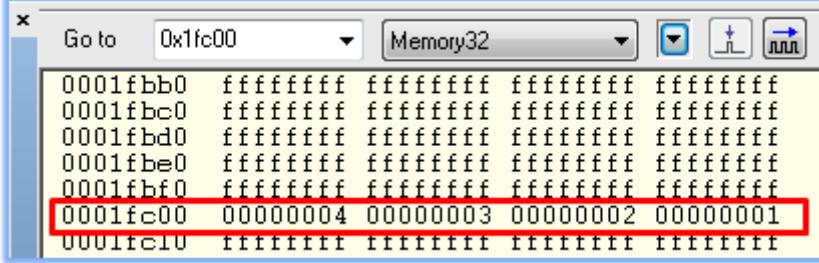


FTFA Example

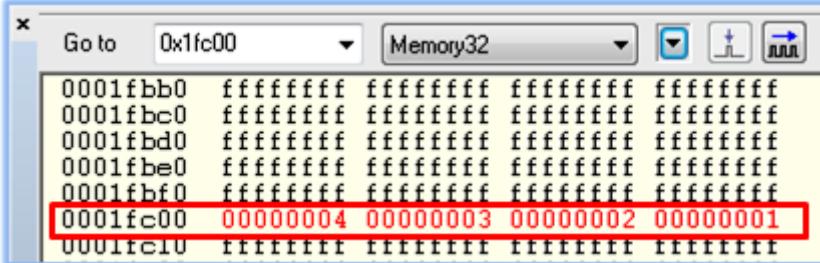
This software example demonstrates typical use of the Flash Memory Module (FTFA) module. The software initializes `nvm_data[]` flash array, placed by linker at address 0x1fc00, by the following four unsigned long values:

Flash array prior programming.



Then `FTFA_EraseSector()` function is called. It erases sector thus all 0s, within the 1KB sector pointed by address 0x1fc00, will transition to 1s. After flash sector has been erased, the function `FTFA_WriteArray()` is called. This function writes `ram_data[]` RAM array into flash at address 0x1fc00:

Flash array after programming.



Warning

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

Source code:

```
/*
 * (c) Copyright 2010-2015, Freescale Semiconductor Inc.
 * ALL RIGHTS RESERVED.
 */
/* ftfa_test.c */
#include "drivers.h"

/* this variables is stored in parameter section of the flash  addr=0x003fc00 */
#if defined (_ICCARM_)
#pragma location = ".config"
static const uint32 nvm_data[] = {11, 21, 31, 41};
#elif defined(_GNUC_) || defined(_CC_ARM)
static const uint32 __attribute__((section(".config"))) nvm_data[] = {11, 21, 31,
41};
#endif

void main (void)
{
    uint32 ram_data[] = { 41, 31, 21, 11 };

    FTFA_EraseSector (nvm_data);      /* erases flash sector; sets all 0s to 1s */
    /* write data in reverse order */
    FTFA_WriteArray (nvm_data,ram_data,sizeof(nvm_data));

    while(1);
}
```

Linker configuration file:

```
/*###IHF## Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
```

```

/*-Specials*/
define symbol __ICFEDIT_intvec_start__      = 0x00000000;
/*-Memory Regions*/
define symbol __ICFEDIT_region_ROM_start__  = 0x00000000;
define symbol __ICFEDIT_region_ROM_end__    = 0x0003ffff;
define symbol __ICFEDIT_region_RAM_start__  = 0x1ffffe000;
define symbol __ICFEDIT_region_RAM_end__    = 0x20005fff;
/*-Sizes*/
define symbol __ICFEDIT_size_cstack__       = 0x200;
define symbol __ICFEDIT_size_heap__         = 0x200;
/** End of ICF editor section. #####*/

define symbol __memcfg_start__              = 0x00000400;
define symbol __mtbram_start__              = 0x1ffffe000;
define symbol __CONFIG_FLASH_BLOCK__       = 0x0003f800;

define memory mem with size = 4G;
define region ROM_region = mem:[from __ICFEDIT_region_ROM_start__ to
                                __ICFEDIT_region_ROM_end__];
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to
                                __ICFEDIT_region_RAM_end__];

define block CSTACK      with alignment = 8, size = __ICFEDIT_size_cstack__ { };
define block HEAP        with alignment = 8, size = __ICFEDIT_size_heap__ { };

initialize manually { readwrite };
initialize manually { section .data};
initialize manually { section .textrw };
do not initialize { zeroinit };

define block CodeRelocate { section .textrw_init };
define block CodeRelocateRam { section .textrw };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };
place at address mem:__memcfg_start__          { readonly section .memcfg };
place at address mem:__mtbram_start__          { readwrite section .mtbram };
place at address mem:__CONFIG_FLASH_BLOCK__   { readonly section .config };

place in ROM_region { readonly, block CodeRelocate };
place in RAM_region { readwrite, block CodeRelocateRam, block HEAP, block CSTACK };

```

Toolchain support:

IAR EWARM 7.40.7	KEIL uVision 5.15	CrossWorks 3.6	ATOLLIC TrueStudio 5.3.0	Kinetis Design Studio 3.0.0
◆	◆	◆	◆	◆