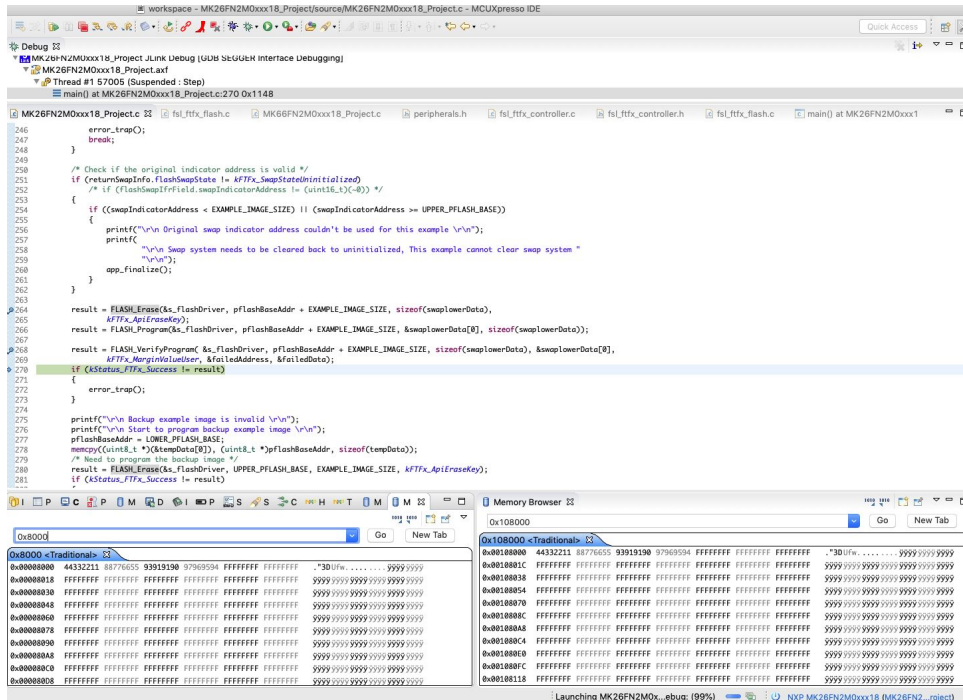
Address	Value
0x00000000	03820100 07969504 000A0908 0F0E0D0C FFFFFFFF FFFFFFFF ... 30Uf... 9999 9999
0x00000010	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000020	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000030	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000040	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000050	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000060	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000070	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000080	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00000090	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000A0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000B0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000C0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000D0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000E0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x000000F0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999

Memory Browser: 0x108000

Address	Value
0x00108000	44332211 88776655 93915190 07969504 FFFFFFFF FFFFFFFF ... 30Uf... 9999 9999
0x00108010	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108020	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108030	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108040	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108050	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108060	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108070	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108080	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108090	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080A0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080B0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080C0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080D0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080E0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x001080F0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108100	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999
0x00108110	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 9999 9999 9999 9999 9999 9999

## This is SECOND SWAP

### 1) Write 16 values at the 0x8000



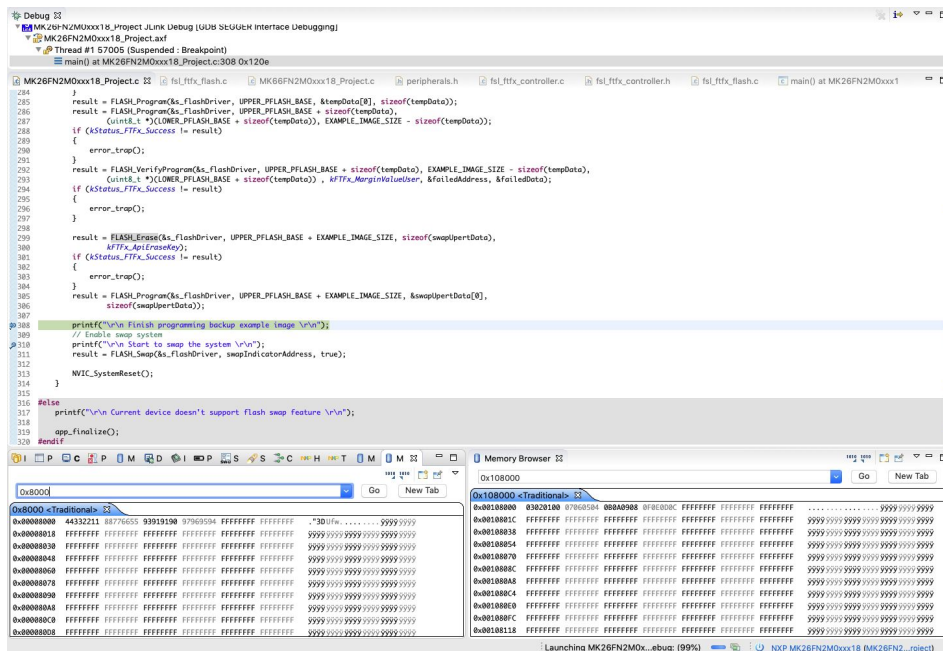
The screenshot shows the MK26FN2M0xxx18 Project IDE. The source code is displayed in the main window, and the Memory Browser is open at the bottom. The code is in C and shows the initialization of the flash driver and the writing of 16 values to the 0x8000 address. The Memory Browser shows the 0x8000 address range, with the first 16 values (0x00000000 to 0x0000000F) being written to the flash.

```
246 error_trap();
247 break;
248
249
250 /* Check if the original indicator address is valid */
251 if (ReturnSwapInfoFlashSwapState != KFTX_SwapStateNotInitialized)
252 /* If (FlashSwapIfField.swapIndicatorAddress != (uint16_t)(-1)) */
253 {
254     if ((SwapIndicatorAddress < EXAMPLE_IMAGE_SIZE) || (SwapIndicatorAddress >= UPPER_FLASH_BASE))
255     {
256         printf("\n\n Original swap indicator address couldn't be used for this example \n\n");
257         printf(
258             "\n\n Swap system needs to be cleared back to uninitialized, This example cannot clear swap system "
259             "\n\n");
260         app_finalize();
261     }
262 }
263
264 result = FLASH_Erase(&ks_flashDriver, pFlashBaseAddr + EXAMPLE_IMAGE_SIZE, sizeof(swapLowerData),
265     KFTX_ApiEraseKey);
266 result = FLASH_Program(&ks_flashDriver, pFlashBaseAddr + EXAMPLE_IMAGE_SIZE, &swapLowerData[0], sizeof(swapLowerData));
267
268 result = FLASH_VerifyProgram(&ks_flashDriver, pFlashBaseAddr + EXAMPLE_IMAGE_SIZE, sizeof(swapLowerData), &swapLowerData[0]);
269 if (KStatus_FTX_Success != result)
270 {
271     error_trap();
272 }
273
274 printf("\n\n Backup example image is invalid \n\n");
275 printf("\n\n Start to program backup example image \n\n");
276 pFlashBaseAddr = LOWER_FLASH_BASE;
277 memcpy(&uint8_t *)(&tempData[0]), (&uint8_t *)(&pFlashBaseAddr), sizeof(tempData));
278 /* Need to program the backup image */
279 result = FLASH_Erase(&ks_flashDriver, UPPER_FLASH_BASE, EXAMPLE_IMAGE_SIZE, KFTX_ApiEraseKey);
280 if (KStatus_FTX_Success != result)
281 {
282     error_trap();
283 }
284
285 result = FLASH_Program(&ks_flashDriver, UPPER_FLASH_BASE, &tempData[0], sizeof(tempData));
286 result = FLASH_VerifyProgram(&ks_flashDriver, UPPER_FLASH_BASE, sizeof(tempData), &tempData[0]);
287 if (KStatus_FTX_Success != result)
288 {
289     error_trap();
290 }
291
292 result = FLASH_VerifyProgram(&ks_flashDriver, UPPER_FLASH_BASE + sizeof(tempData), EXAMPLE_IMAGE_SIZE - sizeof(tempData),
293     (&uint8_t *)(&LOWER_FLASH_BASE + sizeof(tempData)), KFTX_MarginNoUser, &failedAddress, &failedData);
294 if (KStatus_FTX_Success != result)
295 {
296     error_trap();
297 }
298
299 result = FLASH_Erase(&ks_flashDriver, UPPER_FLASH_BASE + EXAMPLE_IMAGE_SIZE, sizeof(swapUpperData),
300     KFTX_ApiEraseKey);
301 if (KStatus_FTX_Success != result)
302 {
303     error_trap();
304 }
305 result = FLASH_Program(&ks_flashDriver, UPPER_FLASH_BASE + EXAMPLE_IMAGE_SIZE, &swapUpperData[0],
306     sizeof(swapUpperData));
307
308 printf("\n\n Finish programming backup example image \n\n");
309 // Enable swap system
310 printf("\n\n Start to swap the system \n\n");
311 result = FLASH_Swap(&ks_flashDriver, swapIndicatorAddress, true);
312
313 NVIC_SystemReset();
314
315
316 #else
317 printf("\n\n Current device doesn't support flash swap feature \n\n");
318
319 app_finalize();
320 #endif
```

Memory Browser (0x108000):

Address	Value
0x00000000	4432211 8877655 9391910 9796954 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF "30!w.....99999999
0x00000010	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000020	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000030	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000040	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000050	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000060	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000070	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000080	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000090	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000A0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000B0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000C0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000D0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000E0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000F0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000100	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000110	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000120	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999

### 2) Write 16 values at 0x10\_8000



The screenshot shows the MK26FN2M0xxx18 Project IDE. The source code is displayed in the main window, and the Memory Browser is open at the bottom. The code is in C and shows the initialization of the flash driver and the writing of 16 values to the 0x10\_8000 address. The Memory Browser shows the 0x108000 address range, with the first 16 values (0x00000000 to 0x0000000F) being written to the flash.

```
285 result = FLASH_Program(&ks_flashDriver, UPPER_FLASH_BASE, &tempData[0], sizeof(tempData));
286 result = FLASH_VerifyProgram(&ks_flashDriver, UPPER_FLASH_BASE, sizeof(tempData), &tempData[0]);
287 if (KStatus_FTX_Success != result)
288 {
289     error_trap();
290 }
291
292 result = FLASH_VerifyProgram(&ks_flashDriver, UPPER_FLASH_BASE + sizeof(tempData), EXAMPLE_IMAGE_SIZE - sizeof(tempData),
293     (&uint8_t *)(&LOWER_FLASH_BASE + sizeof(tempData)), KFTX_MarginNoUser, &failedAddress, &failedData);
294 if (KStatus_FTX_Success != result)
295 {
296     error_trap();
297 }
298
299 result = FLASH_Erase(&ks_flashDriver, UPPER_FLASH_BASE + EXAMPLE_IMAGE_SIZE, sizeof(swapUpperData),
300     KFTX_ApiEraseKey);
301 if (KStatus_FTX_Success != result)
302 {
303     error_trap();
304 }
305 result = FLASH_Program(&ks_flashDriver, UPPER_FLASH_BASE + EXAMPLE_IMAGE_SIZE, &swapUpperData[0],
306     sizeof(swapUpperData));
307
308 printf("\n\n Finish programming backup example image \n\n");
309 // Enable swap system
310 printf("\n\n Start to swap the system \n\n");
311 result = FLASH_Swap(&ks_flashDriver, swapIndicatorAddress, true);
312
313 NVIC_SystemReset();
314
315
316 #else
317 printf("\n\n Current device doesn't support flash swap feature \n\n");
318
319 app_finalize();
320 #endif
```

Memory Browser (0x108000):

Address	Value
0x00000000	4432211 8877655 9391910 9796954 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF "30!w.....99999999
0x00000010	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000020	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000030	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000040	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000050	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000060	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000070	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000080	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000090	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000A0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000B0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000C0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000D0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000E0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x000000F0	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000100	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000110	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999
0x00000120	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 999999999999999999999999

### 3) Execute SWAP command:

```
286 result = FLASH_Program(&ks_flashDriver, UPPER_PFLASH_BASE + sizeof(tempData),
287 (uint8_t *) (LOWER_PFLASH_BASE + sizeof(tempData)), EXAMPLE_IMAGE_SIZE - sizeof(tempData));
288 if (kStatus_FTFx_Success != result)
289 {
290     error_trap();
291 }
292 result = FLASH_VerifyProgram(&ks_flashDriver, UPPER_PFLASH_BASE + sizeof(tempData), EXAMPLE_IMAGE_SIZE - sizeof(tempData),
293 (uint8_t *) (LOWER_PFLASH_BASE + sizeof(tempData)), kFTFx_MarginValueUser, &failedAddress, &failedData);
294 if (kStatus_FTFx_Success != result)
295 {
296     error_trap();
297 }
298
299 result = FLASH_Erase(&ks_flashDriver, UPPER_PFLASH_BASE + EXAMPLE_IMAGE_SIZE, sizeof(swapUptData),
300 kFTFx_ApiEraseKey);
301 if (kStatus_FTFx_Success != result)
302 {
303     error_trap();
304 }
305 result = FLASH_Program(&ks_flashDriver, UPPER_PFLASH_BASE + EXAMPLE_IMAGE_SIZE, &swapUptData[0],
306 sizeof(swapUptData));
307
308 printf("\r\n Finish programming backup example image \r\n");
309 // Enable swap system
310 printf("\r\n Start to swap the system \r\n");
311 result = FLASH_Swap(&ks_flashDriver, swapIndicatorAddress, true);
312
313 NVIC_SystemReset();
314
315
316 #else
317 printf("\r\n Current device doesn't support flash swap feature \r\n");
318
319 app_finalize();
320 #endif
321
322
```

### 4) **PROBLEM:** System reset is never executed

Please note that there is a breakpoint at line 313 (NVIC\_SystemReset) but it is never reached because the SWAP state machine blocks at kFTFx\_SwapStateUpdate (it remains always in this state)

```
520
521 /* If current swap mode is Initialized/Ready, Initialize Swap to UPDATE state. */
522 returnCode =
523     FTFx_CMD_SwapControl(ftfxConfig, address, kFTFx_SwapControlOptionSetInUpdateState, &returnInfo);
524 }
525 break;
526 case kFTFx_SwapStateUpdate:
527     /* If current swap mode is Update, Erase indicator sector in non active block
528     * to proceed swap system to update-erased state */
529     returnCode = FTFx_CMD_Erase(ftfxConfig, address + (ftfxConfig->FlashDesc.totalSize >> 1),
530 ftfxConfig->opsConfig.addrAlignement.sectorEnd, kFTFx_ApiEraseKey);
531 break;
532 case kFTFx_SwapStateUpdateErased:
533     /* If current swap mode is Update or Update-Erased, progress Swap to COMPLETE State */
534     returnCode =
535         FTFx_CMD_SwapControl(ftfxConfig, address, kFTFx_SwapControlOptionSetInCompleteState, &returnInfo);
536 break;
537 case kFTFx_SwapStateComplete:
538     break;
539 case kFTFx_SwapStateDisabled:
540     /* When swap system is in disabled state, We need to clear swap system back to uninitialized
541     * by issuing EraseAllBlocks command */
542     returnCode = kStatus_FTFx_SwapSystemNotInInitialized;
543     break;
544 default:
545     returnCode = kStatus_FTFx_InvalidArgument;
546     break;
547 }
548
```